AMD Math Libraries

OpenCL Basic Linear Algebra Subprograms
Levels 1, 2, and 3

March 2013

Revision 1.10

Advanced Micro Devices, Inc.
One AMD Place
P.O. Box 3453
Sunnyvale, CA 94088-3453
www.amd.com

For AMD Accelerated Parallel Processing:

URL:	**developer.amd.com/appsdk**
Developing:	**developer.amd.com/**
Support:	**developer.amd.com/appsdksupport**
Forum:	**developer.amd.com/openclforum**

Contents

Chapter 3 BLAS-2 Functions

Chapter 1
OpenCL BLAS Modules

1.1 Overview

This implementation of the Basic Linear Algebra Subprograms levels 1, 2, and 3 uses OpenCL and is optimized for AMD GPU hardware. It provides the following BLAS-1, BLAS-2, and BLAS-3 functions.

BLAS-1	
Function	**Precision**
SWAP	S, D, C, Z
COPY	S, D, C, Z
SCAL	S, D, C, Z
CSSCAL	
ZDSCAL	
AXPY	S, D, C, Z
DOT	S, D
DOTU	C, Z
DOTC	C, Z
ROTG	S, D, C, Z
ROTMG	S, D
ROT	S, D
CSROT	
ZDROT	
ROTM	S, D
NRM2	S, D
SCNRM2	
DZNRM2	
iAMAX	S, D, C, Z
ASUM	S, D
SCASUM	
DZASUM	

BLAS-2	
Function	**Precision**
GEMV	S, D, C, Z
SYMV	S, D
TRMV	S, D, C, Z
TRSV	S, D, C, Z
HEMV	C, Z
GER	S, D
GERU	C, Z
GERC	C, Z
HER	C, Z
HER2	C, Z
SYR	S, D
SYR2	S, D
TPMV	S, D, C, Z
TPSV	S, D, C, Z
SPMV	S, D
HPMV	C, Z
SPR	S, D
HPR	C, Z
SPR2	S, D
HPR2	C, Z
GBMV	S, D, C, Z
HBMV	C, Z
SBMV	S, D
TBMV	S, D, C, Z
TBSV	S, D, C, Z

BLAS-3	
Function	**Precision**
GEMM	S, D, C, Z
TRMM	S, D, C, Z
TRSM	S, D, C, Z
SYRK	S, D, C, Z
SYR2K	S, D, C, Z
SYMM	S, D, C, Z
HEMM	C, Z
HERK	C, Z
HER2K	C, Z

This library helps end users enqueue OpenCL kernels to process BLAS functions in an OpenCL-efficient manner, while keeping interfaces familiar for users who know how to use BLAS. All functions accept matrices through buffer objects.

Note: Scratch image buffers are deprecated, and users are advised not to use them in new applications.

1.2 Installation of clAmdBlas library

AMD provides clAmdBlas pre-compiled library packages for recent versions of Microsoft Windows operating systems and several flavors of Linux.

The downloadable binary packages are freely available from AMD at `http://developer.amd.com/libraries/appmathlibs/Pages/default.aspx`.

Once the appropriate package for the respective OS has finished downloading, uncompress the package using the native tools available on the platform in a directory of the user's choice. Everything needed to build a program using clAmdBlas is included in the directory tree, including documentation, header files, binary library components, and sample programs for programming illustration.

After the clAmdBlas package is uncompressed on the user's hard drive, a samples directory exists with source code, but no Visual Studio project files, Unix makefiles, or other native build system exist. Instead, it contains a `CMakeLists.txt` file. clAmdBlas uses CMake as its build system, and other build files, such as Visual Studio projects, NMake makefiles, or Unix makefiles, are generated by the CMake build system, during configuration. CMake is freely available for download from: `http://www.cmake.org/`

NOTE: CMake generates the native OS build files, so any changes made to the native build files are overwritten the next time CMake is run.

CMake is written to pull compiler information from environment variables, and to look in default install directories for tools. Once installed, a popular interface to control the process of creating native build files is CMake-gui. When the GUI is launched, two text boxes appear at the top of the dialog: a path to source and a separate path to generate binaries. For the `browse source...` box, find the path to where you unzipped clAmdBlas, and select the root `samples` directory that contains the `CMakeLists.txt`; for clAmdBlas, this should be `clAmdBlas/samples`. For `browse build...`, select an appropriate directory where the build environment generates build files; a convenient location is a sibling directory to the source. This makes it easy to wipe all the binaries and start a fresh build. For instance, for a debug configuration of NMake, an example directory could be `clAmdBlas/bin/NMakeDebug`. This is where the generated makefile, native build files, and intermediate object files are built. These generated files are kept separate from the source; this is referred to as 'out-of-source' builds, and is very similar in concept to what 'autotools' does for Linux. To build using NMake, simply type `NMake` in the build directory containing the makefile. To build using Visual Studio, generate the solution and project files into

a directory such as clAmdBlas/bin/vs10, find the generated .sln file, and open the solution.

The first time the configure button near the bottom of the screen is clicked, it causes CMake to prompt for what type of native build files to make. Various properties appear in red in the properties box. Red indicates that the value has changed since last time configure was clicked. (The first time configure is clicked, everything is red.) CMake tries to configure itself automatically to the client's system by looking at a systems environment variables and by searching through default install locations for project dependencies. Take a moment to verify the settings and paths that are displayed on the configuration screen; if any changes must be made, you can provide correct paths or adjust settings by typing directly into the CMake configuration screen. Click the configure button a second time to 'bake' those settings and serialize them to disk.

Options relevant to the clAmdBlas project include:

- AMDAPPSDKROOT — Location of the Stream SDK installation. This value is already populated if CMake could determine the location by looking at the environment variables. If not, the user must provide a path to the root installation of the Stream SDK here.
- CMAKE_BUILD_TYPE — Defines the build type (default is debug). For Visual Studio projects, this does not appear (modifiable in IDE); for makefile-based builds, this is set in CMake.
- CMAKE_INSTALL_PREFIX — The path to install all binaries and headers generated from the build. This is used when the user types make install or builds the INSTALL project in Visual Studio. All generated binaries and headers are copied into the path prefixed with CMAKE_INSTALL_PREFIX.

  The Visual Studio projects are self explanatory, but a few other projects are autogenerated; these might be unfamiliar.
- ALL_BUILD — A project that is empty of files, but since it depends on all user projects, it provides a convenient way to rebuild everything.
- ZERO_CHECK — A CMake-specific project that checks to see if the generated solution and project files are in sync with the CMakeLists.txt file. If these files are modified, the solutions and projects are now out-of-sync, and this project prompts the user to regenerate their environment.

Note: If the user chooses to build on Windows with a NMake based build, it is important to launch CMake from within a Visual Studio Command Prompt (20xx). This is because CMake must be able to parse environment variables to properly initialize NMake. This is not necessary if a Visual Studio solution is generated, because solution files contain their own environmental setup.

1.3 Enumerations

1.3.1 enum clAmdBlasDiag

It is used by the triangular matrix routines to specify whether the matrix is unit triangular.

- `clAmdBlasUnit` Unit triangular.
- `clAmdBlasNonUnit` Non-unit triangular.

1.3.2 enum clAmdBlasOrder

Shows how matrices are placed in memory

- `clAmdBlasRowMajor` Every row is placed sequentially
- `clAmdBlasColumnMajor` Every column is placed sequentially

1.3.3 enum clAmdBlasSide

Indicates the side matrix A is located relative to matrix B during multiplication.

- `clAmdBlasLeft` Multiply general matrix by symmetric, Hermitian or triangular matrix on the left.
- `clAmdBlasRight` Multiply general matrix by symmetric, Hermitian, or triangular matrix on the right.

1.3.4 enum clAmdBlasStatus

clAmdBlas error codes definition, incorporating OpenCL error definitions.

This enumeration is a superset of the OpenCL error codes extended with additional extra codes.

- `clAmdBlasNotImplemented` Functionality is not implemented.
- `clAmdBlasNotInitialized` clAmdBlas library is not initialized yet.
- `clAmdBlasSuccess` CL_SUCCESS.
- `clAmdBlasInvalidValue` CL_INVALID_VALUE.
- `clAmdBlasInvalidCommandQueue` CL_INVALID_COMMAND_QUEUE.
- `clAmdBlasInvalidContext` CL_INVALID_CONTEXT.
- `clAmdBlasInvalidMemObject` CL_INVALID_MEM_OBJECT.
- `clAmdBlasInvalidDevice` CL_INVALID_DEVICE.
- `clAmdBlasInvalidEventWaitList` CL_INVALID_EVENT_WAIT_LIST.
- `clAmdBlasOutOfResources` CL_OUT_OF_RESOURCES.
- `clAmdBlasOutOfHostMemory` CL_OUT_OF_HOST_MEMORY.
- `clAmdBlasInvalidOperation` CL_INVALID_OPERATION.

- clAmdBlasCompilerNotAvailable CL_COMPILER_NOT_AVAILABLE.
- clAmdBlasBuildProgramFailure CL_BUILD_PROGRAM_FAILURE.
- clAmdBlasNotImplemented Functionality is not implemented.
- clAmdBlasNotInitialized clAmdBlas library is not initialized yet.
- clAmdBlasInvalidMatA Matrix A is not a valid memory object.
- clAmdBlasInvalidMatB Matrix B is not a valid memory object.
- clAmdBlasInvalidMatC Matrix C is not a valid memory object.
- clAmdBlasInvalidVecX Vector X is not a valid memory object.
- clAmdBlasInvalidVecY Vector Y is not a valid memory object.
- clAmdBlasInvalidDim An input dimension (M,N,K) is invalid.
- clAmdBlasInvalidLeadDimA Leading dimension A must not be less than the size of the first dimension.
- clAmdBlasInvalidLeadDimB Leading dimension B must not be less than the size of the second dimension.
- clAmdBlasInvalidLeadDimC Leading dimension C must not be less than the size of the third dimension.
- clAmdBlasInvalidIncX The increment for a vector X must not be 0.
- clAmdBlasInvalidIncY The increment for a vector Y must not be 0.
- clAmdBlasInsufficientMemMatA The memory object for Matrix A is too small.
- clAmdBlasInsufficientMemMatB The memory object for Matrix B is too small.
- clAmdBlasInsufficientMemMatC The memory object for Matrix C is too small.
- clAmdBlasInsufficientMemVecX The memory object for Vector X is too small.
- clAmdBlasInsufficientMemVecY The memory object for Vector Y is too small.

1.3.5 enum clAmdBlasTranspose

It is used to specify whether the matrix is to be transposed or not.

- clAmdBlasNoTrans Operate with the matrix.
- clAmdBlasTrans Operate with the transpose of the matrix.
- clAmdBlasConjTrans Operate with the conjugate transpose of the matrix.

1.3.6 enum clAmdBlasUplo

Used by the Hermitian, symmetric, and triangular matrix routines to specify whether the upper or lower triangle is being referenced.

- `clAmdBlasUpper` Upper triangle.
- `clAmdBlasLower` Lower triangle.

1.4 Support Functions

Version information

Function `cl_int clAmdBlasGetVersion (cl_uint *major, cl_uint *minor, cl_uint *patch)`

Description Get the clAmdBlas library version info.

Parameters

out	*major*	Location to store library's major version.
out	*minor*	Location to store library's minor version.
out	*patch*	Location to store library's patch version.

Returns Always clAmdBlasSuccess.

Initialize library

Function `clAmdBlasStatus clAmdBlasSetup (void)`

Description Initialize the clAmdBlas library.
Must be called before any other clAmdBlas API function is invoked. This function is not thread safe.

Returns `clAmdBlasSucces` on success.
`clAmdBlasOutOfHostMemory` if there is not enough of memory to allocate library's internal structures.
`clAmdBlasOutOfResources` in case of requested resources scarcity.

Examples `example_sgemm.c`, `example_sgemv.c`, `example_ssymv.c`, `example_ssyr2k.c`, `example_-ssyrk.c`, `example_strmm.c`, and `example_strsm.c`.

Finalize usage of library

Function `void clAmdBlasTeardown (void)`

Description Finalize the usage of the clAmdBlas library.
Frees all memory allocated for different computational kernel and other internal data. This function is not thread safe.

Examples `example_sgemm.c`, `example_sgemv.c`, `example_ssymv.c`, `example_ssyr2k.c`, `example_-ssyrk.c`, `example_strmm.c`, and `example_strsm.c`.

Create scratch image

Function `cl_ulong clAmdBlasAddScratchImage (cl_context context, size_t width, size_t height, clAmdBlasStatus *status);`

Description This function has been deprecated

Returns A created image identifier.

Examples `example_sgemm.c`, `example_strmm.c`, and `example_strsm.c`.

Release scratch image

Function	`cl_int clAmdBlasRemoveScratchImage (cl_ulong imageID)`
Description	This function has been deprecated.
Returns	0 on success; `CL_INVALID_VALUE` if a wrong image ID is specified.
Examples	`example_sgemm.c`, `example_strmm.c`, and `example_strsm.c`.

1.5 Tools

Automatically tune the clAmdBlas library for specific hardware

Module `clAmdBlasTune`

Description This tool selects the fastest OpenCL kernels on the GPU hardware for the BLAS level 3 function. Also, it allows building the database of the given kernels in order to reduce the time needed for on-the-fly building. Optional parameters are accepted by the tool through the command line and are listed and described below. When the tool is run without parameters, it tunes all available kernels for all functions dealing with all possible data types. After the tool determines the best kernels for the task, it writes the internal state to a file named as <*device name*>.kdb. The library uses this file during runtime to get information about an optimal kernel for a specific function, as well as the kernel itself, if available. If the file is missing, the library selects a default kernel, which usually is not optimal. If the file is corrupted, the user is notified by a message output to the standard error stream. The contents and layout of this file is not public, and applications cannot assume compatibility with future releases.

The tuning process can be interrupted at any time; this tool then resumes the process from the point it was interrupted. Remember to rerun the tool after adding a new GPU device to the system or when the existing file is corrupted.

Parameters

Function-Related If any of these parameters is not specified, the tool tries kernels for all the functions.	
--gemm	Tune kernels for the GEMM function family.
--tmm	Tune kernels for the TRMM function family.
--trsm	Tune kernels for the TRSM function family.
--gemv	Tune kernels for the GEMV function family.
--symv	Tune kernels for the SYMV function family.
--syrk	Tune kernels for the SYRK function family.
--syr2k	Tune kernels for the SYR2K function family.
Used Data Types Parameters restricting data types. If multiple types are given, only the last is used.	
--float	Limits processing to single float version of functions.
--double	Limits processing to double float version of functions.
--complex	Limits processing to single complex float version of functions.
--double-complex	Limits processing to double complex float version of functions.
Kernel Generation	
--store-kernels	Stores optimal kernels in addition to the default information about them. This consumes a significant amount of disk space.
--fast	Using this option allows you to accelerate tuning in up to two or three times. Achieving optimal results is not guaranteed.
--rebuild	Re-tuning the fastest OpenCL kernels. Can be used after the driver update.

Specify the directory where configuration is saved

Module	AMD_CLBLAS_STORAGE_PATH
Description	Specifies the directory where the tuning results produced by clAmdBlasTune are saved, and where the library looks for tuning information and kernels.

Chapter 2
BLAS-1 Functions

This chapter describes the Level 1 Basic Linear Algebra functions.

2.1 xSWAP - SWAP Elements of Two Vectors

2.1.1 Sswap

Interchange two float-type vectors

Function

```
clAmdBlasStatus
clAmdBlasSswap(
     size_t N,
     cl_mem X,
     size_t offx,
     int incx,
     cl_mem Y,
     size_t offy,
     int incy,
     cl_uint numCommandQueues,
     cl_command_queue *commandQueues,
     cl_uint numEventsInWaitList,
     const cl_event *eventWaitList,
     cl_event *events);
```

Description Interchange two vectors of single-precision floating-point elements.

Parameters

in	*N*	Number of elements in vector *X*.
in/ out	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector X in buffer object. Counted in elements.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in/ out	*Y*	Buffer object storing the vector *Y*.
in	*offy*	Offset of first element of vector *Y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of *Y*. Must not be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects for each command queue that identify a particular kernel execution instance.

Interchange two float-type vectors (Cont.)

Returns

- `clAmdBlasSuccess` on success.
- `clAmdBlasNotInitialized` if `clAmdBlasSetup()` was not called.
- `clAmdBlasInvalidValue` if invalid parameters are passed:
  - N is zero, or
  - either `incx` or `incy` is zero, or
  - the vector sizes along with the increments lead to accessing outside of any of the buffers.
- `clAmdBlasInvalidMemObject` if either `X` or `Y` object is invalid, or an image object rather than the buffer one.
- `clAmdBlasOutOfHostMemory` if the library cannot allocate memory for internal structures.
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid.
- `clAmdBlasInvalidContext` if a context to which a passed command queue belongs was released.
- `clAmdBlasInvalidOperation` if the kernel compilation relating to a previous call has not completed for any of the target devices.
- `clAmdBlasCompilerNotAvailable` if a compiler is not available.
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

Examples `example_sswap.c`.

2.1.2 Dswap

Interchange two double-type vectors

Function

```
clAmdBlasStatus
clAmdBlasDswap(
    size_t N,
    cl_mem X,
    size_t offx,
    int incx,
    cl_mem Y,
    size_t offy,
    int incy,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description Interchange two vectors of double-precision floating point elements.

Parameters

in	*N*	Number of elements in vector *X*.
in/ out	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in/ out	*Y*	Buffer object storing the vector *Y*.
in	*offy*	Offset of first element of vector *Y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of *Y*. Must not be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects for each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support the floating point arithmetic with double precision;
- otherwise, the same error codes as the `clAmdBlasSswap()` function.

2.1.3 Cswap

Interchange two vectors of complex-float elements

Function

```
clAmdBlasStatus
clAmdBlasCswap(
    size_t N,
    cl_mem X,
    size_t offx,
    int incx,
    cl_mem Y,
    size_t offy,
    int incy,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description Interchange two vectors of complex single-precision floating point elements.

Parameters

in	*N*	Number of elements in vector *X*.
in/ out	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in/ out	*Y*	Buffer object storing the vector *Y*.
in	*offy*	Offset of first element of vector *Y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of *Y*. Must not be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects in each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- otherwise, the same error codes as the `clAmdBlasSwap()` function.

2.1.4 Zswap

Interchanges two vectors of double-complex elements

Function

```
clAmdBlasStatus
clAmdBlasZswap(
    size_t N,
    cl_mem X,
    size_t offx,
    int incx,
    cl_mem Y,
    size_t offy,
    int incy,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description Interchange two vectors of complex double-precision floating point elements.

Parameters

in	*N*	Number of elements in vector *X*.
in/ out	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in/ out	*Y*	Buffer object storing the vector *Y*.
in	*offy*	Offset of first element of vector *Y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of *Y*. Must not be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects in each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- otherwise, the same error codes as the `clAmdBlasDswap()` function.

2.2 xSCAL - SCALe a Vector by a Constant

2.2.1 Sscal

Scale a float-type vector by a float constant

Function

```
clAmdBlasStatus
clAmdBlasSscal(
    size_t N,
    cl_float alpha,
    cl_mem X,
    size_t offx,
    int incx,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description $X \leftarrow \alpha X$

Parameters

in	*N*	Number of columns in matrix *A*.
in	*alpha*	The factor of vector *X*.
in/ out	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects in each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasNotInitialized` if `clAmdBlasSetup()` was not called.
- `clAmdBlasInvalidValue` if invalid parameters are passed:
  - *N* is zero, or
  - *incx* zero, or
  - the vector sizes along with the increments lead to accessing outside of any of the buffers
- `clAmdBlasInvalidMemObject` if either *X,* or *Y* object is Invalid, or an image object rather than the buffer one.
- `clAmdBlasOutOfHostMemory` if the library cannot allocate memory for internal structures.
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid.
- `clAmdBlasInvalidContext` if a context to which a passed command queue belongs was released.
- clAmdBlasInvalidOperation if the kernel compilation relating to a previous call has not completed for any of the target devices.
- `clAmdBlasCompilerNotAvailable` if a compiler is not available.
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

Examples `example_sscal.c`.

2.2.2 Dscal

Scale a double-type vector by a double constant

Function

```
clAmdBlasStatus
clAmdBlasDscal(
    size_t N,
    cl_double alpha,
    cl_mem X,
    size_t offx,
    int incx,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
     cl_event *events);
```

Description

$X \leftarrow \alpha X$

Parameters

in	*N*	Number of columns in matrix *A*.
in	*alpha*	The factor of matrix *A*.
in/ out	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects in each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasSuccess` on success; floating point arithmetic with double precision;
- otherwise, the same error codes as the `clAmdBlasSscal()` function.

2.2.3 Cscal

Scale a complex-float vector by a complex-float constant

Function

```
clAmdBlasStatus
clAmdBlasCscal(
    size_t N,
    cl_float2 alpha,
    cl_mem X,
    size_t offx,
    int incx,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description $X \leftarrow \alpha X$

Parameters

in	*N*	Number of elements in vector *X*.
in	*alpha*	The constant factor for vector *X*.
in/ out	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects in each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- otherwise, the same error codes as the `clAmdBlasSscal()` function.

2.2.4 Zscal

Scale a complex-double vector by a complex-double constant

Function

```
clAmdBlasStatus
clAmdBlasZscal(
    size_t N,
    cl_double2 alpha,
    cl_mem X,
    size_t offx,
    int incx,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description $X \leftarrow \alpha X$

Parameters

in	*N*	Number of elements in vector *X*.
in	*alpha*	The constant factor for vector *X*.
in/ out	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects in each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- otherwise, the same error codes as the `clAmdBlasDscal()` function.

2.3 xySCAL - Scale a Complex Vector by a Real Constant

2.3.1 Csscal

Scale a complex-float vector by a float constant

Function

```
clAmdBlasStatus
clAmdBlasCsscal(
    size_t N,
    cl_float alpha,
    cl_mem X,
    size_t offx,
    int incx,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description $X \leftarrow \alpha X$

Parameters

in	*N*	Number of elements in vector *X*.
in	*alpha*	The constant factor for vector *X*.
in/ out	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects for each command queue that identify a particular kernel execution instance.

Scale a complex-float vector by a float constant (Cont.)

Returns

- `clAmdBlasSuccess` on success.
- `clAmdBlasNotInitialized` if `clAmdBlasSetup()` was not called.
- `clAmdBlasInvalidValue` if invalid parameters are passed:
  - N is zero, or
  - incx is zero, or
  - the vector sizes along with the increments lead to accessing outside of any of the buffers.
- `clAmdBlasInvalidMemObject` if either X, or Y object is invalid, or an image object rather than the buffer one.
- `clAmdBlasOutOfHostMemory` if the library cannot allocate memory for internal structures.
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid.
- `clAmdBlasInvalidContext` if a context to which a passed command queue belongs was released.
- `clAmdBlasInvalidOperation` if the kernel compilation relating to a previous call has not completed for any of the target devices.
- `clAmdBlasCompilerNotAvailable` if a compiler is not available.
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

Examples `example_csscal.c`

2.3.2 Zdscal

Scale a complex-double vector by a double constant

Function

```
clAmdBlasStatus
clAmdBlasZdscal(
    size_t N,
    cl_double alpha,
    cl_mem X,
    size_t offx,
    int incx,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description

$X \leftarrow \alpha X$

Parameters

in	*N*	Number of elements in vector *X*.
in	*alpha*	The constant factor of vector *X*.
in/ out	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects for each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support the floating point arithmetic with double precision;
- otherwise, the same error codes as the `clAmdBlasCsscal()` function.

2.4 xCOPY - COPY Elements from Vector X to Vector Y

2.4.1 Scopy

Copy float elements from vector X to vector Y

Function

```
clAmdBlasStatus
clAmdBlasScopy(
    size_t N,
    const cl_mem X,
    size_t offx,
    int incx,
    cl_mem Y,
    size_t offy,
    int incy,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
     cl_event *events);
```

Description $Y \leftarrow X$

Parameters

in	*N*	Number of elements in vector *X*.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of vector *X*. Must not be zero.
out	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset of first element of vector *Y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of vector *Y*. Must not be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects in each command queue that identify a particular kernel execution instance.

Copy float elements from vector X to vector Y (Cont.)

Returns

- `clAmdBlasSuccess` on success.
- `clAmdBlasNotInitialized` if `clAmdBlasSetup()` was not called.
- `clAmdBlasInvalidValue` if invalid parameters are passed:
  - *N* is zero, or
  - either *incx* or *incy* is zero, or
  - the vector sizes along with the increments lead to accessing outside of any of the buffers.
- `clAmdBlasInvalidMemObject` if either *X* or *Y* object is invalid, or an image object rather than the buffer one.
- `clAmdBlasOutOfHostMemory` if the library cannot allocate memory for internal structures.
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid.
- `clAmdBlasInvalidContext` if a context to which a passed command queue belongs was released.
- `clAmdBlasInvalidOperation` if the kernel compilation relating to a previous call has not completed for any of the target devices.
- `clAmdBlasCompilerNotAvailable` if a compiler is not available.
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

Examples `example_scopy.c`

2.4.2 Dcopy

Copy double elements from vector X to vector Y

Function

```
clAmdBlasStatus
clAmdBlasDcopy(
    size_t N,
    const cl_mem X,
    size_t offx,
    int incx,
    cl_mem Y,
    size_t offy,
    int incy,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
     cl_event *events);
```

Description

$Y \leftarrow X$

Parameters

in	*N*	Number of rows and columns in matrix *A*.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of vector *X*. Must not be zero.
out	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset of first element of vector *Y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of vector *Y*. Must not be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects in each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support the floating point arithmetic with double precision;
- otherwise, the same error codes as the `clAmdBlasScopy()` function.

2.4.3 Ccopy

Copy complex float elements from vector X to vector Y

Function

```
clAmdBlasStatus
clAmdBlasCcopy(
    size_t N,
    const cl_mem X,
    size_t offx,
    int incx,
    cl_mem Y,
    size_t offy,
    int incy,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
     cl_event  *events);
```

Description $Y \leftarrow X$

Parameters

in	*N*	Number of elements in vector *X*.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of vector *X*. Must not be zero.
out	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset of first element of vector *Y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of vector *Y*. Must not be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects in each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- otherwise, the same error codes as the `clAmdBlasScopy()` function.

2.4.4 Zcopy

Copy complex-double elements from vector X to vector Y

Function

```
clAmdBlasStatus
clAmdBlasZcopy(
    size_t N,
    const cl_mem X,
    size_t offx,
    int incx,
    cl_mem Y,
    size_t offy,
    int incy,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
     cl_event *events);
```

Description $Y \leftarrow X$

Parameters

in	*N*	Number of elements in vector *X*.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of vector *X*. Must not be zero.
out	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset of first element of vector *Y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of vector *Y*. Must not be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects in each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- otherwise, the same error codes as the `clAmdBlasDcopy()` function.

2.5 xAXPY - Scale X and Add to Y

2.5.1 Saxpy

Scale vector X of float elements and add to Y

Function

```
clAmdBlasStatus
clAmdBlasSaxpy(
    size_t N,
    cl_float alpha,
    const cl_mem X,
    size_t offx,
    int incx,
    cl_mem Y,
    size_t offy,
    int incy,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description $Y \leftarrow \alpha X + Y$

Parameters

in	*N*	Number of elements in vector *X*.
in	*alpha*	The factor of matrix *A*.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of vector *X*. Must not be zero.
in/ out	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset of first element of vector *Y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of vector *Y*. Must not be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects in each command queue that identify a particular kernel execution instance.

Scale vector X of float elements and add to Y (Cont.)

Returns

- `clAmdBlasSuccess` on success.
- `clAmdBlasNotInitialized` if `clAmdBlasSetup()` was not called.
- `clAmdBlasInvalidValue` if invalid parameters are passed:
  - *N* is zero, or
  - either *incx* or *incy* is zero, or
  - the vector sizes along with the increments lead to accessing outside of any of the buffers.
- `clAmdBlasInvalidMemObject` if either *X* or *Y* object is invalid, or an image object rather than the buffer one.
- `clAmdBlasOutOfHostMemory` if the library cannot allocate memory for internal structures.
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid.
- `clAmdBlasInvalidContext` if a context to which a passed command queue belongs was released.
- `clAmdBlasInvalidOperation` if the kernel compilation relating to a previous call has not completed for any of the target devices.
- `clAmdBlasCompilerNotAvailable` if a compiler is not available.
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

Examples `example_saxpy.c`

2.5.2 Daxpy

Scale vector X of double elements and add to Y

Function

```
clAmdBlasStatus
clAmdBlasDaxpy(
    size_t N,
    cl_double alpha,
    const cl_mem X,
    size_t offx,
    int incx,
    cl_mem Y,
    size_t offy,
    int incy,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description $Y \leftarrow \alpha X + Y$

Parameters

in	*N*	Number of elements in vector *X*.
in	*alpha*	The constant factor for vector *X*.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of vector *X*. Must not be zero.
in/ out	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset of first element of vector *Y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of vector *Y*. Must not be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects in each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support the floating point arithmetic with double precision;
- otherwise, the same error codes as the `clAmdBlasSaxpy()` function.

2.5.3 Caxpy

Scale vector X of complex-float elements and add to Y

Function

```
clAmdBlasStatus
clAmdBlasCaxpy(
    size_t N,
    cl_float2 alpha,
    const cl_mem X,
    size_t offx,
    int incx,
    cl_mem Y,
    size_t offy,
    int incy,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description

$Y \leftarrow \alpha X + Y$

Parameters

in	*N*	Number of elements in vector *X*.
in	*alpha*	The constant factor for vector *X*.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of vector *X*. Must not be zero.
in/ out	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset of first element of vector *Y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of vector *Y*. Must not be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects in each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- otherwise, the same error codes as the `clAmdBlasSaxpy()` function.

2.5.4 Zaxpy

Scale vector X of double-complex elements and add to Y

Function

```
clAmdBlasStatus
clAmdBlasZaxpy(
    size_t N,
    cl_double2 alpha,
    const cl_mem X,
    size_t offx,
    int incx,
    cl_mem Y,
    size_t offy,
    int incy,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description $Y \leftarrow \alpha X + Y$

Parameters

in	*N*	Number of elements in vector *X*.
in	*alpha*	The constant factor for vector *X*.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of vector *X*. Must not be zero.
in/ out	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset of first element of vector *Y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of vector *Y*. Must not be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects in each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- otherwise, the same error codes as the `clAmdBlasDaxpy()` function.

2.6 DOT - Dot Product of Two Vectors

2.6.1 Sdot

DOT product of two vectors containing float elements

Function

```
clAmdBlasStatus
clAmdBlasSdot(
    size_t N,
    cl_mem dotProduct,
    size_t offDP,
    const cl_mem X,
    size_t offx,
    int incx,
    const cl_mem Y,
    size_t offy,
    int incy,
    cl_mem scratchBuff,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description DOT product of two vectors containing float elements.

Parameters

in	*N*	Number of elements in vector *X*.
out	*dotProduct*	Buffer object to contain the dot-product value.
in	*offDP*	Offset to dot-product in the *dotProduct* buffer object. Counted in elements.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of vector *X*. Must not be zero.
in	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset of first element of vector *Y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of vector *Y*. Must not be zero.
in	*scratchBuff*	Temporary cl_mem scratch buffer object of minimum size N.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects for each command queue that identify a particular kernel execution instance.

DOT product of two vectors containing float elements (Cont.)

Returns

- `clAmdBlasSuccess` on success.
- `clAmdBlasNotInitialized` if `clAmdBlasSetup()` was not called.
- `clAmdBlasInvalidValue` if invalid parameters are passed:
  - *N* is zero, or
  - either *incx* or *incy* is zero, or
  - the vector sizes along with the increments lead to accessing outside of any of the buffers.
- `clAmdBlasInvalidMemObject` if either *X* or *Y* object is invalid, or an image object rather than the buffer one.
- `clAmdBlasOutOfHostMemory` if the library cannot allocate memory for internal structures.
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid.
- `clAmdBlasInvalidContext` if a context to which a passed command queue belongs was released.
- `clAmdBlasInvalidOperation` if the kernel compilation relating to a previous call has not completed for any of the target devices.
- `clAmdBlasCompilerNotAvailable` if a compiler is not available.
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

Examples `example_sdot.c`

2.6.2 Ddot

DOT product of two vectors containing double elements

Function

```
clAmdBlasStatus
clAmdBlasDdot(
    size_t N,
    cl_mem dotProduct,
    size_t offDP,
    const cl_mem X,
    size_t offx,
    int incx,
    const cl_mem Y,
    size_t offy,
    int incy,
    cl_mem scratchBuff,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description DOT product of two vectors containing double elements.

Parameters

in	*N*	Number of elements in vector *X*.
out	*dotProduct*	Buffer object to contain the dot-product value.
in	*offDP*	Offset to dot-product in *dotProduct* buffer object. Counted in elements.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of vector *X*. Must not be zero.
in	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset of first element of vector *Y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of vector *Y*. Must not be zero.
in	*scratchBuff*	Temporary cl_mem scratch buffer object of minimum size N.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects for each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support the floating point arithmetic with double precision;
- otherwise, the same error codes as the `clAmdBlasSdot()` function.

2.6.3 Cdotu

DOT product of two vectors containing float-complex elements

Function

```
clAmdBlasStatus
clAmdBlasCdotu(
    size_t N,
    cl_mem dotProduct,
    size_t offDP,
    const cl_mem X,
    size_t offx,
    int incx,
    const cl_mem Y,
    size_t offy,
    int incy,
    cl_mem scratchBuff,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description DOT product of two vectors containing float-complex elements.

Parameters

in	*N*	Number of elements in vector *X*.
out	*dotProduct*	Buffer object to contain the dot-product value.
in	*offDP*	Offset to dot-product in *dotProduct* buffer object. Counted in elements.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of vector *X*. Must not be zero.
in	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset of first element of vector *Y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of vector *Y*. Must not be zero.
in	*scratchBuff*	Temporary cl_mem scratch buffer object of minimum size N.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects for each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- otherwise, the same error codes as the `clAmdBlasSdot()` function.

2.6.4 Zdotu

DOT product of two vectors containing double-complex elements

Function

```
clAmdBlasStatus
clAmdBlasZdotu(
    size_t N,
    cl_mem dotProduct,
    size_t offDP,
    const cl_mem X,
    size_t offx,
    int incx,
    const cl_mem Y,
    size_t offy,
    int incy,
    cl_mem scratchBuff,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description DOT product of two vectors containing double-complex elements.

Parameters

in	*N*	Number of elements in vector *X*.
out	*dotProduct*	Buffer object to contain the dot-product value.
in	*offDP*	Offset to dot-product in *dotProduct* buffer object. Counted in elements.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of vector *X*. Must not be zero.
in	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset of first element of vector *Y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of vector *Y*. Must not be zero.
in	*scratchBuff*	Temporary cl_mem scratch buffer object of minimum size N.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects for each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support the floating point arithmetic with double precision;
- otherwise, the same error codes as the `clAmdBlasSdot()` function.

2.6.5 Cdotc

DOT product of two vectors containing float-complex elements conjugating the first vector

Function

```
clAmdBlasStatus
clAmdBlasCdotc(
    size_t N,
    cl_mem dotProduct,
    size_t offDP,
    const cl_mem X,
    size_t offx,
    int incx,
    const cl_mem Y,
    size_t offy,
    int incy,
    cl_mem scratchBuff,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description DOT product of two vectors containing float-complex elements conjugating the first vector.

Parameters

in	*N*	Number of elements in vector *X*.
out	*dotProduct*	Buffer object to contain the dot-product value.
in	*offDP*	Offset to dot-product in *dotProduct* buffer object. Counted in elements.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of vector *X*. Must not be zero.
in	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset of first element of vector *Y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of vector *Y*. Must not be zero.
in	*scratchBuff*	Temporary cl_mem scratch buffer object of minimum size N.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects for each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- otherwise, the same error codes as the `clAmdBlasSdot()` function.

2.6.6 Zdotc

DOT product of two vectors containing double-complex elements conjugating the first vector

Function

```
clAmdBlasStatus
clAmdBlasZdotc(
    size_t N,
    cl_mem dotProduct,
    size_t offDP,
    const cl_mem X,
    size_t offx,
    int incx,
    const cl_mem Y,
    size_t offy,
    int incy,
    cl_mem scratchBuff,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description DOT product of two vectors containing double-complex elements conjugating the first vector.

Parameters

in	*N*	Number of elements in vector *X*.
out	*dotProduct*	Buffer object to contain the dot-product value.
in	*offDP*	Offset to dot-product in *dotProduct* buffer object. Counted in elements.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of vector *X*. Must not be zero.
in	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset of first element of vector *Y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of vector *Y*. Must not be zero.
in	*scratchBuff*	Temporary cl_mem scratch buffer object of minimum size N.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects for each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support the floating point arithmetic with double precision;
- otherwise, the same error codes as the `clAmdBlasSdot()` function.

2.7 xROTG - Constructs Givens Plane ROTation

2.7.1 Srotg

Construct Givens plane rotation on float elements

Function

```
clAmdBlasStatus
clAmdBlasSrotg(
    cl_mem SA,
    size_t offSA,
    cl_mem SB,
    size_t offSB,
    cl_mem C,
    size_t offC,
    cl_mem S,
    size_t offS,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description Construct Givens plane rotation on float elements.

Parameters

in/ out	*SA*	Buffer object that contains *SA*.
in	*offSA*	Offset to SA in *SA* buffer object. Counted in elements.
in/ out	*SB*	Buffer object that contains *SB*.
in	*offSB*	Offset to SB in *SB* buffer object. Counted in elements.
out	*C*	Buffer object that contains *C*.
in	*offC*	Offset to C in *C* buffer object. Counted in elements.
out	*S*	Buffer object that contains *S*.
in	*offS*	Offset to S in *S* buffer object. Counted in elements.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects for each command queue that identify a particular kernel execution instance.

Construct Givens plane rotation on float elements (Cont.)

Returns

- clAmdBlasSuccess on success;
- clAmdBlasNotInitialized if clAmdBlasSetup() was not called;
- clAmdBlasInvalidMemObject if either *SA*, *SB*, *C*, or *S* object is Invalid, or an image object rather than the buffer one.
- clAmdBlasOutOfHostMemory if the library can't allocate memory for internal structures;
- clAmdBlasInvalidCommandQueue if the passed command queue is invalid;
- clAmdBlasInvalidContext if a context to which a passed command queue belongs was released;
- clAmdBlasInvalidOperation if the kernel compilation relating to a previous call has not completed for any of the target devices;
- clAmdBlasCompilerNotAvailable if a compiler is not available;
- clAmdBlasBuildProgramFailure if there is a failure to build a program executable.

Examples example_srotg.c

2.7.2 Drotg

Construct Givens plane rotation on double elements

Function

```
clAmdBlasStatus
clAmdBlasDrotg(
    cl_mem DA,
    size_t offDA,
    cl_mem DB,
    size_t offDB,
    cl_mem C,
    size_t offC,
    cl_mem S,
    size_t offS,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description Construct Givens plane rotation on double elements.

Parameters

in/ out	*DA*	Buffer object that contains *DA*.
in	*offDA*	Offset to DA in *DA* buffer object. Counted in elements.
in/ out	*DB*	Buffer object that contains *DB*.
in	*offDB*	Offset to DB in *DB* buffer object. Counted in elements.
out	*C*	Buffer object that contains *C*.
in	*offC*	Offset to C in *C* buffer object. Counted in elements.
out	*S*	Buffer object that contains *S*.
in	*offS*	Offset to S in *S* buffer object. Counted in elements.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects for each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support the floating point arithmetic with double precision.
- otherwise, same error codes as the `clAmdBlasSrotg()` function.

2.7.3 Crotg

Construct Givens plane rotation on float-complex elements

Function

```
clAmdBlasStatus
clAmdBlasCrotg(
    cl_mem CA,
    size_t offCA,
    cl_mem CB,
    size_t offCB,
    cl_mem C,
    size_t offC,
    cl_mem S,
    size_t offS,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description Construct Givens plane rotation on float-complex elements.

Parameters

in/ out	*CA*	Buffer object that contains *CA*.
in	*offCA*	Offset to CA in *CA* buffer object. Counted in elements.
in/ out	*CB*	Buffer object that contains *CB*.
in	*offCB*	Offset to CB in *CB* buffer object. Counted in elements.
out	*C*	Buffer object that contains *C*.
in	*offC*	Offset to C in *C* buffer object. Counted in elements.
out	*S*	Buffer object that contains *S*.
in	*offS*	Offset to S in *S* buffer object. Counted in elements.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects for each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- otherwise, same error codes as the `clAmdBlasSrotg()` function.

2.7.4 Zrotg

Construct Givens plane rotation on double-complex elements

Function

```
clAmdBlasStatus
clAmdBlasZrotg(
    cl_mem CA,
    size_t offCA,
    cl_mem CB,
    size_t offCB,
    cl_mem C,
    size_t offC,
    cl_mem S,
    size_t offS,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description Construct Givens plane rotation on double-complex elements.

Parameters

in/ out	*CA*	Buffer object that contains *CA*.
in	*offCA*	Offset to CA in *CA* buffer object. Counted in elements.
in/ out	*CB*	Buffer object that contains *CB*.
in	*offCB*	Offset to CB in *CB* buffer object. Counted in elements.
out	*C*	Buffer object that contains *C*.
in	*offC*	Offset to C in *C* buffer object. Counted in elements.
out	*S*	Buffer object that contains *S*.
in	*offS*	Offset to S in *S* buffer object. Counted in elements.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects for each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- otherwise, same error codes as the `clAmdBlasDrotg()` function.

2.8 xROTMG - Construct the Modified Givens ROTation

2.8.1 Srotmg

Construct the modified Givens rotation on float elements

Function

```
clAmdBlasStatus
clAmdBlasSrotmg(
    cl_mem SD1,
    size_t offSD1,
    cl_mem SD2,
    size_t offSD2,
    cl_mem SX1,
    size_t offSX1,
    const cl_mem SY1,
    size_t offSY1,
    cl_mem SPARAM,
    size_t offSparam,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description Construct the modified Givens rotation on float elements.

Parameters

in/ out	*SD1*	Buffer object that contains SD1.
in	*offSD1*	Offset to SD1 in *SD1* buffer object. Counted in elements.
in/ out	*SD2*	Buffer object that contains SD2.
in	*offSD2*	Offset to SD2 in *SD2* buffer object. Counted in elements.
in/ out	*SX1*	Buffer object that contains SX1.
in	*offSX1*	Offset to SX1 in *SX1* buffer object. Counted in elements.
in	*SY1*	Buffer object that contains SY1.
in	*offSY1*	Offset to SY1 in *SY1* buffer object. Counted in elements.

in/ out	*SPARAM*	Buffer object that contains SPARAM array of minimum length 5.
		`SPARAM(0) = SFLAG  SPARAM(3) = SH12`
		`SPARAM(1) = SH11  SPARAM(4) = SH22`
		`SPARAM(2) = SH21`

in	*offSparam*	Offset to SPARAM in SPARAM buffer object. Counted in elements.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects for each command queue that identify a particular kernel execution instance.

Construct the modified Givens rotation on float elements (Cont.)

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasNotInitialized` if `clAmdBlasSetup()` was not called;
- `clAmdBlasInvalidMemObject` if either *SX1*, *SY1*, *SD1*, *SD2*, or *SPARAM* object is Invalid, or an image object rather than the buffer one.
- `clAmdBlasInvalidMemObject` if either *A* or *X* object is Invalid, or an image object rather than the buffer one;
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures;
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid;
- `clAmdBlasInvalidContext` if a context to which a passed command queue belongs was released;
- `clAmdBlasInvalidOperation` if the kernel compilation relating to a previous call has not completed for any of the target devices;
- `clAmdBlasCompilerNotAvailable` if a compiler is not available;
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

Examples `example_srotmg.c`

2.8.2 Drotmg

Construct the modified Givens rotation on double elements

Function

```
clAmdBlasStatus
clAmdBlasDrotmg(
    cl_mem DD1,
    size_t offDD1,
    cl_mem DD2,
    size_t offDD2,
    cl_mem DX1,
    size_t offDX1,
    const cl_mem DY1,
    size_t offDY1,
    cl_mem DPARAM,
    size_t offDparam,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description Construct the modified Givens rotation on double elements.

Construct the modified Givens rotation on double elements (Cont.)

Parameters

in/ out	*DD1*	Buffer object that contains DD1.
in	*offDD1*	Offset to DD1 in *DD1* buffer object. Counted in elements.
in/ out	*DD2*	Buffer object that contains DD2.
in	*offDD2*	Offset to DD2 in \b DD2 buffer object. Counted in elements.
in/ out	*DX1*	Buffer object that contains DX1.
in	*offDX1*	Offset to DX1 in *DX1* buffer object. Counted in elements.
in	*DY1*	Buffer object that contains DY1.
in	*offDY1*	Offset to DY1 in DY1 buffer object. Counted in elements.

in/ out	*DPARAM*	Buffer object that contains DPARAM array of minimum length 5.
		`DPARAM(0) = DFLAG  DPARAM(3) = DH12`
		`DPARAM(1) = DH11  DPARAM(4) = DH22`
		`DPARAM(2) = DH21`

in	*offDparam*	Offset to DPARAM in DPARAM buffer object. Counted in elements.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects for each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support the floating point arithmetic with double precision.
- otherwise, same error codes as the `clAmdBlasSrotmg()` function.

2.9 ROT - Apply Givens ROTation

2.9.1 Srot

Apply plane rotation on float elements

Function

```
clAmdBlasStatus
clAmdBlasSrot(
    size_t N,
    cl_mem X,
    size_t offx,
    int incx,
    cl_mem Y,
    size_t offy,
    int incy,
    cl_float C,
    cl_float S,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description Apply plane rotation on float elements.

Parameters

in	*N*	Number of elements of vector *X*.
in/ out	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of vector *X*. Must not be zero.
in/ out	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset of first element of vector *Y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of vector *Y*. Must not be zero.
in	*C*	Specifies the cosine, cos.
in	*S*	Specifies the sine, sin.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects for each command queue that identify a particular kernel execution instance.

Apply plane rotation on float elements (Cont.)

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasNotInitialized` if `clAmdBlasSetup()` was not called.
- `clAmdBlasInvalidValue` if invalid parameters are passed:
  - either *N* or *incx* is zero, or
  - the vector sizes along with the increments lead to accessing outside of any of the buffers.
- `clAmdBlasInvalidMemObject` if either *A* or *X* object is Invalid, or an image object rather than the buffer one;
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures;
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid;
- `clAmdBlasInvalidContext` if a context to which a passed command queue belongs was released;
- `clAmdBlasInvalidOperation` if the kernel compilation relating to a previous call has not completed for any of the target devices;
- `clAmdBlasCompilerNotAvailable` if a compiler is not available;
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

Examples `example_srot.c`

2.9.2 Drot

Apply plane rotation on double elements

Function

```
clAmdBlasStatus
clAmdBlasDrot(
    size_t N,
    cl_mem X,
    size_t offx,
    int incx,
    cl_mem Y,
    size_t offy,
    int incy,
    cl_double C,
    cl_double S,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description Apply plane rotation on double elements.

Parameters

in	*N*	Number of elements of vector *X*.
in/ out	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of vector *X*. Must not be zero.
in/ out	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset of first element of vector *Y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of vector *Y*. Must not be zero.
in	*C*	Specifies the cosine, cos.
in	*S*	Specifies the sine, sin.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects for each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support the floating point arithmetic with double precision;
- otherwise, the same error code as the `clAmdBlasSrot()` function.

2.9.3 Csrot

Apply plane rotation on float-complex elements

Function

```
clAmdBlasStatus
clAmdBlasCsrot(
    size_t N,
    cl_mem X,
    size_t offx,
    int incx,
    cl_mem Y,
    size_t offy,
    int incy,
    cl_float C,
    cl_float S,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description Apply plane rotation on float-complex elements.

Parameters

in	*N*	Number of elements of vectors *X*. and *Y*.
in/ out	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of vector *X*. Must not be zero.
in/ out	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset of first element of vector *Y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of vector *Y*. Must not be zero.
in	*C*	Specifies the cosine, cos.
in	*S*	Specifies the sine, sin.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects for each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- otherwise, the same error code as the `clAmdBlasSrot()` function.

2.9.4 Zdrot

Apply plane rotation on double-complex elements

Function

```
clAmdBlasStatus
clAmdBlasZdrot(
    size_t N,
    cl_mem X,
    size_t offx,
    int incx,
    cl_mem Y,
    size_t offy,
    int incy,
    cl_double C,
    cl_double S,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description Apply plane rotation on double-complex elements.

Parameters

in	*N*	Number of elements of vectors *X*. and *Y*.
in/ out	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of vector *X*. Must not be zero.
in/ out	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset of first element of vector *Y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of vector *Y*. Must not be zero.
in	*C*	Specifies the cosine. This number is real.
in	*S*	Specifies the sine. This number is real.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects for each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support the floating point arithmetic with double precision;
- otherwise, the same error code as the `clAmdBlasSrot()` function.

2.10 xROTM - Apply Modified Givens ROTation for Points in the Plane

2.10.1 Srotm

Apply modified Givens rotation to float elements

Function

```
clAmdBlasStatus
clAmdBlasSrotm(
    size_t N,
    cl_mem X,
    size_t offx,
    int incx,
    cl_mem Y,
    size_t offy,
    int incy,
    const cl_mem SPARAM,
    size_t offSparam,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description Apply modified Givens rotation to float elements.

Parameters

in	*N*	Number of elements of vectors *X*. and *Y*.
in/ out	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of elements for first element in vector *X*.
in	*incx*	Increment for the elements of X. Must not be zero.
in/ out	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset in number of elements for first element in vector *Y*.
in	*incy*	Increment for the elements of Y Must not be zero.
in/ out	*SPARAM*	Buffer object that contains SPARAM array of minimum length 5: SPARAM(1)=SFLAG SPARAM(2)=SH11 SPARAM(3)=SH21 SPARAM(4)=SH12 SPARAM(5)=SH22
in	*offSparam*	Offset of first element of array *SPARAM* in buffer object. Counted in elements.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects for each command queue that identify a particular kernel execution instance.

Apply modified Givens rotation to float elements (Cont.)

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasNotInitialized` if `clAmdBlasSetup()` was not called.
- `clAmdBlasInvalidValue` if invalid parameters are passed:
  - either *N* is zero, or
  - or *incx* or *incy* is zero, or
  - the vector sizes along with the increments lead to accessing outside of any of the buffers.
- `clAmdBlasInvalidMemObject` if either *A* or *X* object is Invalid, or an image object rather than the buffer one;
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures;
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid;
- `clAmdBlasInvalidContext` if a context to which a passed command queue belongs was released;
- `clAmdBlasInvalidOperation` if the kernel compilation relating to a previous call has not completed for any of the target devices;
- `clAmdBlasCompilerNotAvailable` if a compiler is not available;
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

Examples `example_srotm.c`

2.10.2 Drotm

Apply modified Givens rotation to double elements

Function

```
clAmdBlasStatus
clAmdBlasDrotm(
    size_t N,
    cl_mem X,
    size_t offx,
    int incx,
    cl_mem Y,
    size_t offy,
    int incy,
    const cl_mem DPARAM,
    size_t offDparam,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description Apply modified Givens rotation to double elements.

Parameters

in	*N*	Number of elements in vectors *X* and *Y*.
in/ out	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of elements for first element in vector *X*. Counted in elements.
in	*incx*	Increment for the elements of X. Must not be zero.
in/ out	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset in number of elements for first element in vector *Y*. Counted in elements.
in	*incy*	Increment for the elements of Y Must not be zero.

in/ out	*DPARAM*	Buffer object that contains DPARAM array of minimum length 5:
		`DPARAM(1)=DFLAG` `DPARAM(4)=DH12`
		`DPARAM(2)=DH11` `DPARAM(5)=DH22`
		`DPARAM(3)=DH21`

in	*offDparam*	Offset of first element of array *DPARAM* in buffer object. Counted in elements.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects for each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support the floating point arithmetic with double precision;
- otherwise, the same error code as the `clAmdBlasSrotm()` function.

2.11 NRM2 - Euclidean Norm of a Vector

2.11.1 Snrm2

Compute the Euclidean norm of a vector containing float elements

Function

```
clAmdBlasStatus
clAmdBlasSnrm2(
    size_t N,
    cl_mem NRM2,
    size_t offNRM2,
    const cl_mem X,
    size_t offx,
    int incx,
    cl_mem scratchBuff,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description

```
NRM2 = sqrt( X' * X )
```

Parameters

in	*N*	Number of elements in vector *X*.
out	*NRM2*	Buffer object that contains the NRM2 value.
in	*offNRM2*	Offset to NRM2 value in *NRM2* buffer object. Counted in elements.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of X. Must not be zero.
in/ out	*scratchBuff*	Temporary cl_mem scratch buffer object that can hold at least (2*N) elements.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects for each command queue that identify a particular kernel execution instance.

Compute the Euclidean norm of a vector containing float elements (Cont.)

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasNotInitialized` if `clAmdBlasSetup()` was not called.
- `clAmdBlasInvalidValue` if invalid parameters are passed:
  - either *N* is zero, or
  - or *incx* is zero, or
  - the vector sizes along with the increments lead to accessing outside of any of the buffers.
- `clAmdBlasInvalidMemObject` if either *A* or *X* object is Invalid, or an image object rather than the buffer one;
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures;
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid;
- `clAmdBlasInvalidContext` if a context to which a passed command queue belongs was released;
- `clAmdBlasInvalidOperation` if the kernel compilation relating to a previous call has not completed for any of the target devices;
- `clAmdBlasCompilerNotAvailable` if a compiler is not available;
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

Examples `example_snrm2.c`

2.11.2 Dnrm2

Compute the Euclidean norm of a vector containing double elements

Function

```
clAmdBlasStatus
clAmdBlasDnrm2(
    size_t N,
    cl_mem NRM2,
    size_t offNRM2,
    const cl_mem X,
    size_t offx,
    int incx,
    cl_mem scratchBuff,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description

```
NRM2 = sqrt( X' * X )
```

Parameters

in	*N*	Number of elements in vector *X*.
out	*NRM2*	Buffer object that contains the NRM2 value.
in	*offNRM2*	Offset to NRM2 value in *NRM2* buffer object. Counted in elements.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of X. Must not be zero.
in/ out	*scratchBuff*	Temporary cl_mem scratch buffer object that can hold at least (2*N) elements.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects for each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support the floating point arithmetic with double precision;
- otherwise, the same error code as the `clAmdBlasSnrm2()` function.

2.11.3 Scnrm2

Compute the Euclidean norm of a vector containing float-complex elements

Function

```
clAmdBlasStatus
clAmdBlasScnrm2(
    size_t N,
    cl_mem NRM2,
    size_t offNRM2,
    const cl_mem X,
    size_t offx,
    int incx,
    cl_mem scratchBuff,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description

```
NRM2 = sqrt( X**H * X )
```

Parameters

in	*N*	Number of elements in vector *X*.
out	*NRM2*	Buffer object that contains the NRM2 value. Note that the result of Scnrm2 is a real number.
in	*offNRM2*	Offset to NRM2 value in *NRM2* buffer object. Counted in elements.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of X. Must not be zero.
in/ out	*scratchBuff*	Temporary cl_mem scratch buffer object that can hold at least (2*N) elements.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects for each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- otherwise, the same error code as the `clAmdBlasSnrm2()` function.

2.11.4 Dznrm2

Compute the Euclidean norm of a vector containing double-complex elements

Function

```
clAmdBlasStatus
clAmdBlasDznrm2(
    size_t N,
    cl_mem NRM2,
    size_t offNRM2,
    const cl_mem X,
    size_t offx,
    int incx,
    cl_mem scratchBuff,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description

```
NRM2 = sqrt( X**H * X )
```

Parameters

in	*N*	Number of elements in vector *X*.
out	*NRM2*	Buffer object that contains the NRM2 value. Note that the result of Dznrm2 is a real number.
in	*offNRM2*	Offset to NRM2 value in *NRM2* buffer object. Counted in elements.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of X. Must not be zero.
in/ out	*scratchBuff*	Temporary cl_mem scratch buffer object that can hold at least (2*N) elements.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects for each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support the floating point arithmetic with double precision;
- otherwise, the same error code as the `clAmdBlasSnrm2()` function.

2.12 ixAMAX - Index of MAX Absolute Value

2.12.1 iSamax

Index of max absolute value in a float array

Function

```
clAmdBlasStatus
clAmdBlasiSamax(
    size_t N,
    cl_mem iMax,
    size_t offiMax,
    const cl_mem X,
    size_t offx,
    int incx,
    cl_mem scratchBuff,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description Index of max absolute value in a float array.

Parameters

in	*N*	Number of elements in vector *X*.
out	*iMax*	Buffer object storing the index of first absolute max. The index is of type unsigned int.
in	*offiMax*	Offset for storing index in the buffer iMax. Counted in elements.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of the first element of vector *X* in the buffer object. Counted in elements.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in/ out	*scratchBuff*	Temporary cl_mem object to store intermediate results. Must hold at least (2*N) elements.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects for each command queue that identify a particular kernel execution instance.

Index of max absolute value in a float array (Cont.)

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasNotInitialized` if `clAmdBlasSetup()` was not called;
- `clAmdBlasInvalidValue` if invalid parameters are passed:
  - *N* is zero, or
  - *incx* is zero, or
  - the vector sizes along with the increments lead to accessing outside of any of the buffers.
- `clAmdBlasInvalidMemObject` if any of *iMax, X*, or *scratchBuff* object is invalid, or an image object rather than the buffer one;
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures;
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid;
- `clAmdBlasInvalidContext` **if a context to which a passed command queue belongs was** released;
- `clAmdBlasInvalidOperation` if the kernel compilation relating to a previous call has not completed for any of the target devices;
- `clAmdBlasCompilerNotAvailable` if a compiler is not available;
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

Examples `example_isamax.c`

2.12.2 iDamax

Index of max absolute value in a double array

Function

```
clAmdBlasStatus
clAmdBlasiDamax(
    size_t N,
    cl_mem iMax,
    size_t offiMax,
    const cl_mem X,
    size_t offx,
    int incx,
    cl_mem scratchBuff,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description Index of max absolute value in a double array.

Parameters

in	*N*	Number of elements in vector *X*.
out	*iMax*	Buffer object storing the index of first absolute max. The index is of type unsigned int.
in	*offiMax*	Offset for storing index in the buffer iMax. Counted in elements.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of the first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in/ out	*scratchBuff*	Temporary cl_mem object to store intermediate results. Must hold at least (2*N) elements.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects for each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support the floating point arithmetic with double precision;
- otherwise, the same error codes as the `clAmdBlasiSamax()` function.

2.12.3 iCamax

Index of max absolute value in a complex-float array

Function

```
clAmdBlasStatus
clAmdBlasiCamax(
    size_t N,
    cl_mem iMax,
    size_t offiMax,
    const cl_mem X,
    size_t offx,
    int incx,
    cl_mem scratchBuff,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description Index of max absolute value in a complex-float array.

Parameters

in	*N*	Number of elements in vector *X*.
out	*iMax*	Buffer object storing the index of first absolute max. The index is of type unsigned int.
in	*offiMax*	Offset for storing index in the buffer iMax. Counted in elements.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of the first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in/ out	*scratchBuff*	Temporary cl_mem object to store intermediate results. Must hold at least (2*N) elements.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects for each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- otherwise, the same error codes as the `clAmdBlasiSamax()` function.

2.12.4 iZamax

Index of max absolute value in a complex-double array

Function

```
clAmdBlasStatus
clAmdBlasiZamax(
    size_t N,
    cl_mem iMax,
    size_t offiMax,
    const cl_mem X,
    size_t offx,
    int incx,
    cl_mem scratchBuff,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description Index of max absolute value in a complex-double array.

Parameters

in	*N*	Number of elements in vector *X*.
out	*iMax*	Buffer object storing the index of first absolute max. The index is of type unsigned int.
in	*offiMax*	Offset for storing index in the buffer iMax. Counted in elements.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of the first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in/ out	*scratchBuff*	Temporary cl_mem object to store intermediate results. Must hold at least (2*N) elements.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects for each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support the floating point arithmetic with double precision;
- otherwise, the same error codes as the `clAmdBlasiSamax()` function.

2.13 ASUM - SUM of Absolute Values

2.13.1 Sasum

Absolute sum of values of a vector containing float elements

Function

```
clAmdBlasStatus
clAmdBlasSasum(
    size_t N,
    cl_mem asum,
    size_t offAsum,
    const cl_mem X,
    size_t offx,
    int incx,
    cl_mem scratchBuff,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description Absolute sum of values of a vector containing float elements.

Parameters

in	*N*	Number of elements in vector *X*.
out	*asum*	Buffer object storing the absolute sum value.
in	*offAsum*	Offset to absolute sum in the buffer *asum* buffer object. Counted in elements.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of the first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in/ out	*scratchBuff*	Temporary cl_mem scratch buffer object of minimum size N.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects for each command queue that identify a particular kernel execution instance.

Absolute sum of values of a vector containing float elements (Cont.)

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasNotInitialized` if `clAmdBlasSetup()` was not called;
- `clAmdBlasInvalidValue` if invalid parameters are passed:
  - *N* is zero or
    either *incx* is zero, or
  - the vector sizes and the increments lead to accessing outside the buffers;
- `clAmdBlasInvalidMemObject` if *X*, or *asum* or *scratchbuff* is invalid, or an image object rather than the buffer one;
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures;
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid;
- `clAmdBlasInvalidContext` **if a context to which a passed command queue belongs was** released;
- `clAmdBlasInvalidOperation` if the kernel compilation relating to a previous call has not completed for any of the target devices;
- `clAmdBlasCompilerNotAvailable` if a compiler is not available;
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

Examples `example_sasum.c`

2.13.2 Dasum

Absolute sum of values of a vector containing double elements

Function

```
clAmdBlasStatus
clAmdBlasDasum(
    size_t N,
    cl_mem asum,
    size_t offAsum,
    const cl_mem X,
    size_t offx,
    int incx,
    cl_mem scratchBuff,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description Absolute sum of values of a vector containing double elements.

Parameters

in	*N*	Number of elements in vector *X*.
out	*asum*	Buffer object storing the absolute sum value.
in	*offAsum*	Offset to absolute sum in the buffer *asum* buffer object. Counted in elements.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of the first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in/ out	*scratchBuff*	Temporary cl_mem scratch buffer object of minimum size N.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects for each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support the floating point arithmetic with double precision;
- otherwise, the same error codes as the `clAmdBlasSasum()` function.

2.13.3 Scasum

Absolute sum of values of a vector containing float-complex elements

Function

```
clAmdBlasStatus
clAmdBlasScasum(
    size_t N,
    cl_mem asum,
    size_t offAsum,
    const cl_mem X,
    size_t offx,
    int incx,
    cl_mem scratchBuff,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description Absolute sum of values of a vector containing float-complex elements.

Parameters

in	*N*	Number of elements in vector *X*.
out	*asum*	Buffer object storing the absolute sum value.
in	*offAsum*	Offset to absolute sum in the buffer *asum* buffer object. Counted in elements.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of the first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in/ out	*scratchBuff*	Temporary cl_mem scratch buffer object of minimum size N.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects for each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- otherwise, the same error codes as the `clAmdBlasSasum()` function.

2.13.4 Dzasum

Absolute sum of values of a vector containing double-complex elements

Function

```
clAmdBlasStatus
clAmdBlasDzasum(
    size_t N,
    cl_mem asum,
    size_t offAsum,
    const cl_mem X,
    size_t offx,
    int incx,
    cl_mem scratchBuff,
    cl_uint numCommandQueues,
    cl_command_queue *commandQueues,
    cl_uint numEventsInWaitList,
    const cl_event *eventWaitList,
    cl_event *events);
```

Description Absolute sum of values of a vector containing double-complex elements.

Parameters

in	*N*	Number of elements in vector *X*.
out	*asum*	Buffer object storing the absolute sum value.
in	*offAsum*	Offset to the absolute sum in *asum* buffer object. Counted in elements.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in/ out	*scratchBuff*	Temporary cl_mem scratch buffer object of minimum size N.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects for each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support the floating point arithmetic with double precision;
- otherwise, the same error codes as the `clAmdBlasSasum()` function.

Chapter 3
BLAS-2 Functions

This chapter describes the Level 2 Basic Linear Algebra functions.

3.1 xGEMV - GEneral Matrix-Vector Multiplication

3.1.1 Sgemv

Matrix-vector product with a general rectangular matrix and float elements

Function

`clAmdBlasStatus clAmdBlasSgemv ( clAmdBlasOrder` *`order`*`, clAmdBlasTranspose` *`transA`*`, size_t` *`M`*`, size_t` *`N`*`, cl_float` *`alpha`*`, const cl_mem` *`A`* `size_t` *`lda`*`, const cl_mem` *`x`*`, size_t` *`offx`*`, int` *`incx`*`, cl_float` *`beta`*`, cl_mem` *`y`*`, size_t` *`offy`*`, int` *`incy`*`, cl_uint` *`numCommandQueues`*`, cl_command_queue *` *`commandQueues`*`, cl_uint` *`numEventsInWaitList`*`, const cl_event *` *`eventWaitList`*`, cl_event *` *`events`* `)`

Description

Matrix-vector product with a general rectangular matrix and float elements.
Matrix-vector products:

- $y \leftarrow \alpha Ax + \beta y$
- $y \leftarrow \alpha A^T x + \beta y$

Parameters

in	*order*	Row/column order.
in	*transA*	How matrix *A* is to be transposed.
in	*M*	Number of rows in matrix *A*.
in	*N*	Number of columns in matrix *A*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *N* when the `order` parameter is set to `clAmdBlasRowMajor`, or less than *M* when the parameter is set to `clAmdBlasColumnMajor`.
in	*x*	Buffer object storing vector *x*.
in	*offx*	Offset of first element of vector *x* in buffer object.
in	*incx*	Increment for the elements of *x*. Must not be zero.
in	*beta*	The factor of the vector *y*.
in/ out	*y*	Buffer object storing the vector *y*.
in	*offy*	Offset of first element of vector *y* in buffer object.
in	*incy*	Increment for the elements of *y*. Must not be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects in each command queue that identify a particular kernel execution instance.

Matrix-vector product with a general rectangular matrix and float elements (Cont.)

Returns

- `clAmdBlasSuccess` on success.
- `clAmdBlasNotInitialized` if `clAmdBlasSetup()` was not called.
- `clAmdBlasInvalidValue` if invalid parameters are passed:
  - either *M* or *N* is zero, or
  - either *incx* or *incy* is zero, or
  - the leading dimension is invalid.
- `clAmdBlasInvalidMemObject` if either *A*, *x*, or *y* object is Invalid, or an image object rather than the buffer one.
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures.
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid.
- `clAmdBlasInvalidContext` if a context a passed command queue belongs to was released.
- `clAmdBlasInvalidOperation` if kernel compilation relating to a previous call has not completed for any of the target devices.
- `clAmdBlasCompilerNotAvailable` if a compiler is not available.
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

Examples `example_sgemv.c`.

3.1.2 Dgemv

Matrix-vector product with a general rectangular matrix and double elements

Function

```
clAmdBlasStatus clAmdBlasDgemv ( clAmdBlasOrder order, clAmdBlasTranspose
transA, size_t M, size_t N, cl_double alpha, const cl_mem A, size_t lda, const
cl_mem x, size_t offx, int incx, cl_double beta, cl_mem y, size_t offy, int
incy, cl_uint numCommandQueues, cl_command_queue * commandQueues, cl_uint
numEventsInWaitList, const cl_event * eventWaitList, cl_event * events )
```

Description

This function has been deprecated.
Matrix-vector product with a general rectangular matrix and double elements.
Matrix-vector products:

- $y \leftarrow \alpha Ax + \beta y$
- $y \leftarrow \alpha A^T x + \beta y$

Parameters

in	*order*	Row/column order.
in	*transA*	How matrix *A* is to be transposed.
in	*M*	Number of rows in matrix *A*.
in	*N*	Number of columns in matrix *A*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*lda*	Leading dimension of matrix *A*. See `clAmdBlasSgemv()`.
in	*x*	Buffer object storing vector *x*.
in	*offx*	Offset of first element of vector *x* in buffer object.
in	*incx*	Increment for the elements of *x*. It cannot be zero.
in	*beta*	The factor of the vector *y*.
in/ out	*y*	Buffer object storing the vector *y*.
in	*offy*	Offset of first element of vector *y* in buffer object.
in	*incy*	Increment for the elements of *y*. It cannot be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which the task is done.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success.
- `clAmdBlasInvalidDevice` if a target device does not support the floating point arithmetic with double precision.
- For other returns, the same error codes as the `clAmdBlasSgemv()` function.

3.1.3 Cgemv

Matrix-vector product with a general rectangular matrix and float-complex elements

Function

```
clAmdBlasStatus clAmdBlasCgemv ( clAmdBlasOrder order, clAmdBlasTranspose
transA, size_t M, size_t N, FloatComplex alpha, const cl_mem A, size_t lda,
const cl_mem x, size_t offx, int incx, FloatComplex beta, cl_mem y,
size_toffy, int incy, cl_uint numCommandQueues, cl_command_queue *
commandQueues, cl_uint numEventsInWaitList, const cl_event *
eventWaitList, cl_event * events )
```

Description

Matrix-vector product with a general rectangular matrix and float-complex elements.
Matrix-vector products:

- $y \leftarrow \alpha Ax + \beta y$
- $y \leftarrow \alpha A^T x + \beta y$

Parameters

in	*order*	Row/column order.
in	*transA*	How matrix *A* is to be transposed.
in	*M*	Number of rows in matrix *A*.
in	*N*	Number of columns in matrix *A*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*Ida*	Leading dimension of matrix *A*. For a detailed description, see `clAmdBlasSgemv()`.
in	*x*	Buffer object storing vector *x*.
in	*offx*	Offset of first element of vector *x* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of *x*. It cannot be zero.
in	*beta*	The factor of the vector *y*.
in/ out	*y*	Buffer object storing the vector *y*.
in	*offy*	Offset of first element of vector *y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of *y*. It cannot be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects in each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- otherwise, the same error codes as the `clAmdBlasSgemv()` function.

3.1.4 Zgemv

Matrix-vector product with a general rectangular matrix and double-complex elements

Function

```
clAmdBlasStatus clAmdBlasZgemv ( clAmdBlasOrder order, clAmdBlasTranspose
transA, size_t M, size_t N, DoubleComplex alpha, const cl_mem A, size_t lda,
const cl_mem x, size_t offx, int incx, DoubleComplex beta, cl_mem y, size_t
offy, int incy, cl_uint numCommandQueues, cl_command_queue * commandQueues,
cl_uint numEventsInWaitList, const cl_event * eventWaitList, cl_event *
events )
```

Description

Matrix-vector product with a general rectangular matrix and double-complex elements. Matrix-vector products:

- $y \leftarrow \alpha Ax + \beta y$
- $y \leftarrow \alpha A^T x + \beta y$

Parameters

in	*order*	Row/column order.
in	*transA*	How matrix *A* is to be transposed.
in	*M*	Number of rows in matrix *A*.
in	*N*	Number of columns in matrix *A*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*lda*	Leading dimension of matrix *A*. For a detailed description, see `clAmdBlasSgemv()`.
in	*x*	Buffer object storing vector *x*.
in	*offx*	Offset of first element of vector *x* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of *x*. It cannot be zero.
in	*beta*	The factor of the vector *y*.
in/ out	*y*	Buffer object storing the vector *y*.
in	*offy*	Offset of first element of vector *y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of *y*. It cannot be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects in each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support the floating point arithmetic with double precision;
- otherwise, the same error codes as the `clAmdBlasSgemv()` function.

3.2 xGEMVEX - GEneral Matrix-Vector Multiplication, Extended Version

3.2.1 SgemvEx

Matrix-vector product with a general rectangular matrix and float elements

Function

`clAmdBlasStatus clAmdBlasSgemvEx ( clAmdBlasOrder` *`order`*`, clAmdBlasTranspose` *`transA`*`, size_t` *`M`*`, size_t` *`N`*`, cl_float` *`alpha`*`, const cl_mem` *`A`*`, size_t` *`offA`*`, size_t` *`lda`*`, const cl_mem` *`x`*`, size_t` *`offx`*`, int` *`incx`*`, cl_float` *`beta`*`, cl_mem` *`y`*`, size_t` *`offy`*`, int` *`incy`*`, cl_uint` *`numCommandQueues`*`, cl_command_queue * ` *`commandQueues`*`, cl_uint` *`numEventsInWaitList`*`, const cl_event * ` *`eventWaitList`*`, cl_event * ` *`events`* `)`

Description

Matrix-vector product with a general rectangular matrix and float elements. Extended version, which takes an offset value for all matrix arguments.
Matrix-vector products:

- $y \leftarrow \alpha Ax + \beta y$
- $y \leftarrow \alpha A^T x + \beta y$

Parameters

in	*order*	Row/column order.
in	*transA*	How matrix *A* is to be transposed.
in	*M*	Number of rows in matrix *A*.
in	*N*	Number of columns in matrix *A*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offA*	Offset of the first element of the matrix *A* in the buffer object. Counted in elements.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *N* when the order parameter is set to `clAmdBlasRowMajor`, or less than *M* when the parameter is set to `clAmdBlasColumnMajor`.
in	*x*	Buffer object storing vector *x*.
in	*offx*	Offset of first element of vector *x* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of *x*. It cannot be zero.
in	*beta*	The factor of the vector *y*.
in/ out	*y*	Buffer object storing the vector *y*.
in	*offy*	Offset of first element of vector *y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of *y*. It cannot be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects in each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidValue` if *offA* exceeds the size of *A* buffer object;
- otherwise, the same error codes as the `clAmdBlasSgemv()` function.

3.2.2 DgemvEx

Matrix-vector product with a general rectangular matrix and double elements

Function

```
clAmdBlasStatus clAmdBlasDgemvEx ( clAmdBlasOrder order, clAmdBlasTranspose
transA, size_t M, size_t N, cl_double alpha, const cl_mem A, size_t offA, size_t
lda, const cl_mem x, size_t offx, int incx, cl_double beta, cl_mem y, size_t
offy, int incy, cl_uint numCommandQueues, cl_command_queue * commandQueues,
cl_uint numEventsInWaitList, const cl_event * eventWaitList, cl_event *
events )
```

Description

Matrix-vector product with a general rectangular matrix and double elements. Extended version, which takes an offset value for all matrix arguments.
Matrix-vector products:

- $y \leftarrow \alpha Ax + \beta y$
- $y \leftarrow \alpha A^T x + \beta y$

Parameters

in	*order*	Row/column order.
in	*transA*	How matrix *A* is to be transposed.
in	*M*	Number of rows in matrix *A*.
in	*N*	Number of columns in matrix *A*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offA*	Offset of the first element of *A* in the buffer object. Counted in elements.
in	*lda*	Leading dimension of matrix *A*. For a detailed description, see `clAmdBlasSgemv()`.
in	*x*	Buffer object storing vector *x*.
in	*offx*	Offset of first element of vector *x* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of *x*. It cannot be zero.
in	*beta*	The factor of the vector *y*.
in/ out	*y*	Buffer object storing the vector *y*.
in	*offy*	Offset of first element of vector *y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of *y*. It cannot be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support the floating point arithmetic with double precision;
- `clAmdBlasInvalidValue` if `offA` exceeds the size of `A` buffer object;
- otherwise, the same error codes as the `clAmdBlasSgemv()` function.

3.2.3 CgemvEx

Matrix-vector product with a general rectangular matrix and float-complex elements

Function

```
clAmdBlasStatus clAmdBlasCgemvEx ( clAmdBlasOrder order, clAmdBlasTranspose
transA, size_t M, size_t N, FloatComplex alpha, const cl_mem A, size_t offA,
size_t lda, const cl_mem x, size_t offx, int incx, FloatComplex beta, cl_mem
y, size_t offy, int incy, cl_uint numCommandQueues, cl_command_queue *
commandQueues, cl_uint numEventsInWaitList, const cl_event * eventWaitList,
cl event * events )
```

Description

Matrix-vector product with a general rectangular matrix and float-complex elements. Extended version, which takes an offset value for all matrix arguments.
Matrix-vector products:

- $y \leftarrow \alpha Ax + \beta y$
- $y \leftarrow \alpha A^T x + \beta y$

Parameters

in	*order*	Row/column order.
in	*transA*	How matrix *A* is to be transposed.
in	*M*	Number of rows in matrix *A*.
in	*N*	Number of columns in matrix *A*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offA*	Offset of the first element of the matrix *A* in the buffer object. Counted in elements.
in	*lda*	Leading dimension of matrix *A*. For a detailed description, see `clAmdBlasSgemv()`.
in	*x*	Buffer object storing vector *x*.
in	*offx*	Offset of first element of vector *x* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of *x*. It cannot be zero.
in	*beta*	The factor of the vector *y*.
in/ out	*y*	Buffer object storing the vector *y*.
in	*offy*	Offset of first element of vector *y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of *y*. It cannot be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects in each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidValue` if *offA* exceeds the size of *A* buffer object;
- otherwise, the same error codes as the `clAmdBlasSgemv()` function.

3.2.4 ZgemvEx

Matrix-vector product with a general rectangular matrix and double-complex elements

Function

```
clAmdBlasStatus clAmdBlasZgemvEx ( clAmdBlasOrder order, clAmdBlasTranspose
transA, size_t M, size_t N, DoubleComplex alpha, const cl_mem A, size_t offA,
size_t lda, const cl_mem x, size_t offx, int incx, DoubleComplex beta, cl_mem
y, size_t offy, int incy, cl_uint numCommandQueues, cl_command_queue *
commandQueues, cl_uint numEventsInWaitList, const cl_event *
eventWaitList, cl_event * events )
```

Description

Matrix-vector product with a general rectangular matrix and double-complex elements. Extended version, which takes an offset value for all matrix arguments.
Matrix-vector products:

- $y \leftarrow \alpha Ax + \beta y$
- $y \leftarrow \alpha A^T x + \beta y$

Parameters

in	*order*	Row/column order.
in	*transA*	How matrix *A* is to be transposed.
in	*M*	Number of rows in matrix *A*.
in	*N*	Number of columns in matrix *A*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offA*	Offset of the first element of the matrix *A* in the buffer object. Counted in elements.
in	*lda*	Leading dimension of matrix *A*. For a detailed description, see `clAmdBlasSgemv()`.
in	*x*	Buffer object storing vector *x*.
in	*offx*	Offset of first element of vector *x* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of *x*. It cannot be zero.
in	*beta*	The factor of the vector *y*.
in/ out	*y*	Buffer object storing the vector *y*.
in	*offy*	Offset of first element of vector *y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of *y*. It cannot be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects in each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support the floating point arithmetic with double precision;
- `clAmdBlasInvalidValue` if *offA* exceeds the size of *A* buffer object;
- otherwise, the same error codes as the `clAmdBlasSgemv()` function.

3.3 xSYMV - SYmmetric Matrix-Vector Multiplication

3.3.1 Ssymv

Matrix-vector product with a symmetric matrix and float elements

Function

```
clAmdBlasStatus clAmdBlasSsymv ( clAmdBlasOrder order, clAmdBlasUplo uplo,
size_t N, cl_float alpha, const cl_mem A, size_t lda, const cl_mem x, size_t
offx, int incx, cl_float beta, cl_mem y, size_t offy, int incy, cl_uint
numCommandQueues, cl_command_queue * commandQueues, cl_uint numEventsInWaitList,
const cl_event * eventWaitList, cl_event * events )
```

Description

Matrix-vector product with a symmetric matrix and float elements. Matrix-vector products:

- $y \leftarrow \alpha Ax + \beta y$

Parameters

in	*order*	Row/columns order.
in	*uplo*	The triangle in matrix being referenced.
in	*N*	Number of rows and columns in matrix *A*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*Ida*	Leading dimension of matrix *A*. It cannot less than *N*.
in	*x*	Buffer object storing vector *x*.
in	*offx*	Offset of first element of vector *x* in buffer object.
in	*incx*	Increment for the elements of vector *x*. It cannot be zero.
in	*beta*	The factor of vector *y*.
in/ out	*y*	Buffer object storing vector *y*.
in	*offy*	Offset of first element of vector *y* in buffer object.
in	*incy*	Increment for the elements of vector *y*. It cannot be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects in each command queue that identify a particular kernel execution instance.

Matrix-vector product with a symmetric matrix and float elements (Cont.)

Returns

- `clAmdBlasSuccess` on success.
- `clAmdBlasNotInitialized` if `clAmdBlasSetup()` was not called.
- `clAmdBlasInvalidValue` if invalid parameters are passed:
  - *N* is zero, or
  - either *incx* or *incy* is zero, or
  - the leading dimension is invalid.
- `clAmdBlasInvalidMemObject` if either *A*, *x*, or *y* object is invalid, or an image object rather than the buffer one.
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures.
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid.
- `clAmdBlasInvalidContext` if a context a passed command queue belongs to was released.
- `clAmdBlasInvalidOperation` if kernel compilation relating to a previous call has not completed for any of the target devices.
- `clAmdBlasCompilerNotAvailable` if a compiler is not available.
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

Examples `example_ssymv.c`.

3.3.2 Dsymv

Matrix-vector product with a symmetric matrix and double elements

Function

```
clAmdBlasStatus clAmdBlasDsymv ( clAmdBlasOrder order, clAmdBlasUplo uplo,
size_t N, cl_double alpha, const cl_mem A, size_t lda, const cl_mem x, size_t
offx, int incx, cl_double beta, cl_mem y, size_t offy, int incy, cl_uint
numCommandQueues, cl_command_queue * commandQueues, cl_uint numEventsInWaitList,
const cl_event * eventWaitList, cl_event * events )
```

Description

This function has been deprecated.
Matrix-vector product with a symmetric matrix and double elements.
Matrix-vector products:

- $y \leftarrow \alpha Ax + \beta y$

Parameters

in	*order*	Row/columns order.
in	*uplo*	The triangle in matrix being referenced.
in	*N*	Number of rows and columns in matrix *A*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*lda*	Leading dimension of matrix *A*. It cannot less than *N*.
in	*x*	Buffer object storing vector *x*.
in	*offx*	Offset of first element of vector *x* in buffer object.
in	*incx*	Increment for the elements of vector *x*. It cannot be zero.
in	*beta*	The factor of vector *y*.
in/ out	*y*	Buffer object storing vector *y*.
in	*offy*	Offset of first element of vector *y* in buffer object.
in	*incy*	Increment for the elements of vector *y*. It cannot be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success.
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision.
- For other returns, the same error codes as the `clAmdBlasSsymv()` function.

3.4 xSYMVEX - SYmmetric Matrix-Vector Multiplication, Extended Version

3.4.1 SsymvEx

Matrix-vector product with a symmetric matrix and float elements

Function

```
clAmdBlasStatus clAmdBlasSsymvEx ( clAmdBlasOrder order, clAmdBlasUplo uplo,
size_t N, cl_float alpha, const cl_mem A, size_t offA, size_t lda, const
cl_mem x, size_t offx, int incx, cl_float beta, cl_mem y, size_t offy, int
incy, cl_uint numCommandQueues, cl_command_queue * commandQueues, cl_uint
numEventsInWaitList, const cl_event * eventWaitList, cl_event * events )
```

Description

Matrix-vector product with a symmetric matrix and float elements. Extended version, which takes an offset value for all matrix arguments.
Matrix-vector products:

- $y \leftarrow \alpha Ax + \beta y$

Parameters

in	*order*	Row/columns order.
in	*uplo*	The triangle in matrix being referenced.
in	*N*	Number of rows and columns in matrix *A*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offA*	Offset of the first element of the matrix *A* in the buffer object. Counted in elements.
in	*lda*	Leading dimension of matrix *A*. It cannot less than *N*.
in	*x*	Buffer object storing vector *x*.
in	*offx*	Offset of first element of vector *x* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of vector *x*. It cannot be zero.
in	*beta*	The factor of vector *y*.
in/ out	*y*	Buffer object storing vector *y*.
in	*offy*	Offset of first element of vector *y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of vector *y*. It cannot be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects in each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidValue` if *offA* exceeds the size of *A* buffer object;
- otherwise, the same error codes as the `clAmdBlasSgemv()` function.

3.4.2 DsymvEx

Matrix-vector product with a symmetric matrix and double elements

Function

```
clAmdBlasStatus clAmdBlasDsymvEx ( clAmdBlasOrder order, clAmdBlasUplo uplo,
size_t N, cl_double alpha, const cl_mem A, size_t offA, size_t lda, const
cl_mem x, size_t offx, int incx, cl_double beta, cl_mem y, size_t offy, int
incy, cl_uint numCommandQueues, cl_command_queue * commandQueues, cl_uint
numEventsInWaitList, const cl_event * eventWaitList, cl_event * events )
```

Description

Matrix-vector product with a symmetric matrix and double elements. Extended version, which takes an offset value for all matrix arguments.
Matrix-vector products:

- $y \leftarrow \alpha Ax + \beta y$

Parameters

in	*order*	Row/columns order.
in	*uplo*	The triangle in matrix being referenced.
in	*N*	Number of rows and columns in matrix *A*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offA*	Offset of the first element of the matrix *A* in the buffer object. Counted in elements.
in	*lda*	Leading dimension of matrix *A*. It cannot less than *N*.
in	*x*	Buffer object storing vector *x*.
in	*offx*	Offset of first element of vector *x* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of vector *x*. It cannot be zero.
in	*beta*	The factor of vector *y*.
in/ out	*y*	Buffer object storing vector *y*.
in	*offy*	Offset of first element of vector *y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of vector *y*. It cannot be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- `clAmdBlasInvalidValue` if *offA* exceeds the size of *A* buffer object;
- otherwise, the same error codes as the `clAmdBlasSsymv()` function.

3.5 xHEMV - HErmitian Matrix-Vector Multiplication

3.5.1 Chemv

Matrix-vector product with a hermitian matrix and float-complex elements

Function

```
clAmdBlasStatus clAmdBlasChemv ( clAmdBlasOrder order, clAmdBlasUplo uplo,
size_t N, FloatComplex alpha, const cl_mem A, size_t offa, size_t lda, const
cl_mem X, size_t offx, int incx, FloatComplex beta, cl_mem Y, size_t offy,
int incy, cl_uint numCommandQueues, cl_command_queue * commandQueues,
cl_uint numEventsInWaitList, const cl_event * eventWaitList, cl_event *
events )
```

Description

Matrix-vector product with a hermitian matrix and float-complex elements.
Matrix-vector products:

- $y \leftarrow \alpha Ax + \beta y$

Parameters

in	*order*	Row/columns order.
in	*uplo*	The triangle in matrix being referenced.
in	*N*	Number of rows and columns in matrix *A*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offa*	Offset in number of elements for first element in matrix *A*.
in	*lda*	Leading dimension of matrix *A*. It cannot less than *N*.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of vector *X*. It cannot be zero.
in	*beta*	The factor of vector *Y*.
in/ out	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset of first element of vector *Y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of vector *Y*. It cannot be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Matrix-vector product with a hermitian matrix and float-complex elements (Cont.)

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasNotInitialized` if `clAmdBlasSetup()` was not called;
- `clAmdBlasInvalidValue` if invalid parameters are passed:
  - *N* is zero, or
  - either *incx* or *incy* is zero, or
  - any of the leading dimensions is invalid;
  - the matrix sizes or the vector sizes along with the increments lead to accessing outsize of any of the buffers;
- `clAmdBlasInvalidMemObject` if either *A*, *X*, or *Y* object is invalid, or an image object rather than the buffer one;
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures;
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid;
- `clAmdBlasInvalidContext` if a context a passed command queue belongs to was released;
- `clAmdBlasInvalidOperation` if kernel compilation relating to a previous call has not completed for any of the target devices;
- `clAmdBlasCompilerNotAvailable` if a compiler is not available;
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

3.5.2 Zhemv

Matrix-vector product with a hermitian matrix and double-complex elements

Function `clAmdBlasStatus clAmdBlasZhemv (clAmdBlasOrder order, clAmdBlasUplo uplo, size_t N, DoubleComplex alpha, const cl_mem A, size_t offa, size_t lda, const cl_mem X, size_t offx, int incx, DoubleComplex beta, cl_mem Y, size_t offy, int incy, cl_uint numCommandQueues, cl_command_queue * commandQueues, cl_uint numEventsInWaitList, const cl_event * eventWaitList, cl_event * events)`

Description Matrix-vector product with a hermitian matrix and double-complex elements.
Matrix-vector products:

- $y \leftarrow \alpha Ax + \beta y$

Parameters

in	*order*	Row/columns order.
in	*uplo*	The triangle in matrix being referenced.
in	*N*	Number of rows and columns in matrix *A*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offa*	Offset in number of elements for first element in matrix *A*.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *N*.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of vector *X*. It cannot be zero.
in	*beta*	The factor of vector *Y*.
in/ out	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset of first element of vector *Y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of vector *Y*. It cannot be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- otherwise, the same error codes as the `clAmdBlasChemv()` function.

Examples `example_zhemv.cpp`.

3.6 xTRMV - TRiangular Matrix-Vector Multiplication

3.6.1 Strmv

Matrix-vector product with a triangular matrix and float elements

Function

```
clAmdBlasStatus clAmdBlasStrmv ( clAmdBlasOrder order, clAmdBlasUplo uplo,
clAmdBlasTranspose trans, clAmdBlasDiag diag, size_t N, const cl_mem A, size_t
offa, size_t lda, cl_mem X, size_t offx, int incx, cl_mem scratchBuff,
cl_uint numCommandQueues, cl_command_queue * commandQueues, cl_uint
numEventsInWaitList, const cl_event * eventWaitList, cl_event * events )
```

Description

Matrix-vector product with a triangular matrix and float elements.
Matrix-vector products:

- $x \leftarrow Ax$
- $x \leftarrow A^T x$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*trans*	How matrix *A* is to be transposed.
in	*diag*	Specify whether matrix *A* is unit triangular.
in	*N*	Number of rows/columns in matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offa*	Offset in number of elements for first element in matrix *A*.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *N*.
in/ out	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of elements for first element in vector *X*.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in	*scratchBuff*	Temporary cl_mem scratch buffer object which can hold a minimum of (1 + (N-1)∗abs(incx)) elements
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Matrix-vector product with a triangular matrix and float elements (Cont.)

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasNotInitialized` if `clAmdBlasSetup()` was not called;
- `clAmdBlasInvalidValue` if invalid parameters are passed:
  - either *N* or *incx* is zero, or
  - the leading dimension is invalid;
- `clAmdBlasInvalidMemObject` if either *A* or *X* object is Invalid, or an image object rather than the buffer one;
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures;
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid;
- `clAmdBlasInvalidContext` if a context a passed command queue belongs to was released;
- `clAmdBlasInvalidOperation` if kernel compilation relating to a previous call has not completed for any of the target devices;
- `clAmdBlasCompilerNotAvailable` if a compiler is not available;
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

Examples `example_strmv.c`.

3.6.2 Dtrmv

Matrix-vector product with a triangular matrix and double elements

Function

```
clAmdBlasStatus clAmdBlasDtrmv ( clAmdBlasOrder order, clAmdBlasUplo uplo,
clAmdBlasTranspose trans, clAmdBlasDiag diag, size_t N, const cl_mem A, size_t
offa, size_t lda, cl_mem X, size_t offx, int incx, cl_mem scratchBuff,
cl_uint numCommandQueues, cl_command_queue * commandQueues, cl_uint
numEventsInWaitList, const cl_event * eventWaitList, cl event * events )
```

Description

Matrix-vector product with a triangular matrix and double elements.
Matrix-vector products:

- $x \leftarrow Ax$
- $x \leftarrow A^T x$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*trans*	How matrix *A* is to be transposed.
in	*diag*	Specify whether matrix *A* is unit triangular.
in	*N*	Number of rows/columns in matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offa*	Offset in number of elements for first element in matrix *A*.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *N*.
in/ out	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of element for first element in vector *X*.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in	*scratchBuff*	Temporary cl_mem scratch buffer object which can hold a minimum of (1 + (N-1)∗abs(incx)) elements
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- otherwise, the same error codes as the `clAmdBlasStrmv()` function.

3.6.3 Ctrmv

Matrix-vector product with a triangular matrix and float-complex elements

Function

```
clAmdBlasStatus clAmdBlasCtrmv ( clAmdBlasOrder order, clAmdBlasUplo uplo,
clAmdBlasTranspose trans, clAmdBlasDiag diag, size_t N, const cl_mem A, size_t
offa, size_t lda, cl_mem X, size_t offx, int incx, cl_mem scratchBuff,
cl_uint numCommandQueues, cl_command_queue * commandQueues, cl_uint
numEventsInWaitList, const cl_event * eventWaitList, cl_event * events )
```

Description

Matrix-vector product with a triangular matrix and float-complex elements.
Matrix-vector products:

- $x \leftarrow Ax$
- $x \leftarrow A^T x$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*trans*	How matrix *A* is to be transposed.
in	*diag*	Specify whether matrix *A* is unit triangular.
in	*N*	Number of rows/columns in matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offa*	Offset in number of elements for first element in matrix *A*.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *N*.
in/ out	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of elements for first element in vector *X*.
in	*incx*	Increment for the elements of vector *X*. Must not be zero.
in	*scratchBuff*	Temporary cl_mem scratch buffer object which can hold a minimum of (1 + (N-1)∗abs(incx)) elements
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

The same result as the `clAmdBlasStrmv()` function.

3.6.4 Ztrmv

Matrix-vector product with a triangular matrix and double-complex elements

Function

```
clAmdBlasStatus clAmdBlasZtrmv ( clAmdBlasOrder order, clAmdBlasUplo uplo,
clAmdBlasTranspose trans, clAmdBlasDiag diag, size_t N, const cl_mem A, size_t
offa, size_t lda, cl_mem X, size_t offx, int incx, cl_mem scratchBuff,
cl_uint numCommandQueues, cl_command_queue * commandQueues, cl_uint
numEventsInWaitList, const cl_event * eventWaitList, cl_event * events )
```

Description

Matrix-vector product with a triangular matrix and double-complex elements.
Matrix-vector products:

- $x \leftarrow Ax$
- $x \leftarrow A^T x$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*trans*	How matrix *A* is to be transposed.
in	*diag*	Specify whether matrix *A* is unit triangular.
in	*N*	Number of rows/columns in matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offa*	Offset in number of elements for first element in matrix *A*.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *N*.
in/ out	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of elements for first element in vector *X*.
in	*incx*	Increment for the elements of X. Must not be zero.
in	*scratchBuff*	Temporary cl_mem scratch buffer object which can hold a minimum of (1 + (N-1)∗abs(incx)) elements
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

The same result as the `clAmdBlasDtrmv()` function.

3.7 xTRSV - TRiangular matrix-Vector Solve

3.7.1 Strsv

Solving triangular matrix problems with float elements

Function

```
clAmdBlasStatus clAmdBlasStrsv ( clAmdBlasOrder order, clAmdBlasUplo uplo,
clAmdBlasTranspose trans, clAmdBlasDiag diag, size_t N, const cl_mem A, size_t
offa, size_t lda, cl_mem X, size_t offx, int incx, cl_uint numCommandQueues,
cl_command_queue * commandQueues, cl_uint numEventsInWaitList, const
cl_event * eventWaitList, cl_event * events )
```

Description

Solving triangular matrix problems with float elements.
Matrix-vector products:

- $Ax \leftarrow x$
- $A^T x \leftarrow x$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*trans*	How matrix *A* is to be transposed.
in	*diag*	Specify whether matrix *A* is unit triangular.
in	*N*	Number of rows/columns in matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offa*	Offset in number of elements for first element in matrix *A*.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *N*.
in/ out	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *x* in buffer object.
in	*incx*	Increment for the elements of X. Must not be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Solving triangular matrix problems with float elements (Cont.)

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasNotInitialized` if `clAmdBlasSetup()` was not called.
- `clAmdBlasInvalidValue` if invalid parameters are passed:
  - either *N* or *incx* is zero, or
  - the leading dimension is invalid;
- `clAmdBlasInvalidMemObject` if either *A* or *X* object is Invalid, or an image object rather than the buffer one;
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures;
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid;
- `clAmdBlasInvalidContext` if a context a passed command queue belongs to was released;
- `clAmdBlasInvalidOperation` if kernel compilation relating to a previous call has not completed for any of the target devices;
- `clAmdBlasCompilerNotAvailable` if a compiler is not available;
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

Examples

`example_strsv.c`.

3.7.2 Dtrsv

Solving triangular matrix problems with double elements

Function

```
clAmdBlasStatus clAmdBlasDtrsv ( clAmdBlasOrder order, clAmdBlasUplo uplo,
clAmdBlasTranspose trans, clAmdBlasDiag diag, size_t N, const cl_mem A, size_t
offa, size_t lda, cl_mem X, size_t offx, int incx, cl_uint numCommandQueues,
cl_command_queue * commandQueues, cl_uint numEventsInWaitList, const
cl_event * eventWaitList, cl_event * events )
```

Description

Solving triangular matrix problems with double elements.
Matrix-vector products:

- $Ax \leftarrow x$
- $A^T x \leftarrow x$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*trans*	How matrix *A* is to be transposed.
in	*diag*	Specify whether matrix *A* is unit triangular.
in	*N*	Number of rows/columns in matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offa*	Offset in number of elements for first element in matrix *A*.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *N*.
in/ out	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of elements for first element in vector *X*.
in	*incx*	Increment for the elements of X Must not be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- otherwise, the same error codes as the `clAmdBlasStrsv()` function.

3.7.3 Ctrsv

Solving triangular matrix problems with float-complex elements

Function

```
clAmdBlasStatus clAmdBlasCtrsv ( clAmdBlasOrder order, clAmdBlasUplo uplo,
clAmdBlasTranspose trans, clAmdBlasDiag diag, size_t N, const cl_mem A, size_t
offa, size_t lda, cl_mem X, size_t offx, int incx, cl_uint numCommandQueues,
cl_command_queue * commandQueues, cl_uint numEventsInWaitList, const
cl_event * eventWaitList, cl_event * events )
```

Description

Solving triangular matrix problems with float-complex elements.
Matrix-vector products:

- $Ax \leftarrow x$
- $A^T x \leftarrow x$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*trans*	How matrix *A* is to be transposed.
in	*diag*	Specify whether matrix *A* is unit triangular.
in	*N*	Number of rows/columns in matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offa*	Offset in number of elements for first element in matrix *A*.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *N*.
in/ out	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of elements for first element in vector *X*.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

The same result as the `clAmdBlasStrsv()` function.

3.7.4 Ztrsv

Solving triangular matrix problems with double-complex elements

Function `clAmdBlasStatus clAmdBlasZtrsv (clAmdBlasOrder order, clAmdBlasUplo uplo, clAmdBlasTranspose trans, clAmdBlasDiag diag, size_t N, const cl_mem A, size_t offa, size_t lda, cl_mem X, size_t offx, int incx, cl_uint numCommandQueues, cl_command_queue * commandQueues, cl_uint numEventsInWaitList, const cl_event * eventWaitList, cl_event * events)`

Description Solving triangular matrix problems with double-complex elements.
Matrix-vector products:

- $Ax \leftarrow x$
- $A^T x \leftarrow x$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*trans*	How matrix *A* is to be transposed.
in	*diag*	Specify whether matrix *A* is unit triangular.
in	*N*	Number of rows/columns in matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offa*	Offset in number of elements for first element in matrix *A*.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *N*.
in/ out	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of elements for first element in vector *X*.
in	*incx*	Increment for the elements of X. Must not be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns The same result as the `clAmdBlasDtrsv()` function.

3.8 xGER - GEneral matrix Rank 1 operation

3.8.1 Sger

Vector-vector product with float elements and performs the rank 1 operation A

Function

`clAmdBlasStatus clAmdBlasSger ( clAmdBlasOrder` *`order`*`, size_t` *`M`*`, size_t` *`N`*`, cl_float` *`alpha`*`, const cl_mem` *`X`*`, size_t` *`offx`*`, int` *`incx`*`, const cl_mem` *`Y`*`, size_t` *`offy`*`, int` *`incy`*`, cl_mem` *`A`*`, size_t` *`offa`*`, size_t` *`lda`*`, cl_uint` *`numCommandQueues`*`, cl_command_queue * ` *`commandQueues`*`, cl_uint` *`numEventsInWaitList`*`, const cl_event * ` *`eventWaitList`*`, cl_event * ` *`events`* `)`

Description

Vector-vector product with float elements and performs the rank 1 operation A.
Vector-vector products:

- $A \leftarrow \alpha xy^T + A$

Parameters

in	*order*	Row/column order.
in	*M*	Number of rows in matrix *A*.
in	*N*	Number of columns in matrix *A*.
in	*alpha*	Specifies the scalar alpha.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of elements for the first element vector *X*.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset in number of elements for the first element in vector *Y*.
in	*incy*	Increment for the elements of Y. Must not be zero.
in/ out	*A*	Buffer object storing matrix *A*. On exit, *A* is overwritten by the updated matrix.
in	*offa*	Offset in number of elements for the first element in matrix *A*.
in	*Ida*	Leading dimension of matrix *A*. It cannot be less than *N* when the *order* parameter is set to `clAmdBlasRowMajor`, or less than *M* when the parameter is set to `clAmdBlasColumnMajor`.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Vector-vector product with float elements and performs the rank 1 operation A (Cont.)

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasNotInitialized` if `clAmdBlasSetup()` was not called;
- `clAmdBlasInvalidValue` if invalid parameters are passed:
  - *M*, *N* or
    either *incx* or *incy* is zero, or
  - the leading dimension is invalid;
- `clAmdBlasInvalidMemObject` if *A, X*, or *Y* object is invalid, or an image object rather than the buffer one;
- `clAmdBlasOutOfResources` if you use image-based function implementation and no suitable scratch image available;
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures;
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid;
- `clAmdBlasInvalidContext` if a context a passed command queue belongs to was released;
- `clAmdBlasInvalidOperation` if kernel compilation relating to a previous call has not completed for any of the target devices;
- `clAmdBlasCompilerNotAvailable` if a compiler is not available;
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

Examples

`example_sger.c`.

3.8.2 Dger

Vector-vector product with double elements and performs the rank 1 operation A

Function `clAmdBlasStatus clAmdBlasDger (clAmdBlasOrder order, size_t M, size_t N, cl_double alpha, const cl_mem X, size_t offx, int incx, const cl_mem Y, size_t offy, int incy, cl_mem A, size_t offa, size_t lda, cl_uint numCommandQueues, cl_command_queue * commandQueues, cl_uint numEventsInWaitList, const cl_event * eventWaitList, cl_event * events)`

Description Vector-vector product with double elements and performs the rank 1 operation A.
Vector-vector products:

- $A \leftarrow \alpha xy^T + A$

Parameters

in	*order*	Row/column order.
in	*M*	Number of rows in matrix *A*.
in	*N*	Number of columns in matrix *A*.
in	*alpha*	Specifies the scalar alpha.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of elements for the first element in vector *X*.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset in number of elements for the first element in vector *Y*.
in	*incy*	Increment for the elements of *Y*. Must not be zero.
in/ out	*A*	Buffer object storing matrix *A*. On exit, *A* is overwritten by the updated matrix.
in	*offa*	Offset in number of elements for the first element in matrix *A*.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *N* when the order parameter is set to `clAmdBlasRowMajor`, or less than *M* when the parameter is set to `clAmdBlasColumnMajor`.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- otherwise, the same error codes as the `clAmdBlasSger()` function.

3.9 xGERU - GEneral matrix Rank 1 operation

3.9.1 Cgeru

Vector-vector product with float-complex elements and performs the rank 1 operation A

Function

```
clAmdBlasStatus clAmdBlasCgeru ( clAmdBlasOrder order, size_t M, size_t N,
cl_float2 alpha, const cl_mem X, size_t offx, int incx, const cl_mem Y, size_t
offy, int incy, cl_mem A, size_t offa, size_t lda, cl_uint numCommandQueues,
cl_command_queue * commandQueues, cl_uint numEventsInWaitList, const
cl_event * eventWaitList, cl_event * events )
```

Description

Vector-vector product with float-complex elements and performs the rank 1 operation A. Vector-vector products:

- $A \leftarrow \alpha xy^T + A$

Parameters

in	*order*	Row/column order.
in	*M*	Number of rows in matrix *A*.
in	*N*	Number of columns in matrix *A*.
in	*alpha*	Specifies the scalar alpha.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of elements for the first element in vector *X*.
in	*incx*	Increment for the elements of X. Must not be zero.
in	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset in number of elements for the first element in vector *Y*.
in	*incy*	Increment for the elements of *Y*. Must not be zero.
in/ out	*A*	Buffer object storing matrix *A*. On exit, *A* is overwritten by the updated matrix.
in	*offa*	Offset in number of elements for the first element in matrix *A*.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *N* when the *order* parameter is set to `clAmdBlasRowMajor`, or less than *M* when the parameter is set to `clAmdBlasColumnMajor`.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Vector-vector product with float-complex elements and performs the rank 1 operation A

Returns

- clAmdBlasSuccess on success;
- clAmdBlasNotInitialized if clAmdBlasSetup() was not called;
- clAmdBlasInvalidValue if invalid parameters are passed:
  - *M*, *N* or
    either *incx* or *incy* is zero, or
  - a leading dimension is invalid;
- clAmdBlasInvalidMemObject if *A*, *X*, or *Y* object is invalid, or an image object rather than the buffer one;
- clAmdBlasOutOfResources if you use image-based function implementation and no suitable scratch image available;
- clAmdBlasOutOfHostMemory if the library can't allocate memory for internal structures;
- clAmdBlasInvalidCommandQueue if the passed command queue is invalid;
- clAmdBlasInvalidContext if a context a passed command queue belongs to was released;
- clAmdBlasInvalidOperation if kernel compilation relating to a previous call has not completed for any of the target devices;
- clAmdBlasCompilerNotAvailable if a compiler is not available;
- clAmdBlasBuildProgramFailure if there is a failure to build a program executable.

3.9.2 Zgeru

Vector-vector product with double-complex elements and performs the rank 1 operation A

Function

```
clAmdBlasStatus clAmdBlasZgeru ( clAmdBlasOrder order, size_t M, size_t N,
cl_double2 alpha, const cl_mem X, size_t offx, int incx, const cl_mem Y,
size_t offy, int incy, cl_mem A, size_t offa, size_t lda, cl_uint
numCommandQueues, cl_command_queue * commandQueues, cl_uint
numEventsInWaitList, const cl_event * eventWaitList, cl_event * events )
```

Description

Vector-vector product with double-complex elements and performs the rank 1 operation A. Vector-vector products:

- $A \leftarrow \alpha xy^T + A$

Parameters

in	*order*	Row/column order.
in	*M*	Number of rows in matrix *A*.
in	*N*	Number of columns in matrix *A*.
in	*alpha*	Specifies the scalar alpha.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of elements for the first element in vector *X*.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset in number of elements for the first element in vector *Y*.
in	*incy*	Increment for the elements of *Y*. Must not be zero.
in/ out	*A*	Buffer object storing matrix *A*. On exit, *A* is overwritten by the updated matrix.
in	*offa*	Offset in number of elements for the first element in matrix *A*.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *N* when the *order* parameter is set to `clAmdBlasRowMajor`, or less than *M* when the parameter is set to `clAmdBlasColumnMajor`.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects in each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- otherwise, the same error codes as the `clAmdBlasCgeru()` function.

3.10 xGERC - GEneral matrix Rank 1 operation

3.10.1 Cgerc

Vector-vector product with float-complex elements and performs the rank 1 operation A

Function

```
clAmdBlasStatus clAmdBlasCgerc ( clAmdBlasOrder order, size_t M, size_t N,
cl_float2 alpha, const cl_mem X, size_t offx, int incx, const cl_mem Y, size_t
offy, int incy, cl_mem A, size_t offa, size_t lda, cl_uint numCommandQueues,
cl_command_queue * commandQueues, cl_uint numEventsInWaitList, const
cl_event * eventWaitList, cl_event * events )
```

Description

Vector-vector product with float-complex elements and performs the rank 1 operation A. Vector-vector products:

- $A \leftarrow \alpha x y^H + A$

Parameters

in	*order*	Row/column order.
in	*M*	Number of rows in matrix *A*.
in	*N*	Number of columns in matrix *A*.
in	*alpha*	Specifies the scalar alpha.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of elements for the first element in vector *X*.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in/ out	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset in number of elements for the first element in vector *Y*.
in	*incy*	Increment for the elements of *Y*. Must not be zero.
in/ out	*A*	Buffer object storing matrix *A*. On exit, *A* is overwritten by the updated matrix.
in	*offa*	Offset in number of elements for the first element in matrix *A*.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *N* when the *order* parameter is set to `clAmdBlasRowMajor`, or less than *M* when the parameter is set to `clAmdBlasColumnMajor`.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Vector-vector product with float-complex elements and performs the rank 1 operation A

Returns

- clAmdBlasSuccess on success;
- clAmdBlasNotInitialized if clAmdBlasSetup() was not called;
- clAmdBlasInvalidValue if invalid parameters are passed:
  - *M*, *N* or
    either *incx* or *incy* is zero, or
  - a leading dimension is invalid;
- clAmdBlasInvalidMemObject if A, X, or Y object is invalid, or an image object rather than the buffer one;
- clAmdBlasOutOfResources if you use image-based function implementation and no suitable scratch image available;
- clAmdBlasOutOfHostMemory if the library can't allocate memory for internal structures;
- clAmdBlasInvalidCommandQueue if the passed command queue is invalid;
- clAmdBlasInvalidContext if a context a passed command queue belongs to was released;
- clAmdBlasInvalidOperation if kernel compilation relating to a previous call has not completed for any of the target devices;
- clAmdBlasCompilerNotAvailable if a compiler is not available;
- clAmdBlasBuildProgramFailure if there is a failure to build a program executable.

3.10.2 Zgerc

Vector-vector product with double-complex elements and performs the rank 1 operation A

Function

```
clAmdBlasStatus clAmdBlasZgerc ( clAmdBlasOrder order, size_t M, size_t N,
cl_double2 alpha, const cl_mem X, size_t offx, int incx, const cl_mem Y,
size_t offy, int incy, cl_mem A, size_t offa, size_t lda, cl_uint
numCommandQueues, cl_command_queue * commandQueues, cl_uint
numEventsInWaitList, const cl_event * eventWaitList, cl_event * events )
```

Description

Vector-vector product with double-complex elements and performs the rank 1 operation A. Vector-vector products:

- $A \leftarrow \alpha xy^H + A$

Parameters

in	*order*	Row/column order.
in	*M*	Number of rows in matrix *A*.
in	*N*	Number of columns in matrix *A*.
in	*alpha*	Specifies the scalar alpha.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of elements for the first element in vector *X*.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset in number of elements for the first element in vector *Y*.
in	*incy*	Increment for the elements of *Y*. Must not be zero.
in/ out	*A*	Buffer object storing matrix *A*. On exit, *A* is overwritten by the updated matrix.
in	*offa*	Offset in number of elements for the first element in matrix *A*.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *N* when the *order* parameter is set to `clAmdBlasRowMajor`, or less than *M* when the parameter is set to `clAmdBlasColumnMajor`.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- otherwise, the same error codes as the `clAmdBlasCgerc()` function.

3.11 xSYR - SYmmetric Rank 1 update

3.11.1 Ssyr

Symmetric rank 1 operation with a general triangular matrix and float elements

Function

```
clAmdBlasStatus clAmdBlasSsyr ( clAmdBlasOrder order, clAmdBlasUplo uplo,
size_t N, cl_float alpha, const cl_mem X, size_t offx, int incx, cl_mem A,
size_t offa, size_t lda, cl_uint numCommandQueues, cl_command_queue *
commandQueues, cl_uint numEventsInWaitList, const cl_event *
eventWaitList, cl_event * events )
```

Description

Symmetric rank 1 operation with a general triangular matrix and float elements.
Symmetric rank 1 operation:

- $A \leftarrow \alpha x x^T + A$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*N*	Number of columns in matrix *A*.
in	*alpha*	The factor of matrix *A*.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in/ out	*A*	Buffer object storing matrix *A*.
in	*offa*	Offset of first element of matrix *A* in buffer object.
in	*lda*	Leading dimension of matrix *A*. It cannot less than *N*.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasNotInitialized` if `clAmdBlasSetup()` was not called;
- `clAmdBlasInvalidValue` if invalid parameters are passed:
  - *N* is zero, or
  - either *incx* is zero, or
  - the leading dimension is invalid;
- `clAmdBlasInvalidMemObject` if either *A*, *X* object is Invalid, or an image object rather than the buffer one;
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures;
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid;
- `clAmdBlasInvalidContext` if a context a passed command queue belongs to was released;
- `clAmdBlasInvalidOperation` if kernel compilation relating to a previous call has not completed for any of the target devices;
- `clAmdBlasCompilerNotAvailable` if a compiler is not available;
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

3.11.2 Dsyr

Symmetric rank 1 operation with a general triangular matrix and double elements

Function

```
clAmdBlasStatus clAmdBlasDsyr ( clAmdBlasOrder order, clAmdBlasUplo uplo,
size_t N, cl_double alpha, const cl_mem X, size_t offx, int incx, cl_mem A,
size_t offa, size_t lda, cl_uint numCommandQueues, cl_command_queue *
commandQueues, cl_uint numEventsInWaitList, const cl_event *
eventWaitList, cl_event * events )
```

Description

Symmetric rank 1 operation with a general triangular matrix and double elements.
Symmetric rank 1 operation:

- $A \leftarrow \alpha x x^T + A$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*N*	Number of columns in matrix *A*.
in	*alpha*	The factor of matrix *A*.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in/ out	*A*	Buffer object storing matrix *A*.
in	*offa*	Offset of first element of matrix *A* in buffer object.
in	*lda*	Leading dimension of matrix *A*. It cannot less than *N*.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- otherwise, the same error codes as the `clAmdBlasSsyr()` function.

3.12 xHER - HErmitian Rank 1 operation

3.12.1 Cher

Hermitian rank 1 operation with a general triangular matrix and float-complex elements

Function

```
clAmdBlasStatus clAmdBlasCher ( clAmdBlasOrder order, clAmdBlasUplo uplo,
size_t N, cl_float alpha, const cl_mem X, size_t offx, int incx, cl_mem A,
size_t offa, size_t lda, cl_uint numCommandQueues, cl_command_queue *
commandQueues, cl_uint numEventsInWaitList, const cl_event *
eventWaitList, cl_event * events )
```

Description

Hermitian rank 1 operation with a general triangular matrix and float-complex elements.
hermitian rank 1 operation:

- $A \leftarrow \alpha x x^H + A$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*N*	Number of columns in matrix *A*.
in	*alpha*	The factor of matrix *A* (a scalar float value).
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of elements for the first element in vector *X*.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in/ out	*A*	Buffer object storing matrix *A*.
in	*offa*	Offset in number of elements for the first element in matrix *A*.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *N*.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Hermitian rank 1 operation with a general triangular matrix and float-complex elements

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasNotInitialized` if `clAmdBlasSetup()` was not called;
- `clAmdBlasInvalidValue` if invalid parameters are passed:
  - *N* is zero, or
  - either *incx* is zero, or
  - the leading dimension is invalid;
- `clAmdBlasInvalidMemObject` if either *A*, *X* object is Invalid, or an image object rather than the buffer one;
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures;
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid;
- `clAmdBlasInvalidContext` if a context a passed command queue belongs to was released;
- `clAmdBlasInvalidOperation` if kernel compilation relating to a previous call has not completed for any of the target devices;
- `clAmdBlasCompilerNotAvailable` if a compiler is not available;
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

Examples `example_cher.c`.

3.12.2 Zher

Hermitian rank 1 operation with a general triangular matrix and double-complex elements

Function

```
clAmdBlasStatus clAmdBlasZher ( clAmdBlasOrder order, clAmdBlasUplo uplo,
size_t N, cl double alpha, const cl_mem X, size_t offx, int incx, cl_mem A,
size_t offa, size_t lda, cl_uint numCommandQueues, cl_command_queue *
commandQueues, cl_uint numEventsInWaitList, const cl_event *
eventWaitList, cl_event * events )
```

Description

Hermitian rank 1 operation with a general triangular matrix and double-complex elements. hermitian rank 1 operation:

- $A \leftarrow \alpha x x^H + A$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*N*	Number of columns in matrix *A*.
in	*alpha*	The factor of matrix *A (a scalar double value)*.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of elements for the first element in vector *X*.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in/ out	*A*	Buffer object storing matrix *A*.
in	*offa*	Offset in number of elements for the first element in matrix *A*.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *N*.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- otherwise, the same error codes as the `clAmdBlasCher()` function.

3.13 xSYR2 - SYmmetric Rank 2 update

3.13.1 Ssyr2

Symmetric rank 2 operation with a general triangular matrix and float elements

Function

```
clAmdBlasStatus clAmdBlasSsyr2 ( clAmdBlasOrder order, clAmdBlasUplo uplo,
size_t N, cl_float alpha, const cl_mem X, size_t offx, int incx, const cl_mem
Y, size_t offy, int incy, cl_mem A, size_t offa, size_t lda, cl_uint
numCommandQueues, cl_command_queue * commandQueues, cl_uint
numEventsInWaitList, const cl_event * eventWaitList, cl_event * events )
```

Description

Symmetric rank 2 operation with a general triangular matrix and float elements.
Symmetric rank 2 operation:

- $A \leftarrow \alpha xy^T + \alpha yx^T + A$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*N*	Number of columns in matrix *A*.
in	*alpha*	The factor of matrix *A*.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object.
in	*incx*	Increment for the elements of vector *X*. Must not be zero.
in	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset of first element of vector *Y* in buffer object.
in	*incy*	Increment for the elements of *Y*. Must not be zero.
in/ out	*A*	Buffer object storing matrix *A*.
in	*offa*	Offset of first element of matrix *A* in buffer object.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *N*.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Symmetric rank 2 operation with a general triangular matrix and float elements (Cont.)

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasNotInitialized` if `clAmdBlasSetup()` was not called;
- `clAmdBlasInvalidValue` if invalid parameters are passed:
  - either *N* is zero, or
  - either *incx* or *incy* is zero, or
  - the leading dimension is invalid;
- `clAmdBlasInvalidMemObject` if either *A*, *X*, or *Y* object is Invalid, or an image object rather than the buffer one;
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures;
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid;
- `clAmdBlasInvalidContext` if a context a passed command queue belongs to was released;
- `clAmdBlasInvalidOperation` if kernel compilation relating to a previous call has not completed for any of the target devices;
- `clAmdBlasCompilerNotAvailable` if a compiler is not available;
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

3.13.2 Dsyr2

Symmetric rank 2 operation with a general triangular matrix and double elements

Function

```
clAmdBlasStatus clAmdBlasDsyr2 ( clAmdBlasOrder order, clAmdBlasUplo uplo,
size_t N, cl_double alpha, const cl_mem X, size_t offx, int incx, const cl_mem
Y, size_t offy, int incy, cl_mem A, size_t offa, size_t lda, cl_uint
numCommandQueues, cl_command_queue * commandQueues, cl_uint
numEventsInWaitList, const cl_event * eventWaitList, cl_event * events )
```

Description

Symmetric rank 2 operation with a general triangular matrix and double elements.
Symmetric rank 2 operation:

- $A \leftarrow \alpha xy^T + \alpha yx^T + A$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*N*	Number of columns in matrix *A*.
in	*alpha*	The factor of matrix *A*.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset of first element of vector *Y* in buffer object.
in	*incy*	Increment for the elements of *Y*. Must not be zero.
in/ out	*A*	Buffer object storing matrix *A*.
in	*offa*	Offset of first element of matrix *A* in buffer object.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *N*.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasNotInitialized` if `clAmdBlasSetup()` was not called;
- `clAmdBlasInvalidValue` if invalid parameters are passed:
  - either *N* is zero, or
  - either *incx* or *incy* is zero, or
  - the leading dimension is invalid;
- `clAmdBlasInvalidMemObject` if either *A*, *X*, or *Y* object is Invalid, or an image object rather than the buffer one;
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures;
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid;
- `clAmdBlasInvalidContext` if a context a passed command queue belongs to was released;
- `clAmdBlasInvalidOperation` if kernel compilation relating to a previous call has not completed for any of the target devices;
- `clAmdBlasCompilerNotAvailable` if a compiler is not available;
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

3.14 xHER2 - HErmitian Rank 2 update

3.14.1 Cher2

Hermitian rank 2 operation with general triangular matrix and float-complex elements

Function

```
clAmdBlasStatus clAmdBlasCher2 ( clAmdBlasOrder order, clAmdBlasUplo uplo,
size_t N, cl_float2 alpha, const cl_mem X, size_t offx, int incx, const cl_mem
Y, size_t offy, int incy, cl_mem A, size_t offa, size_t lda, cl_uint
numCommandQueues, cl_command_queue * commandQueues, cl_uint
numEventsInWaitList, const cl_event * eventWaitList, cl_event * events )
```

Description

Hermitian rank 2 operation with a general triangular matrix and float-complex elements.
Hermitian rank 2 operation:

- $A \leftarrow \alpha xy^H + \overline{\alpha} yx^H + A$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*N*	Number of columns in matrix *A*.
in	*alpha*	The factor of matrix *A*.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of elements for the first element in vector *X*.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset in number of elements for the first element in vector *Y*.
in	*incy*	Increment for the elements of *Y*. Must not be zero.
in/ out	*A*	Buffer object storing matrix *A*.
in	*offa*	Offset in number of elements for the first element in matrix *A*.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *N*.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Hermitian rank 2 operation with general triangular matrix and float-complex elements (Cont.)

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasNotInitialized` if `clAmdBlasSetup()` was not called;
- `clAmdBlasInvalidValue` if invalid parameters are passed:
  - either *N* is zero, or
  - either *incx* or *incy* is zero, or
  - the leading dimension is invalid;
- `clAmdBlasInvalidMemObject` if either *A*, *X*, or *Y* object is Invalid, or an image object rather than the buffer one;
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures;
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid;
- `clAmdBlasInvalidContext` if a context a passed command queue belongs to was released;
- `clAmdBlasInvalidOperation` if kernel compilation relating to a previous call has not completed for any of the target devices;
- `clAmdBlasCompilerNotAvailable` if a compiler is not available;
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

3.14.2 Zher2

Hermitian rank 2 operation with a general triangular matrix and double-complex elements

Function

```
clAmdBlasStatus clAmdBlasZher2 ( clAmdBlasOrder order, clAmdBlasUplo uplo,
size_t N, cl_double2 alpha, const cl_mem X, size_t offx, int incx, const
cl_mem Y, size_t offy, int incy, cl_mem A, size_t offa, size_t lda, cl_uint
numCommandQueues, cl_command_queue * commandQueues, cl_uint
numEventsInWaitList, const cl_event * eventWaitList, cl_event * events )
```

Description

Hermitian rank 2 operation with a general triangular matrix and double-complex elements. Hermitian rank 2 operation:

- $A \leftarrow \alpha xy^H + \overline{\alpha} yx^H + A$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*N*	Number of columns in matrix *A*.
in	*alpha*	The factor of matrix *A*.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of elements for the first element in vector *X*.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset in number of elements for the first element in vector *Y*.
in	*incy*	Increment for the elements of *Y*. Must not be zero.
in/ out	*A*	Buffer object storing matrix *A*.
in	*offa*	Offset in number of elements for the first element in matrix *A*.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *N*.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- otherwise, the same error codes as the `clAmdBlasCher2()` function.

Examples

`example_zher2.c`.

3.15 xTPMV - Triangle Packed Matrix-Vector multiple

3.15.1 Stpmv

Matrix-vector product with a packed triangular matrix and float elements

Function

```
clAmdBlasStatus clAmdBlasStpmv( clAmdBlasOrder order, clAmdBlasUplo uplo,
clAmdBlasTranspose trans, clAmdBlasDiag diag, size_t N, const cl_mem AP, size_t
offa, cl_mem AP, size_t offa, cl_mem x, cl_mem scratcbbuff, cl_uint
numCommandQueues, cl_command_queue *commandQueues, cl_uint numEventsInWaitList,
const cl_event *eventWaitList, cl_event *events);
```

Description

Matrix-vector product with a packed triangular matrix and float elements.
Matrix-vector products:

- $x \leftarrow A\,x$
- $x \leftarrow A^T\,x$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*trans*	How the matrix *AP* is to be transposed.
in	*diag*	Specify whether matrix *AP* is unit triangular.
in	*N*	Number of rows/columns in matrix *AP*.
in	*AP*	Buffer object storing packed-matrix *AP*.
in	*offa*	Offset in number of elements for first element in matrix AP.
in/ out	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of elements for first element in vector *X*.
in	*incx*	Increment for the elements of *X*. Must not be zero
in	*scratchBuff*	Temporary cl_mem scratch buffer object which can hold a minimum of (1 + (*N*-1)*abs(*incx*)) elements.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Matrix-vector product with a packed triangular matrix and float elements (Cont.)

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasNotInitialized` if `clAmdBlasSetup`() was not called;
- `clAmdBlasInvalidValue` if invalid parameters are passed:

- either *N* or *incx* is zero

- `clAmdBlasInvalidMemObject` if either *AP* or *X* object is Invalid, or an image object rather than the buffer one;
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures;
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid;
- `clAmdBlasInvalidContext` if a context a passed command queue belongs to was released;
- `clAmdBlasInvalidOperation` if kernel compilation relating to a previous call has not completed for any of the target devices;
- `clAmdBlasCompilerNotAvailable` if a compiler is not available;
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

3.15.2 Dtpmv

Matrix-vector product with a packed triangular matrix and double elements

Function

```
clAmdBlasStatus clAmdBlasDtpmv( clAmdBlasOrder order, clAmdBlasUplo uplo,
clAmdBlasTranspose trans, clAmdBlasDiag diag, size_t N, const cl_mem AP, size_t
offa, cl_mem X, size_t offx, int incx, cl_mem scratchBuff, cl_uint
numCommandQueues, cl_command_queue *commandQueues, cl_uint numEventsInWaitList,
const cl_event *eventWaitList, cl_event *events);
```

Description

Matrix-vector product with a packed triangular matrix and double elements.
Matrix-vector products:

- $x \leftarrow A\,x$
- $x \leftarrow A^T\,x$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*trans*	How matrix *AP* is to be transposed.
in	*diag*	Specify whether matrix *AP* is unit triangular.
in	*N*	Number of rows/columns in matrix *AP*.
in	*AP*	Buffer object storing matrix *AP* in packed format.
in	*offa*	Offset in number of elements for first element in matrix *AP*.
in/ out	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of elements for first element in vector *X*.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in	*scratchBuff*	Temporary cl_mem scratch buffer object which can hold a minimum of (1 + (*N*-1)*abs(*incx*)) elements.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- otherwise, the same error codes as the `clAmdBlasStpmv()` function.

3.15.3 Ctpmv

Matrix-vector product with a packed triangular matrix and float-complex elements

Function

```
clAmdBlasStatus clAmdBlasCtpmv( clAmdBlasOrder order, clAmdBlasUplo uplo,
clAmdBlasTranspose trans, clAmdBlasDiag diag, size_t N, const cl_mem AP, size_t
offa, cl_mem X, size_t offx, int incx, cl_mem scratchBuff, cl_uint
numCommandQueues, cl_command_queue *commandQueues, cl_uint numEventsInWaitList,
const cl_event *eventWaitList, cl_event *events);
```

Description

Matrix-vector product with a packed triangular matrix and float-complex elements.
Matrix-vector products:

- $x \leftarrow A\ x$
- $x \leftarrow A^T\ x$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*trans*	How matrix *AP* is to be transposed.
in	*diag*	Specify whether matrix *AP* is unit triangular.
in	*N*	Number of rows/columns in matrix *AP*.
in	*AP*	Buffer object storing matrix *AP* in packed format.
in	*offa*	Offset in number of elements for first element in matrix *AP*.
in/ out	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of elements for first element in vector *X*.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in	*scratchBuff*	Temporary cl_mem scratch buffer object which can hold a minimum of (1 + (*N*-1)*abs(*incx*)) elements.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- The same result as the `clAmdBlasStpmv()` function.

3.15.4 Ztpmv

Matrix-vector product with a packed triangular matrix and double-complex elements

Function `clAmdBlasStatus clAmdBlasZtpmv( clAmdBlasOrder` *`order`*`, clAmdBlasUplo` *`uplo`*`, clAmdBlasTranspose` *`trans`*`, clAmdBlasDiag` *`diag`*`, size_t` *`N`*`, const cl_mem` *`AP`*`, size_t` *`offa`*`, cl_mem` *`X`*`, size_t` *`offx`*`, int` *`incx`*`, cl_mem` *`scratchBuff`*`, cl_uint` *`numCommandQueues`*`, cl_command_queue *`*`commandQueues`*`, cl_uint` *`numEventsInWaitList`*`, const cl_event *`*`eventWaitList`*`, cl_event *`*`events`*`);`

Description Matrix-vector product with a packed triangular matrix and double-complex elements. Matrix-vector products:

- $x \leftarrow A\ x$
- $x \leftarrow A^T\ x$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*trans*	How matrix *AP* is to be transposed.
in	*diag*	Specify whether matrix *AP* is unit triangular
in	*N*	Number of rows/columns in banded matrix *AP*.
in	*AP*	Buffer object storing matrix *AP* in packed format.
in	*offa*	Offset in number of elements for first element in matrix *AP*.
in/ out	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of elements for first element in vector *X*.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in	*scratchBuff*	Temporary cl_mem scratch buffer object which can hold a minimum of (1 + (*N*-1)*abs(*incx*)) elements.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- The same result as the `clAmdBlasDtbmv`() function.

3.16 xSPR - Symmetric Packed matrix Rank

3.16.1 Sspr

Symmetric rank 1 operation with a general triangular packed-matrix and float elements

Function

```
clAmdBlasStatus clAmdBlasSspr( clAmdBlasOrder order, clAmdBlasUplo uplo, size_t
N, cl_float alpha, const cl_mem X, size_t offx, int incx, cl_mem AP, size_t
offa, cl_uint numCommandQueues, cl_command_queue* commandQueues, cl_uint
numEventsInWaitList, const cl_event* eventWaitList, cl_event* events);
```

Description

Symmetric packed matrix rank 1 update.
Symmetric rank 1 operation:

- $A \leftarrow \alpha\, x\, x^T + A$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*N*	Number of columns in matrix *AP*.
in	*alpha*	The factor of matrix *AP*.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in/ out	*AP*	Buffer object storing packed-matrix *AP*.
in	*offa*	Offset of first element of matrix *AP* in buffer object.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasNotInitialized` if `clAmdBlasSetup`() was not called;
- b clAmdBlasInvalidValue if invalid parameters are passed:
  - *N* is zero, or
  - either *incx* is zero
- `clAmdBlasInvalidMemObject` if either *AP or X* object is Invalid, or an image object rather than the buffer one;
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures;
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid;
- `clAmdBlasInvalidContext` if a context a passed command queue belongs to was released;
- `clAmdBlasInvalidOperation` if kernel compilation relating to a previous call has not completed for any of the target devices;
- `clAmdBlasCompilerNotAvailable` if a compiler is not available;
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

3.16.2 Dspr

Symmetric rank 1 operation with a general triangular packed-matrix and double elements

Function `clAmdBlasStatus clAmdBlasDspr( clAmdBlasOrder order, clAmdBlasUplo uplo, size_t N, cl_double alpha, const cl_mem X, size_t offx, int incx, cl_mem AP, size_t offa, cl_uint numCommandQueues, cl_command_queue* commandQueues, cl_uint numEventsInWaitList, const cl_event* eventWaitList, cl_event* events);`

Description Symmetric rank 1 operation with a general triangular packed-matrix and double elements. Symmetric rank 1 operation:

- $A \leftarrow \alpha \times x^T + A$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*N*	Number of columns in matrix *AP*.
in	*alpha*	The factor of matrix *AP*.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object.
in	*incx*	Increment for the elements of *X*. It cannot be zero.
in/ out	*AP*	Buffer object storing packed-matrix *AP*.
in	*offa*	Offset of first element of matrix *AP* in buffer object.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- otherwise, the same error codes as the `clAmdBlasSspr`() function.

3.17 xTPSV - Triangle Packed matrix Solve Vector

3.17.1 Stpsv

Solving triangular packed matrix problems with float elements

Function

```
clAmdBlasStatus clAmdBlasStpsv( clAmdBlasOrder order, clAmdBlasUplo uplo,
clAmdBlasTranspose trans, clAmdBlasDiag diag, size_t N, const cl_mem A, size_t
offa, cl_mem X, size_t offx, int incx, cl_uint numCommandQueues,
cl_command_queue *commandQueues, cl_uint numEventsInWaitList, const cl_event
*eventWaitList, cl_event *events);
```

Description

Triangle packed matrix vector solve.
Matrix-vector products:

- $A\ x \leftarrow x$
- $A^T\ x \leftarrow x$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*trans*	How matrix *A* is to be transposed.
in	*diag*	Specify whether matrix *A* is unit triangular.
in	*N*	Number of rows/columns in matrix *A*.
in	*A*	Buffer object storing matrix in packed format *A*.
in	*offa*	Offset in number of elements for first element in matrix *A*.
in/ out	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of elements for first element in matrix *X*.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasNotInitialized` if `clAmdBlasSetup`() was not called;
- clAmdBlasInvalidValue if invalid parameters are passed:
  - either *N* or *incx* is zero, or
  - the leading dimension is invalid;
- `clAmdBlasInvalidMemObject` if either *A* or *X* object is Invalid, or an image object rather than the buffer one;
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures;
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid;
- `clAmdBlasInvalidContext` if a context a passed command queue belongs to was released;
- `clAmdBlasInvalidOperation` if kernel compilation relating to a previous call has not completed for any of the target devices;
- `clAmdBlasCompilerNotAvailable` if a compiler is not available;
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

3.17.2 Dtpsv

Solving triangular packed matrix problems with double elements

Function `clAmdBlasStatus clAmdBlasDtpsv(clAmdBlasOrder order, clAmdBlasUplo uplo, clAmdBlasTranspose trans, clAmdBlasDiag diag, size_t N, const cl_mem A, size_t offa, cl_mem X, size_t offx, int incx, cl_uint numCommandQueues, cl_command_queue *commandQueues, cl_uint numEventsInWaitList, const cl_event *eventWaitList, cl_event *events);`

Description Solving triangular packed matrix problems with double elements.
Matrix-vector products:

- $A\ x \leftarrow x$
- $A^T\ x \leftarrow x$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*trans*	How matrix *A* is to be transposed.
in	*diag*	Specify whether matrix *A* is unit triangular.
in	*N*	Number of rows/columns in matrix *A*.
in	*A*	Buffer object storing matrix in packed format *A*.
in	*offa*	Offset in number of elements for first element in matrix *A*.
in/ out	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of elements for first element in vector *X*.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasNotInitialized` if `clAmdBlasSetup`() was not called;
- clAmdBlasInvalidValue if invalid parameters are passed:
  - either *N* or *incx* is zero, or
  - the leading dimension is invalid;
- `clAmdBlasInvalidMemObject` if either *A* or *X* object is Invalid, or an image object rather than the buffer one;
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures;
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid;
- `clAmdBlasInvalidContext` if a context a passed command queue belongs to was released;
- `clAmdBlasInvalidOperation` if kernel compilation relating to a previous call has not completed for any of the target devices;
- `clAmdBlasCompilerNotAvailable` if a compiler is not available;
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

3.17.3 Ctpsv

Solving triangular packed matrix problems with float complex elements

Function

```
clAmdBlasStatus clAmdBlasCtpsv( clAmdBlasOrder order, clAmdBlasUplo uplo,
clAmdBlasTranspose trans, clAmdBlasDiag diag, size_t N, const cl_mem A, size_t
offa, cl_mem X, size_t offx, int incx, cl_uint numCommandQueues,
cl_command_queue *commandQueues, cl_uint numEventsInWaitList, const cl_event
*eventWaitList, cl_event *events);
```

Description

Solving triangular packed matrix problems with float complex elements.
Matrix-vector products:

- $A\ x \leftarrow x$
- $A^T\ x \leftarrow x$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*trans*	How matrix *A* is to be transposed.
in	*diag*	Specify whether matrix *A* is unit triangular.
in	*N*	Number of rows/columns in matrix *A*.
in	*A*	Buffer object storing matrix in packed format *A*.
in	*offa*	Offset in number of elements for first element in matrix *A*.
in/ out	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of elements for first element in vector *X*.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasNotInitialized` if `clAmdBlasSetup`() was not called;
- `clAmdBlasInvalidValue` if invalid parameters are passed:
  - either *N* or *incx* is zero, or
  - the leading dimension is invalid;
- `clAmdBlasInvalidMemObject` if either *A* or *X* object is Invalid, or an image object rather than the buffer one;
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures;
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid;
- `clAmdBlasInvalidContext` if a context a passed command queue belongs to was released;
- `clAmdBlasInvalidOperation` if kernel compilation relating to a previous call has not completed for any of the target devices;
- `clAmdBlasCompilerNotAvailable` if a compiler is not available;
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

3.17.4 Ztpsv

Solving triangular packed matrix problems with double-complex elements

Function

`clAmdBlasStatus clAmdBlasZtpsv( clAmdBlasOrder` *`order`*`, clAmdBlasUplo` *`uplo`*`, clAmdBlasTranspose` *`trans`*`, clAmdBlasDiag` *`diag`*`, size_t` *`N`*`, const cl_mem` *`A`*`, size_t` *`offa`*`, cl_mem` *`X`*`, size_t` *`offx`*`, int` *`incx`*`, cl_uint` *`numCommandQueues`*`, cl_command_queue *`*`commandQueues`*`, cl_uint` *`numEventsInWaitList`*`, const cl_event *`*`eventWaitList`*`, cl_event *`*`events`*`);`

Description

Solving triangular packed matrix problems with double-complex elements.
Matrix-vector products:

- $A\ x \leftarrow x$
- $A^T\ x \leftarrow x$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*trans*	How matrix *A* is to be transposed.
in	*diag*	Specify whether matrix *A* is unit triangular
in	*N*	Number of rows/columns in matrix *A.*
in	*A*	Buffer object storing matrix *A* in packed format.
in	*offa*	Offset in number of elements for first element in matrix *A.*
in/ out	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of elements for first element in vector *X*.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- clAmdBlasNotInitialized if `clAmdBlasSetup`() was not called;
- clAmdBlasInvalidValue if invalid parameters are passed:
  - either *N* or *incx* is zero, or
  - the leading dimension is invalid;
- `clAmdBlasInvalidMemObject` if either *A* or X object is Invalid, or an image object rather than the buffer one;
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures;
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid;
- `clAmdBlasInvalidContext` if a context a passed command queue belongs to was released;
- `clAmdBlasInvalidOperation` if kernel compilation relating to a previous call has not completed for any of the target devices;
- `clAmdBlasCompilerNotAvailable` if a compiler is not available;
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

3.18 xSPMV - Symmetric Packed Matrix Vector

3.18.1 Sspmv

Matrix-vector product with a symmetric packed-matrix and float elements

Function

```
clAmdBlasStatus clAmdBlasSspmv( clAmdBlasOrder order, clAmdBlasUplo uplo, size_t
N, cl_float alpha, const cl_mem AP, size_t offa, const cl_mem X, size_t offx,
int incx, cl_float beta, cl_mem Y, size_t offy, int incy, cl_uint
numCommandQueues, cl_command_queue *commandQueues, cl_uint numEventsInWaitList,
const cl_event *eventWaitList, cl_event *events);
```

Description

Symmetric packed matrix vector multiply.
Matrix-vector products:

- $y \leftarrow \alpha A x + \beta y$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*N*	Number of rows/columns in matrix *AP*.
in	*alpha*	The factor of matrix *AP*.
in	*AP*	Buffer object storing matrix *AP*.
in	*offa*	Offset in number of elements for first element in matrix *AP*.
in	*X*	Buffer object storing matrix *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in	*beta*	The factor of vector *Y*.
in/ out	*Y*	Buffer object storing matrix *Y*.
in	*offy*	Offset of first element of vector *Y* in buffer object. Counted in elements
in	*incy*	Increment for the elements of vector *Y*. It cannot be zero
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Matrix-vector product with a symmetric packed-matrix and float elements (Cont.)

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasNotInitialized` if `clAmdBlasSetup`() was not called;
- clAmdBlasInvalidValue if invalid parameters are passed:
  - *N* is zero, or
  - either *incx* or *incy* is zero, or
  - the matrix sizes or the vector sizes along with the increments lead to accessing outsize of any of the buffers;
- `clAmdBlasInvalidMemObject` if either *AP*, *X*, or *Y* object is invalid, or an image object rather than the buffer one;
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures;
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid;
- `clAmdBlasInvalidContext` if a context a passed command queue belongs to was released;
- `clAmdBlasInvalidOperation` if kernel compilation relating to a previous call has not completed for any of the target devices;
- `clAmdBlasCompilerNotAvailable` if a compiler is not available;
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

3.18.2 Dspmv

Matrix-vector product with symmetric packed-matrix and double elements

Function

```
clAmdBlasStatus clAmdBlasDspmv( clAmdBlasOrder order, clAmdBlasUplo uplo, size_t
N, cl_double alpha, const cl_mem AP, size_t offa, const cl_mem X, size_t offx,
int incx, cl_double beta, cl_mem Y, size_t offy, int incy, cl_uint
numCommandQueues, cl_command_queue *commandQueues, cl_uint numEventsInWaitList,
const cl_event *eventWaitList, cl_event *events);
```

Description

Matrix-vector product with a symmetric packed-matrix and double elements.
Matrix-vector products:

- $y \leftarrow \alpha A x + \beta y$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*N*	Number of rows and columns in matrix *AP*.
in	*alpha*	The factor of matrix *AP*.
in	*offa*	Offset in number of elements for first element in matrix *AP*.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of vector *X*. It cannot be zero.
in	*beta*	The factor of vector *Y*.
in/ out	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset of first element of vector *Y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of vector *Y*. It cannot be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- otherwise, the same error codes as the `clAmdBlasSspmv`() function.

3.19 xHPMV - Hermitian Product Matrix Vector

3.19.1 Chpmv

Matrix-vector product with a packed hermitian matrix and float-complex elements

Function

```
clAmdBlasStatus clAmdBlasChpmv( clAmdBlasOrder order, clAmdBlasUplo uplo, size_t
N, cl_float2 alpha, const cl_mem AP, size_t offa, const cl_mem X, size_t offx,
int incx, cl_float2 beta, cl_mem Y, size_t offy, int incy, cl_uint
numCommandQueues, cl_command_queue *commandQueues, cl_uint numEventsInWaitList,
const cl_event *eventWaitList, cl_event *events);
```

Description

Hermitian packed matrix-vector multiplication.
Matrix-vector products:

- $Y \leftarrow \alpha A x + \beta y$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*N*	Number of rows/columns in matrix *AP*.
in	*alpha*	The factor of matrix *AP*.
in	*AP*	Buffer object storing packed matrix *AP*.
in	*offa*	Offset in number of elements for first element in matrix *AP*.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of *X*. It cannot be zero.
in	*beta*	The factor of vector *Y*.
in/ out	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset of first element of vector *Y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of *Y*. It cannot be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Matrix-vector product with a packed hermitian matrix and float-complex elements (Cont.)

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasNotInitialized` if `clAmdBlasSetup`() was not called;
- clAmdBlasInvalidValue if invalid parameters are passed:
  - *N* is zero, or
  - either *incx* or *incy* is zero, or
  - the matrix sizes or the vector sizes along with the increments lead to accessing outsize of any of the buffers;
- `clAmdBlasInvalidMemObject` if either *AP*, *X*, or *Y* object is invalid, or an image object rather than the buffer one;
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures;
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid;
- `clAmdBlasInvalidContext` if a context a passed command queue belongs to * was released;
- `clAmdBlasInvalidOperation` if kernel compilation relating to a previous call has not completed for any of the target devices;
- `clAmdBlasCompilerNotAvailable` if a compiler is not available;
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

3.19.2 Zhpmv

Matrix-vector product with a packed hermitian matrix and double-complex elements

Function

```
clAmdBlasStatus clAmdBlasZhpmv( clAmdBlasOrder order, clAmdBlasUplo uplo, size_t
N, cl_double2 alpha, const cl_mem AP, size_t offa, const cl_mem X, size_t offx,
int incx, cl_double2 beta, cl_mem Y, size_t offy, int incy, cl_uint
numCommandQueues, cl_command_queue *commandQueues, cl_uint numEventsInWaitList,
const cl_event *eventWaitList, cl_event *events);
```

Description

Matrix-vector product with a packed hermitian matrix and double-complex elements.
Matrix-vector products:

- $y \leftarrow \alpha A x + \beta y$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*N*	Number of rows and columns in matrix *AP*.
in	*alpha*	The factor of matrix *AP*.
in	*AP*	Buffer object storing matrix *AP*.
in	*offa*	Offset in number of elements for first element in matrix *AP*.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of vector *X*. It cannot be zero.
in	*beta*	The factor of vector *Y*.
in/ out	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset of first element of vector *Y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of vector *Y*. It cannot be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- otherwise, the same error codes as the `clAmdBlasChpmv()` function.

3.20 xSPR2 - Symmetric Packed matrix Rank 2

3.20.1 Sspr2

Symmetric rank 2 operation with a general triangular packed-matrix and float elements

Function

```
clAmdBlasStatus clAmdBlasSspr2(clAmdBlasOrder order, clAmdBlasUplo uplo, size_t
N, cl_float alpha, const cl_mem X, size_t offx, int incx, const cl_mem Y,
size_t offy, int incy, cl_mem AP, size_t offa, cl_uint numCommandQueues,
cl_command_queue* commandQueues, cl_uint numEventsInWaitList, const cl_event*
eventWaitList, cl_event* events);
```

Description

Symmetric packed matrix rank 2 update.
Symmetric rank 2 operation:

- $A \leftarrow \alpha\, x\, y^T + \alpha\, y\, x^T + A$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*N*	Number of columns in matrix *AP*.
in	*alpha*	The factor of matrix *AP*.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in/ out	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset of first element of vector *Y* in buffer object.
in	*incy*	Increment for the elements of *Y*. Must not be zero.
in/ out	*AP*	Buffer object storing packed-matrix *AP*.
in	*offa*	Offset of first element of matrix *AP* in buffer object.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Symmetric rank 2 operation with a general triangular packed-matrix and float elements

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasNotInitialized` if `clAmdBlasSetup`() was not called;
- clAmdBlasInvalidValue if invalid parameters are passed:
  - either *N* is zero, or
  - either incx or *incy* is zero
- `clAmdBlasInvalidMemObject` if either *AP*, *X*, or *Y* object is Invalid, or an image object rather than the buffer one;
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures;
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid;
- `clAmdBlasInvalidContext` if a context a passed command queue belongs to was released;
- `clAmdBlasInvalidOperation` if kernel compilation relating to a previous call has not completed for any of the target devices;
- `clAmdBlasCompilerNotAvailable` if a compiler is not available;
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program * executable.

3.20.2 Dspr2

Symmetric rank 2 operation with a general triangular packed-matrix and double elements

Function

```
clAmdBlasStatus clAmdBlasDspr2( clAmdBlasOrder order, clAmdBlasUplo uplo, size_t
N, cl_double alpha, const cl_mem X, size_t offx, int incx, const cl_mem Y,
size_t offy, int incy, cl_mem AP, size_t offa, cl_uint numCommandQueues,
cl_command_queue* commandQueues, cl_uint numEventsInWaitList, const cl_event*
eventWaitList, cl_event* events);
```

Description

Symmetric rank 2 operation with general triangular packed-matrix and double elements.
Symmetric rank 2 operation:

- $A \leftarrow \alpha\ x\ y^T + \alpha\ y\ x^T + A$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*N*	Number of columns in matrix *AP*.
in	*alpha*	The factor of matrix *AP*.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object.
in	*incx*	Increment for the elements of *X*. It cannot be zero.
in	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset of first element of vector *Y* in buffer object.
in	*incy*	Increment for the elements of *Y*. Must not be zero.
in/ out	*AP*	Buffer object storing packed-matrix *AP*.
in	*offa*	Offset of first element of matrix *AP* in buffer object.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- otherwise, the same error codes as the `clAmdBlasSspr2()` function.

3.21 xHPR - Hermitian Packed matrix Rank 1

3.21.1 Chpr

Hermitian rank 1 operation with a general triangular packed-matrix and float-complex elements

Function

```
clAmdBlasStatus clAmdBlasChpr( clAmdBlasOrder order, clAmdBlasUplo uplo, size_t
N, cl_float alpha, const cl_mem X, size_t offx, int incx, cl_mem AP, size_t
offa, cl_uint numCommandQueues, cl_command_queue* commandQueues, cl_uint
numEventsInWaitList, const cl_event* eventWaitList, cl_event* events);
```

Description

Hermitian packed matrix rank 1 update.
Hermitian rank 1 operation:

- $A \leftarrow \alpha\ x\ x^H + A$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*N*	Number of columns in matrix *AP*.
in	*alpha*	The factor of matrix *AP* (a scalar float value).
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of elements for the first element in vector *X*.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in/ out	*AP*	Buffer object storing matrix *AP*.
in	*offa*	Offset in number of elements for the first element in matrix *AP*.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasNotInitialized` if `clAmdBlasSetup`() was not called;
- clAmdBlasInvalidValue if invalid parameters are passed:
  - *N* is zero, or
  - either *incx* is zero
- `clAmdBlasInvalidMemObject` if either *AP* or *X* object is Invalid, or an image object rather than the buffer one;
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures;
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid;
- `clAmdBlasInvalidContext` if a context a passed command queue belongs to was released;
- `clAmdBlasInvalidOperation` if kernel compilation relating to a previous call has not completed for any of the target devices;
- `clAmdBlasCompilerNotAvailable` if a compiler is not available;
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

3.21.2 Zhpr

Hermitian rank 1 operation with a general triangular packed-matrix and double-complex elements

Function

```
clAmdBlasStatus clAmdBlasZhpr( clAmdBlasOrder order, clAmdBlasUplo uplo, size_t
N, cl_double alpha, const cl_mem X, size_t offx, int incx, cl_mem AP, size_t
offa, cl_uint numCommandQueues, cl_command_queue* commandQueues, cl_uint
numEventsInWaitList, const cl_event* eventWaitList, cl_event* events);
```

Description

Hermitian rank 1 operation with a general triangular packed-matrix and double-complex elements.

Hermitian rank 1 operation:

- $A \leftarrow \alpha \times x^{H} + A$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*N*	Number of columns in matrix *AP*.
in	*alpha*	The factor of matrix *AP* (a scalar float value).
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of elements for the first element in vector *X*.
in	*incx*	Increment for the elements of *X*. It cannot be zero.
in/ out	*AP*	Buffer object storing vector *AP*.
in	*offa*	Offset in number of elements for the first element in matrix *AP*.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- otherwise, the same error codes as the `clAmdBlasChpr()` function.

3.22 xGBMV - General Banded Matrix Vector

3.22.1 Sgbmv

Matrix-vector product with a general rectangular banded matrix and float elements

Function

```
clAmdBlasStatus clAmdBlasSgbmv( clAmdBlasOrder order, clAmdBlasTranspose trans,
size_t M, size_t N, size_t KL, size_t KU, cl_float alpha, const cl_mem A,
size_t offa, size_t lda, const cl_mem X, size_t offx, int incx, cl_float beta,
cl_mem Y, size_t offy, int incy, cl_uint numCommandQueues, cl_command_queue
*commandQueues, cl_uint numEventsInWaitList, const cl_event *eventWaitList,
cl_event *events);
```

Description

General banded matrix-vector multiplication.
Matrix-vector products:

- $y \leftarrow \alpha A x + \beta y$
- $y \leftarrow \alpha A^T x + \beta y$

Parameters

in	*order*	Row/column order.
in	*trans*	How matrix *A* is to be transposed
in	*M*	Number of rows in banded matrix *A.*
in	*N*	Number of columns in banded matrix *A.*
in	*KL*	Number of sub-diagonals in banded matrix *A.*
in	*KU*	Number of super-diagonals in banded matrix *A.*
in	*alpha*	The factor of vector *A.*
in	*A*	Buffer object storing banded matrix *A.*
in	*offa*	Offset in number of elements for the first element in banded matrix *A.*
in	*lda*	Leading dimension of banded matrix A. It cannot be less than (*KL* + *KU* + 1).
in	*X*	Buffer object storing banded vector *X.*
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of *X*. Must not be zero
in	*beta*	The factor of the vector *Y.*
in/ out	*Y*	Buffer object storing the vector *Y.*
in	*offy*	Offset of first element of vector *Y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of *Y.* Must not be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Matrix-vector product with a general rectangular banded matrix and float elements (Cont.)

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasNotInitialized` if `clAmdBlasSetup()` was not called;
- clAmdBlasInvalidValue if invalid parameters are passed:
  - either *M* or *N* is zero, or
  - *KL* is greater than *M* - 1, or
  - *KU* is greater than *N* - 1, or
  - either *incx* or *incy* is zero, or
  - any of the leading dimensions is invalid;
  - the matrix size or the vector sizes along with the increments lead to accessing outside of any of the buffers;
- `clAmdBlasInvalidMemObject` if either *A*, *X*, or *Y* object is Invalid, or an image object rather than the buffer one;
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures;
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid;
- `clAmdBlasInvalidContext` if a context a passed command queue belongs to was released;
- `clAmdBlasInvalidOperation` if kernel compilation relating to a previous call has not completed for any of the target devices;
- `clAmdBlasCompilerNotAvailable` if a compiler is not available;
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

3.22.2 Dgbmv

Matrix-vector product with general rectangular banded matrix and double elements

Function

```
clAmdBlasStatus clAmdBlasDgbmv( clAmdBlasOrder order, clAmdBlasTranspose trans,
size_t M, size_t N, size_t KL, size_t KU, cl_double alpha, const cl_mem A,
size_t offa, size_t lda, const cl_mem X, size_t offx, int incx, cl_double beta,
cl_mem Y, size_t offy, int incy, cl_uint numCommandQueues, cl_command_queue
*commandQueues, cl_uint numEventsInWaitList, const cl_event *eventWaitList,
cl_event *events);
```

Description

Matrix-vector with general rectangular banded matrix and double elements.
Matrix-vector products:

- $y \leftarrow \alpha A x + \beta y$
- $y \leftarrow \alpha A^T x + \beta y$

Parameters

in	*order*	Row/column order.
in	*trans*	How matrix *A* is to be transposed.
in	*M*	Number of rows in banded matrix *A.*
in	*N*	Number of columns in banded matrix *A.*
in	*KL*	Number of sub-diagonals in banded matrix *A.*
in	*KU*	Number of super-diagonals in banded matrix *A.*
in	*alpha*	The factor of banded matrix *A.*
in	*A*	Buffer object storing banded matrix *A.*
in	*offa*	Offset in number of elements for first element in banded matrix *A.*
in	*lda*	Leading dimension of banded matrix A. It cannot be less than (*KL* + *KU* + 1).
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in	*beta*	The factor of the vector *Y.*
in/ out	*Y*	Buffer object storing vector *Y.*
in	*offy*	Offset of first element of vector *Y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of *Y.* Must not be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- otherwise, the same error codes as the `clAmdBlasSgbmv`() function.

3.22.3 Cgbmv

Matrix-vector product with a general rectangular banded matrix and float-complex elements

Function

```
clAmdBlasStatus clAmdBlasCgbmv( clAmdBlasOrder order, clAmdBlasTranspose trans,
size_t M, size_t N, size_t KL, size_t KU, cl_float2 alpha, const cl_mem A,
size_t offa, size_t lda, const cl_mem X, size_t offx, int incx, cl_float2 beta,
cl_mem Y, size_t offy, int incy, cl_uint numCommandQueues, cl_command_queue
*commandQueues, cl_uint numEventsInWaitList, const cl_event *eventWaitList,
cl_event *events);
```

Description

Matrix-vector product with a general rectangular banded matrix and float-complex elements. Matrix-vector products:

- $y \leftarrow \alpha A x + \beta y$
- $y \leftarrow \alpha A^T x + \beta y$

Parameters

in	*order*	Row/column order.
in	*trans*	How matrix *A* is to be transposed.
in	*M*	Number of rows in banded matrix *A.*
in	*N*	Number of columns in banded matrix *A.*
in	*KL*	Number of sub-diagonals in banded matrix *A.*
in	*KU*	Number of super-diagonals in banded matrix *A.*
in	*alpha*	The factor of banded matrix *A.*
in	*A*	Buffer object storing banded matrix *A.*
in	*offa*	Offset in number of elements for first element in banded matrix *A.*
in	*lda*	Leading dimension of banded matrix A. It cannot be less than (*KL* + *KU* + 1)
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in	*beta*	The factor of the vector *Y.*
in/ out	*Y*	Buffer object storing the vector *Y.*
in	*offy*	Offset of first element of vector *Y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of *Y.* Must not be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- The same result as the `clAmdBlasSgbmv()` function.

3.22.4 Zgbmv

Matrix-vector product with a general rectangular banded matrix and double-complex elements

Function

```
clAmdBlasStatus clAmdBlasZgbmv( clAmdBlasOrder order, clAmdBlasTranspose trans,
size_t M, size_t N, size_t KL, size_t KU, cl_double2 alpha, const cl_mem A,
size_t offa, size_t lda, const cl_mem X, size_t offx, int incx, cl_double2
beta, cl_mem Y, size_t offy, int incy, cl_uint numCommandQueues,
cl_command_queue *commandQueues, cl_uint numEventsInWaitList, const cl_event
*eventWaitList, cl_event *events);
```

Description

Matrix-vector product with a general rectangular banded matrix and double-complex elements. Matrix-vector products:

- $y \leftarrow \alpha A x + \beta y$
- $y \leftarrow \alpha A^T x + \beta y$

Parameters

in	*order*	Row/column order.
in	*trans*	How matrix *A* is to be transposed.
in	*M*	Number of rows in banded matrix *A*.
in	*N*	Number of columns in banded matrix *A.*
in	*KL*	Number of sub-diagonals in banded matrix *A.*
in	*KU*	Number of super-diagonals in triangular banded matrix *A.*
in	*alpha*	The factor of banded matrix *A.*
in	*A*	Buffer object storing banded matrix *A.*
in	*offa*	Offset in number of elements for first element in banded matrix *A.*
in	*lda*	Leading dimension of banded matrix A. It cannot be less than (*KL* + *KU* + 1)
in	*X*	Buffer object storing vector *X.*
in	*offx*	Offset of first element of vector *X* in buffer object.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in	*beta*	The factor of the vector *Y.*
in/ out	*Y*	Buffer object storing the vector *Y.*
in	*offy*	Offset of first element of vector *Y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of *Y.* Must not be zero
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- The same result as the `clAmdBlasDgbmv()` function.

3.23 xTBMV - Triangle Banded Matrix Vector

3.23.1 Stbmv

Matrix-vector product with a triangular banded matrix and float elements

Function

```
clAmdBlasStatus clAmdBlasStbmv( clAmdBlasOrder order, clAmdBlasUplo uplo,
clAmdBlasTranspose trans, clAmdBlasDiag diag, size_t N, size_t K, const cl_mem
A, size_t offa, size_t lda, cl_mem X, size_t offx, int incx, cl_mem scratchBuff,
cl_uint numCommandQueues, cl_command_queue *commandQueues, cl_uint
numEventsInWaitList, const cl_event *eventWaitList, cl_event *events);
```

Description

Triangular banded matrix vector multiply.
Matrix-vector products:

- $x \leftarrow A\ x$
- $x \leftarrow A^T\ x$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*trans*	How matrix *A* is to be transposed.
in	*diag*	Specify whether matrix *A* is unit triangular.
in	*N*	Number of rows/columns in matrix *A*.
in	*K*	Number of sub-diagonals/super-diagonals in triangular banded matrix *A*.
in	*A*	Buffer object storing matrix.
in	*offa*	Offset in number of elements for first element in matrix *A*.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than (*K* + 1).
in/ out	*X*	Buffer object storing matrix *X*.
in	*offx*	Offset in number of elements for first element in vector *X*.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in	*scratchBuff*	Temporary cl_mem scratch buffer object which can hold a minimum of (1 + (*N*-1)*abs(*incx*)) elements.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Matrix-vector product with a triangular banded matrix and float elements (Cont.)

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasNotInitialized` if `clAmdBlasSetup`() was not called;
- clAmdBlasInvalidValue if invalid parameters are passed:
  - either *N* or *incx* is zero, or
  - *K* is greater than *N* - 1
  - the leading dimension is invalid;
- `clAmdBlasInvalidMemObject` if either *A* or *X* object is Invalid, or an image object rather than the buffer one;
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures;
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid;
- `clAmdBlasInvalidContext` if a context a passed command queue belongs to was released;
- `clAmdBlasInvalidOperation` if kernel compilation relating to a previous call has not completed for any of the target devices;
- `clAmdBlasCompilerNotAvailable` if a compiler is not available;
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

3.23.2 Dtbmv

Matrix-vector product with triangular banded matrix and double elements

Function `cclAmdBlasStatus clAmdBlasDtbmv(clAmdBlasOrder order, clAmdBlasUplo uplo, clAmdBlasTranspose trans, clAmdBlasDiag diag, size_t N, size_t K, const cl_mem A, size_t offa, size_t lda, cl_mem X, size_t offx, int incx, cl_mem scratchBuff, cl_uint numCommandQueues, cl_command_queue *commandQueues, cl_uint numEventsInWaitList, const cl_event *eventWaitList, cl_event *events);`

Description Matrix-vector product with triangular banded matrix and double elements.
Matrix-vector products:

- $x \leftarrow A\ x$
- $x \leftarrow A^T\ x$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*trans*	How matrix *A* is to be transposed.
in	*diag*	Specify whether matrix *A* is unit triangular.
in	*N*	Number of rows/columns in banded matrix *A*.
in	*K*	Number of sub-diagonals/super-diagonals in triangular banded matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offa*	Offset in number of elements for first element in matrix *A*.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than (*K* + 1).
in/ out	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of elements for first element in vector *X*.
in	*incx*	Increment for the elements of *X*. It cannot be zero.
in	*scratchbuff*	Temporary cl_mem scratch buffer object which can hold a minimum of (1 + (*N*-1)*abs(*incx*)) elements.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- otherwise, the same error codes as the `clAmdBlasStbmv`() function.

3.23.3 Ctbmv

Matrix-vector product with triangular banded matrix and float-complex elements

Function

```
clAmdBlasStatus clAmdBlasCtbmv( clAmdBlasOrder order, clAmdBlasUplo uplo,
clAmdBlasTranspose trans, clAmdBlasDiag diag, size_t N, size_t K, const cl_mem
A, size_t offa, size_t lda, cl_mem X, size_t offx, int incx, cl_mem scratchBuff,
cl_uint numCommandQueues, cl_command_queue *commandQueues, cl_uint
numEventsInWaitList, const cl_event *eventWaitList, cl_event *events);
```

Description

Matrix-vector product with triangle banded matrix and float-complex elements.
Matrix-vector products:

- $x \leftarrow A\ x$
- $x \leftarrow A^T\ x$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*trans*	How matrix *A* is to be transposed.
in	*diag*	Specify whether matrix *A* is unit triangular.
in	*N*	Number of rows/columns in matrix *A.*
in	*K*	Number of sub-diagonals/super-diagonals in triangular banded matrix *A.*
in	*A*	Buffer object storing matrix *A.*
in	*offa*	Offset in number of elements for first element in matrix *A.*
in	*lda*	Leading dimension of matrix *A.* It cannot be less than (*K* + 1)
in/ out	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of elements for first element in vector *X.*
in	*incx*	Increment for the elements of *X*. Must not be zero.
in	*scratchBuff*	Temporary cl_mem scratch buffer object which can hold a minimum of (1 + (*N*-1)*abs(*incx*)) elements.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- The same result as the `clAmdBlasStbmv()` function.

3.23.4 Ztbmv

Matrix-vector product with triangular banded matrix and double-complex elements

Function

```
clAmdBlasStatus clAmdBlasZtbmv( clAmdBlasOrder order, clAmdBlasUplo uplo,
clAmdBlasTranspose trans, clAmdBlasDiag diag, size_t N, size_t K, const cl_mem
A, size_t offa, size_t lda, cl_mem X, size_t offx, int incx, cl_mem scratchBuff,
cl_uint numCommandQueues, cl_command_queue *commandQueues, cl_uint
numEventsInWaitList, const cl_event *eventWaitList, cl_event *events);
```

Description

Matrix-vector product with triangle banded matrix and float-complex elements.
Matrix-vector products:

- $x \leftarrow A\ x$
- $x \leftarrow A^T\ x$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*trans*	How matrix *A* is to be transposed.
in	*diag*	Specify whether matrix *A* is unit triangular.
in	*N*	Number of rows/columns in matrix *A*.
in	*K*	Number of sub-diagonals/super-diagonals in triangular banded matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offa*	Offset in number of elements for first element in matrix *A*.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than (*K* + 1)
in/ out	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of elements for first element in vector *X*.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in	*scratchBuff*	Temporary cl_mem scratch buffer object which can hold a minimum of (1 + (*N*-1)*abs(*incx*)) elements.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- The same result as the `clAmdBlasDtbmv()` function.

3.24 xHPR2 - Hermitian Packed matrix Rank 2

3.24.1 Chpr2

Hermitian rank 2 operation with a general triangular packed-matrix and float-complex elements

Function

```
clAmdBlasStatus clAmdBlasChpr2(clAmdBlasOrder order, clAmdBlasUplo uplo, size_t
N, cl_float2 alpha, const cl_mem X, size_t offx, int incx, const cl_mem Y,
size_t offy, int incy, cl_mem AP, size_t offa, cl_uint numCommandQueues,
cl_command_queue* commandQueues, cl_uint numEventsInWaitList, const cl_event*
eventWaitList, cl_event* events);
```

Description

Hermitian packed matrix rank 2 update.
Hermitian rank 2 operation:

- $A \leftarrow \alpha\, x\, y^H + \overline{\alpha}\, y\, x^H + A$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*N*	Number of columns in matrix *AP*.
in	*alpha*	The factor of matrix *AP*.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of elements for first element in vector *X*.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset in number of elements for the first element in vector *Y*.
in	*incy*	Increment for the elements of *Y*. Must not be zero.
in/ out	*AP*	Buffer object storing packed-matrix *AP*.
in	*offa*	Offset in number of elements for the first element in matrix *AP*.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Hermitian rank 2 operation with a general triangular packed-matrix and float-complex elements (Cont.)

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasNotInitialized` if `clAmdBlasSetup`() was not called;
- clAmdBlasInvalidValue if invalid parameters are passed:
  - either *N* or *incx* is zero, or
  - *K* is greater than *N* - 1
  - the leading dimension is invalid;
- `clAmdBlasInvalidMemObject` if either *A* or *X* object is Invalid, or an image object rather than the buffer one;
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures;
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid;
- `clAmdBlasInvalidContext` if a context a passed command queue belongs to was released;
- `clAmdBlasInvalidOperation` if kernel compilation relating to a previous call has not completed for any of the target devices;
- `clAmdBlasCompilerNotAvailable` if a compiler is not available;
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

3.24.2 Zhpr2

Hermitian rank 2 operation with general triangular packed-matrix and double-complex elements

Function

```
clAmdBlasStatus clAmdBlasZhpr2( clAmdBlasOrder order, clAmdBlasUplo uplo, size_t
N, cl_double2 alpha, const cl_mem X, size_t offx, int incx, const cl_mem Y,
size_t offy, int incy, cl_mem AP, size_t offa, cl_uint numCommandQueues,
cl_command_queue* commandQueues, cl_uint numEventsInWaitList, const cl_event*
eventWaitList, cl_event* events);
```

Description

Hermitian rank 2 operation with general triangular packed-matrix and double-complex elements. Hermitian rank 2 operation:

- $A \leftarrow \alpha\ X\ Y^H + \overline{\alpha}\ Y\ X^H + A$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*N*	Number of columns in matrix *AP*.
in	*alpha*	The factor of matrix *AP*.
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of elements for first element in vector *X*.
in	*incx*	Increment for the elements of *X*. It cannot be zero.
in	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset in number of elements for first element in vector *Y*.
in	*incy*	Increment for the elements of *Y*. Must not be zero.
in/ out	*AP*	Buffer object storing packed-matrix *AP*.
in	*offa*	Offset in number of elements for the first element in matrix *AP*.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- otherwise, the same error codes as the `clAmdBlasChpr2`() function.

3.25 xSBMV - Symmetric Banded Matrix Vector

3.25.1 Ssbmv

Matrix-vector product with a symmetric banded matrix and float elements

Function

```
clAmdBlasStatus clAmdBlasSsbmv( clAmdBlasOrder order, clAmdBlasUplo uplo, size_t
N, size_t K, cl_float alpha, const cl_mem A, size_t offa, size_t lda, const
cl_mem X, size_t offx, int incx, cl_float beta, cl_mem Y, size_t offy, int
incy, cl_uint numCommandQueues, cl_command_queue *commandQueues, cl_uint
numEventsInWaitList, const cl_event *eventWaitList, cl_event *events);
```

Description

Symmetric banded matrix-vector multiplication.
Matrix-vector products:

- $y \leftarrow \alpha A x + \beta y$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*N*	Number of rows/columns in matrix *A*.
in	*K*	Number of sub-diagonals/super-diagonals in banded matrix *A*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offa*	Offset in number of elements for first element in matrix *A*.
in	*lda*	Leading dimension of matrix A. It cannot be less than (K + 1).
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in	*beta*	The factor of vector *Y*.
in/ out	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset of first element of vector *Y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of vector *Y*. It cannot be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Matrix-vector product with a symmetric banded matrix and float elements (Cont.)

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasNotInitialized` if `clAmdBlasSetup`() was not called;
- clAmdBlasInvalidValue if invalid parameters are passed:
  - either *N* or incx is zero, or
  - *K* is greater than *N* - 1
  - the leading dimension is invalid;
- `clAmdBlasInvalidMemObject` if either *A* or *X* object is Invalid, or an image object rather than the buffer one;
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures;
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid;
- `clAmdBlasInvalidContext` if a context a passed command queue belongs to was released;
- `clAmdBlasInvalidOperation` if kernel compilation relating to a previous call has not completed for any of the target devices;
- `clAmdBlasCompilerNotAvailable` if a compiler is not available;
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

3.25.2 Dsbmv

Matrix-vector product with symmetric banded matrix and double elements

Function

```
clAmdBlasStatus clAmdBlasDsbmv( clAmdBlasOrder order, clAmdBlasUplo uplo, size_t
N, size_t K, cl_double alpha, const cl_mem A, size_t offa, size_t lda, const
cl_mem X, size_t offx, int incx, cl_double beta, cl_mem Y, size_t offy, int
incy, cl_uint numCommandQueues, cl_command_queue *commandQueues, cl_uint
numEventsInWaitList, const cl_event *eventWaitList, cl_event *events);
```

Description

Matrix-vector product with a symmetric banded matrix and double elements.
Matrix-vector products:

- $y \leftarrow \alpha A x + \beta y$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*N*	Number of rows and columns in matrix *A*.
in	*K*	Number of sub-diagonals/super-diagonals in banded matrix *A*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offa*	Offset in number of elements for first element in matrix *A*.
in	*lda*	Leading dimension of matrix *A*. It cannot be less (*K* + 1).
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of vector *X*. It cannot be zero.
in	*beta*	The factor of vector *Y*.
in/ out	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset of first element of vector *Y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of vector *Y*. It cannot be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- otherwise, the same error codes as the `clAmdBlasSsbmv`() function.

3.25.3 Chbmv

Hermitian banded matrix-vector multiplication

Function

```
clAmdBlasStatus clAmdBlasChbmv( clAmdBlasOrder order, clAmdBlasUplo uplo, size_t
N, size_t K, cl_float2 alpha, const cl_mem A, size_t offa, size_t lda, const
cl_mem X, size_t offx, int incx, cl_float2 beta, cl_mem Y, size_t offy, int
incy, cl_uint numCommandQueues, cl_command_queue *commandQueues, cl_uint
numEventsInWaitList, const cl_event *eventWaitList, cl_event *events);
```

Description

Hermitian banded matrix-vector multiplication.
Matrix-vector products:

- $y \leftarrow \alpha A x + \beta y$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*N*	Number of rows/columns in banded matrix *A*.
in	*K*	Number of sub-diagonals/super-diagonals in triangular banded matrix *A*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offa*	Offset in number of elements for first element in matrix *A*.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than (*K* + 1).
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements
in	*incx*	Increment for the elements of *X*. Must not be zero.
in	*beta*	The factor of vector *Y*.
out	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset of first element of vector *Y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of vector *Y*. It cannot be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Hermitian banded matrix-vector multiplication (Cont.)

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasNotInitialized` if clAmdBlasSetup() was not called;
- clAmdBlasInvalidValue if invalid parameters are passed:
  - either *N* or *incx* is zero, or
  - *K* is greater than *N* - 1
  - the leading dimension is invalid;
- `clAmdBlasInvalidMemObject` if either *A* or *X* object is Invalid, or an image object rather than the buffer one;
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures;
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid;
- `clAmdBlasInvalidContext` if a context a passed command queue belongs to was released;
- `clAmdBlasInvalidOperation` if kernel compilation relating to a previous call has not completed for any of the target devices;
- `clAmdBlasCompilerNotAvailable` if a compiler is not available;
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

3.25.4 Zhbmv

Matrix-vector product with a packed hermitian matrix and double elements

Function

```
clAmdBlasStatus clAmdBlasZhbmv( clAmdBlasOrder order, clAmdBlasUplo uplo, size_t
N, size_t K, cl_double2 alpha, const cl_mem A, size_t offa, size_t lda, const
cl_mem X, size_t offx, int incx, cl_double2 beta, cl_mem Y, size_t offy, int
incy, cl_uint numCommandQueues, cl_command_queue *commandQueues, cl_uint
numEventsInWaitList, const cl_event *eventWaitList, cl_event *events);
```

Description

Matrix-vector product with a packed hermitian matrix and double elements.
Matrix-vector products:

- $y \leftarrow \alpha A x + \beta y$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*N*	Number of rows and columns in matrix *A*.
in	*K*	Number of sub-diagonals/super-diagonals in banded matrix *A*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offa*	Offset in number of elements for first element in matrix *A*.
in	*lda*	Leading dimension of matrix A. It cannot be less than (*K* + 1).
in	*X*	Buffer object storing vector *X*.
in	*offx*	Offset of first element of vector *X* in buffer object. Counted in elements.
in	*incx*	Increment for the elements of vector *X*. It cannot be zero.
in	*beta*	The factor of vector *Y*.
in/ out	*Y*	Buffer object storing vector *Y*.
in	*offy*	Offset of first element of vector *Y* in buffer object. Counted in elements.
in	*incy*	Increment for the elements of vector *Y*. It cannot be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- otherwise, the same error codes as the `clAmdBlasChbmv`() function.

3.26 xTBSV - Solving Triangular Banded matrix Vectors

3.26.1 Stbsv

Solving triangular banded matrix problems with float elements

Function

```
clAmdBlasStatus clAmdBlasStbsv( clAmdBlasOrder order, clAmdBlasUplo uplo,
clAmdBlasTranspose trans, clAmdBlasDiag diag, size_t N, size_t K, const cl_mem
A, size_t offa, size_t lda, cl_mem X, size_t offx, int incx, cl_uint
numCommandQueues, cl_command_queue *commandQueues, cl_uint numEventsInWaitList,
const cl_event *eventWaitList, cl_event *events);
```

Description

Solving triangular banded matrix problems.
Matrix-vector products:

- $A\ x \leftarrow x$
- $A^T\ x \leftarrow x$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*trans*	How matrix *A* is to be transposed.
in	*diag*	Specify whether matrix *A* is unit triangular.
in	*N*	Number of rows/columns in banded matrix *A*.
in	*K*	Number of sub-diagonals/super-diagonals in triangular banded matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offa*	Offset in number of elements for first element in matrix *A*.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than (K + 1).
in/ out	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of elements for first element in vector *X*.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Solving triangular banded matrix problems with float elements (Cont.)

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasNotInitialized` if clAmdBlasSetup() was not called;
- clAmdBlasInvalidValue if invalid parameters are passed:
  - either *N* or *incx* is zero, or
  - *K* is greater than *N* - 1
  - the leading dimension is invalid;
- `clAmdBlasInvalidMemObject` if either *A* or *X* object is Invalid, or an image object rather than the buffer one;
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures;
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid;
- `clAmdBlasInvalidContext` if a context a passed command queue belongs to was released;
- `clAmdBlasInvalidOperation` if kernel compilation relating to a previous call has not completed for any of the target devices;
- `clAmdBlasCompilerNotAvailable` if a compiler is not available;
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

3.26.2 Dtbsv

Solving triangular banded matrix problems with double elements

Function

```
clAmdBlasStatus clAmdBlasDtbsv( clAmdBlasOrder order, clAmdBlasUplo uplo,
clAmdBlasTranspose trans, clAmdBlasDiag diag, size_t N, size_t K, const cl_mem
A, size_t offa, size_t lda, cl_mem X, size_t offx, int incx, cl_uint
numCommandQueues, cl_command_queue *commandQueues, cl_uint numEventsInWaitList,
const cl_event *eventWaitList, cl_event *events);
```

Description

Solving triangular banded matrix problems with double elements.
Matrix-vector products:

- $A\ x \leftarrow x$
- $A^T\ x \leftarrow x$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*trans*	How matrix *A* is to be transposed.
in	*diag*	Specify whether matrix *A* is unit triangular.
in	*N*	Number of rows/columns in banded matrix *A*.
in	*K*	Number of sub-diagonals/super-diagonals in triangular banded matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offa*	Offset in number of elements for first element in matrix *A*.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than (*K* + 1).
in/ out	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of elements for first element in vector *X*.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- otherwise, the same error codes as the `clAmdBlasStbsv()` function.

3.26.3 Ctbsv

Solving triangular banded matrix problems with float-complex elements

Function

```
clAmdBlasStatus clAmdBlasCtbsv( clAmdBlasOrder order, clAmdBlasUplo uplo,
clAmdBlasTranspose trans, clAmdBlasDiag diag, size_t N, size_t K, const cl_mem
A, size_t offa, size_t lda, cl_mem X, size_t offx, int incx, cl_uint
numCommandQueues, cl_command_queue *commandQueues, cl_uint numEventsInWaitList,
const cl_event *eventWaitList, cl_event *events);
```

Description

Solving triangular banded matrix problems with float-complex elements.
Matrix-vector products:

- $A\ x \leftarrow x$
- $A^T\ x \leftarrow x$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*trans*	How matrix *A* is to be transposed.
in	*diag*	Specify whether matrix *A* is unit triangular
in	*N*	Number of rows/columns in banded matrix *A.*
in	*K*	Number of sub-diagonals/super-diagonals in triangular banded matrix *A.*
in	*A*	Buffer object storing matrix *A.*
in	*offa*	Offset in number of elements for first element in matrix *A.*
in	*lda*	Leading dimension of matrix A. It cannot be less than (*K* + 1).
in/ out	*X*	Buffer object storing vector *X*.
in	*offx*	Offset in number of elements for first element in vector *X*.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- The same result as the `clAmdBlasStbsv()` function.

3.26.4 Ztbsv

Solving triangular banded matrix problems with double-complex elements

Function

```
clAmdBlasStatus clAmdBlasZtbsv( clAmdBlasOrder order, clAmdBlasUplo uplo,
clAmdBlasTranspose trans, clAmdBlasDiag diag, size_t N, size_t K, const cl_mem
A, size_t offa, size_t lda, cl_mem X, size_t offx, int incx, cl_uint
numCommandQueues, cl_command_queue *commandQueues, cl_uint numEventsInWaitList,
const cl_event *eventWaitList, cl_event *events);
```

Description

Solving triangular banded matrix problems with double-complex elements.
Matrix-vector products:

- $A\ x \leftarrow x$
- $A^T\ x \leftarrow x$

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix being referenced.
in	*trans*	How matrix *A* is to be transposed.
in	*diag*	Specify whether matrix *A* is unit triangular
in	*N*	Number of rows/columns in banded matrix *A.*
in	*K*	Number of sub-diagonals/super-diagonals in triangular banded matrix *A.*
in	*A*	Buffer object storing matrix *A.*
in	*offa*	Offset in number of elements for first element in matrix *A.*
in	*lda*	Leading dimension of matrix A. It cannot be less than (*K* + 1)
in/ out	*X*	Buffer object storing vector *X.*
in	*offx*	Offset in number of elements for first element in vector *X*.
in	*incx*	Increment for the elements of *X*. Must not be zero.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- The same result as the `clAmdBlasDtbsv()` function.

Chapter 4
BLAS-3 Functions

4.1 xGEMM - GEneral Matrix-matrix Multiplication

4.1.1 Sgemm

Matrix-matrix product of general rectangular matrices with float elements

Function

`cl_int clAmdBlasSgemm (clAmdBlasOrder` *`order`*`, clAmdBlasTranspose` *`transA`*`, clAmdBlasTranspose` *`transB`*`, size_t` *`M`*`, size_t` *`N`*`, size_t` *`K`*`, cl_float` *`alpha`*`, const cl_mem` *`A`*`, size_t` *`lda`*`, const cl_mem` *`B`*`, size_t` *`ldb`*`, cl_float` *`beta`*`, cl_mem` *`C`*`, size_t` *`ldc`*`, cl_uint` *`numCommandQueues`*`, cl_command_queue * ` *`commandQueues`*`, cl_uint` *`numEventsInWaitList`*`, const cl_event * ` *`eventWaitList`*`, cl_event * ` *`events`*`)`

Description

This function has been deprecated.

Matrix-matrix product of general rectangular matrices with float-complex elements. Matrix-matrix products are:

- $C \leftarrow \alpha AB + \beta C$
- $C \leftarrow \alpha AB^T + \beta C$
- $C \leftarrow \alpha A^T B + \beta C$
- $C \leftarrow \alpha A^T B^T + \beta C$

Matrix-matrix product of general rectangular matrices with float elements (Cont.)

Parameters

in	*order*	Row/column order.
in	*transA*	How matrix *A* is to be transposed.
in	*transB*	How matrix *B* is to be transposed.
in	*M*	Number of rows in matrix *A*.
in	*N*	Number of columns in matrix *B*.
in	*K*	Number of columns in matrix *A* and rows in matrix *B*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *K* when the *order* parameter is set to `clAmdBlasRowMajor`, or less than *M* when the parameter is set to `clAmdBlasColumnMajor`.
in	*B*	Buffer object storing matrix *B*.
in	*ldb*	Leading dimension of matrix *B*. It cannot be less than *N* when the *order* parameter is set to `clAmdBlasRowMajor`, or less than *K* when it is set to `clAmdBlasColumnMajor`.
in	*beta*	The factor of matrix *C*.
in/ out	*C*	Buffer object storing matrix *C*.
in	*ldc*	Leading dimension of matrix *C*. Leading dimension of matrix *C*. It cannot be less than *N* when the *order* parameter is set to `clAmdBlasRowMajor`, or less than *M* when it is set to `clAmdBlasColumnMajor`.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task. Currently, only one command queue is supported.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

0 on success; otherwise, following error codes.

- `CL_INVALID_VALUE` if invalid parameters are passed: A, B, or C object is invalid, or an image object rather than the buffer one; M, N, or K is zero, or a leading dimension is invalid, or the `numCommandQueues` parameter is not equal to 1.
- `CL_OUT_OF_HOST_MEMORY` if the library cannot allocate memory for internal structures.
- An error code from `clEnqueueNDRangeKernel()` function call.

4.1.2 Dgemm

Matrix-matrix product of general rectangular matrices with double elements

Function

`cl_int clAmdBlasDgemm (clAmdBlasOrder order, clAmdBlasTranspose transA, clAmdBlasTranspose transB, size_t M, size_t N, size_t K, cl_double alpha, const cl_mem A, size_t lda, const cl_mem B, size_t ldb, cl_double beta, cl_mem C, size_t ldc, cl_uint numCommandQueues, cl_command_queue * commandQueues, cl_uint numEventsInWaitList, const cl_event * eventWaitList, cl_event * events)`

Description

This function has been deprecated. Matrix-matrix product of general rectangular matrices with double elements.

Matrix-matrix products are:

- $C \leftarrow \alpha AB + \beta C$
- $C \leftarrow \alpha AB^T + \beta C$
- $C \leftarrow \alpha A^T B + \beta C$
- $C \leftarrow \alpha A^T B^T + \beta C$

Parameters

in	*order*	Row/column order.
in	*transA*	How matrix *A* is to be transposed.
in	*transB*	How matrix *B* is to be transposed.
in	*M*	Number of rows in matrix *A*.
in	*N*	Number of columns in matrix *B*.
in	*K*	Number of columns in matrix *A* and rows in matrix *B*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*lda*	Leading dimension of matrix *A*. For a detailed description see `clAmdBlasSgemm()`.
in	*B*	Buffer object storing matrix *B*.
in	*ldb*	Leading dimension of matrix *B*. See `clAmdBlasSgemm()`.
in	*beta*	The factor of matrix *C*.
in/ out	*C*	Buffer object storing matrix *C*.
in	*ldc*	Leading dimension of matrix *C*. See `clAmdBlasSgemm()`.
in	*numCommandQueues*	Number of OpenCL command queues in which the task should be performed. Currently, only one command queue is supported.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects in each command queue that identify a particular kernel execution instance.

Returns

0 on success; otherwise, following error codes.

- `CL_INVALID_VALUE` if invalid parameters are passed: A, B, or C object is invalid, or an image object rather than the buffer one; M, N, or K is zero, or a leading dimension is invalid, or the numCommandQueues parameter is not equal to 1.
- `CL_OUT_OF_HOST_MEMORY` if the library cannot allocate memory for internal structures.
- An error code from `clEnqueueNDRangeKernel()` function call.

4.1.3 Cgemm

Matrix-matrix product of general rectangular matrices with float complex elements

Function

`clAmdBlasStatus clAmdBlasCgemm (clAmdBlasOrder` *`order`*`, clAmdBlasTranspose` *`transA`*`, clAmdBlasTranspose` *`transB`*`, size_t` *`M`*`, size_t` *`N`*`, size_t` *`K`*`, FloatComplex` *`alpha`*`, const cl_mem` *`A`*`, size_t` *`lda`*`, const cl_mem` *`B`*`, size_t` *`ldb`*`, FloatComplex` *`beta`*`, cl_mem` *`C`*`, size_t` *`ldc`*`, cl_uint` *`numCommandQueues`*`, cl_command_queue *` *`commandQueues`*`, cl_uint` *`numEventsInWaitList`*`, const cl_event *` *`eventWaitList`*`, cl_event *` *`events`*`)`

Description

This function has been deprecated.
Matrix-matrix product of general rectangular matrices with float elements.
Matrix-matrix products are:

- $C \leftarrow \alpha AB + \beta C$
- $C \leftarrow \alpha AB^T + \beta C$
- $C \leftarrow \alpha A^T B + \beta C$
- $C \leftarrow \alpha A^T B^T + \beta C$

Parameters

in	*order*	Row/column order.
in	*transA*	How matrix *A* is to be transposed.
in	*transB*	How matrix *B* is to be transposed.
in	*M*	Number of rows in matrix *A*.
in	*N*	Number of columns in matrix *B*.
in	*K*	Number of columns in matrix *A* and rows in matrix *B*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *K* when the *order* parameter is set to `clAmdBlasRowMajor`, or less than *M* when the parameter is set to `clAmdBlasColumnMajor`.
in	*B*	Buffer object storing matrix *B*.
in	*ldb*	Leading dimension of matrix *B*. It cannot be less than *N* when the *order* parameter is set to `clAmdBlasRowMajor`, or less than *K* when it is set to `clAmdBlasColumnMajor`.
in	*beta*	The factor of matrix *C*.
in/ out	*C*	Buffer object storing matrix *C*.
in	*ldc*	Leading dimension of matrix *C*. It cannot be less than *N* when the *order* parameter is set to `clAmdBlasRowMajor`, or less than *M* when it is set to `clAmdBlasColumnMajor`.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task. Currently, only one command queue is supported.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects in each command queue that identify a particular kernel execution instance.

Matrix-matrix product of general rectangular matrices with float complex elements (Cont.)

Returns 0 on success; otherwise, following error codes.

- `CL_INVALID_VALUE` if invalid parameters are passed: A, B, or C object is invalid, or an image object rather than the buffer one; M, N, or K is zero, or a leading dimension is invalid, or the `numCommandQueues` parameter is not equal to 1.
- `CL_OUT_OF_HOST_MEMORY` if the library cannot allocate memory for internal structures.
- An error code from `clEnqueueNDRangeKernel()` function call.

Examples `example_sgemm.c`.

4.1.4 Zgemm

Matrix-matrix product of general rectangular matrices with double complex elements

Function

`cl_int clAmdBlasZgemm (clAmdBlasOrder` *`order`*`, clAmdBlasTranspose` *`transA`*`, clAmdBlasTranspose` *`transB`*`, size_t` *`M`*`, size_t` *`N`*`, size_t` *`K`*`, DoubleComplex` *`alpha`*`, const cl_mem` *`A`*`, size_t` *`lda`*`, const cl_mem` *`B`*`, size_t` *`ldb`*`, DoubleComplexbeta, cl_mem` *`C`*`, size_t` *`ldc`*`, cl_uint` *`numCommandQueues`*`, cl_command_queue *` *`commandQueues`*`, cl_uint` *`numEventsInWaitList`*`, const cl_event *` *`eventWaitList`*`,cl_event *` *`events`*`)`

Description

This function has been deprecated.
Matrix-matrix product of general rectangular matrices with double-complex elements.
Matrix-matrix products are:

- $C \leftarrow \alpha AB + \beta C$
- $C \leftarrow \alpha AB^T + \beta C$
- $C \leftarrow \alpha A^TB + \beta C$
- $C \leftarrow \alpha A^TB^T + \beta C$

Parameters

in	*order*	Row/column order.
in	*transA*	How matrix *A* is to be transposed.
in	*transB*	How matrix *B* is to be transposed.
in	*M*	Number of rows in matrix *A*.
in	*N*	Number of columns in matrix *B*.
in	*K*	Number of columns in matrix *A* and rows in matrix *B*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*lda*	Leading dimension of matrix *A*. See `clAmdBlasSgemm()`.
in	*B*	Buffer object storing matrix *B*.
in	*ldb*	Leading dimension of matrix *B*. See `clAmdBlasSgemm()`.
in	*beta*	The factor of matrix *C*.
in/ out	*C*	Buffer object storing matrix *C*.
in	*ldc*	Leading dimension of matrix *C*. See `clAmdBlasSgemm()`.
in	*numCommandQueues*	Number of OpenCL command queues in which the task should be performed. Currently, only one command queue is supported.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects in each command queue that identify a particular kernel execution instance.

Returns

0 on success; otherwise, following error codes.

- `CL_INVALID_VALUE` if invalid parameters are passed: *A*, *B*, or *C* object is invalid, or an image object rather than the buffer one; *M*, *N*, or *K* is zero, or a leading dimension is invalid, or the `numCommandQueues` parameter is not equal to 1.
- `CL_OUT_OF_HOST_MEMORY` if the library cannot allocate memory for internal structures.
- An error code from `clEnqueueNDRangeKernel()` function call.

4.2 xGEMMEX - GEneral Matrix-matrix Multiplication, Extended

4.2.1 SgemmEx

Matrix-matrix product of general rectangular matrices with float elements

Function

```
clAmdBlasStatus clAmdBlasSgemmEx ( clAmdBlasOrder order, clAmdBlasTranspose
transA, clAmdBlasTranspose transB, size_t M, size_t N, size_t K, cl_float
alpha, const cl_mem A, size_t offA, size_t lda, const cl_mem B, size_t offB,
size_t ldb, cl_float beta, cl_mem C, size_t offC, size_t ldc, cl_uint
numCommandQueues, cl_command_queue * commandQueues, cl_uint
numEventsInWaitList, const cl_event * eventWaitList, cl_event * events )
```

Description

Matrix-matrix product of general rectangular matrices with float elements. Extended version, which takes an offset value for all matrix arguments. Matrix-matrix products:

- $C \leftarrow \alpha AB + \beta C$
- $C \leftarrow \alpha AB^T + \beta C$
- $C \leftarrow \alpha A^T B + \beta C$
- $C \leftarrow \alpha A^T B^T + \beta C$

Parameters

in	*order*	Row/column order.
in	*transA*	How matrix *A* is to be transposed.
in	*transB*	How matrix *B* is to be transposed.
in	*M*	Number of rows in matrix *A*.
in	*N*	Number of columns in matrix *B*.
in	*K*	Number of columns in matrix *A* and rows in matrix *B*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offA*	Offset of the first element of the matrix *A* in the buffer object. Counted in elements.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *K* when the *order* parameter is set to `clAmdBlasRowMajor`, or less than *M* when the parameter is set to `clAmdBlasColumnMajor`.
in	*B*	Buffer object storing matrix *B*.
in	*offB*	Offset of the first element of the matrix *B* in the buffer object. Counted in elements.
in	*ldb*	Leading dimension of matrix *B*. It cannot be less than *N* when the *order* parameter is set to `clAmdBlasRowMajor`, or less than *K* when it is set to `clAmdBlasColumnMajor`.
in	*beta*	The factor of matrix *C*.
in/ out	*C*	Buffer object storing matrix *C*.
in	*offC*	Offset of the first element of the matrix *C* in the buffer object. Counted in elements.
in	*ldc*	Leading dimension of matrix *C*. It cannot be less than *N* when the *order* parameter is set to `clAmdBlasRowMajor`, or less than *M* when it is set to `clAmdBlasColumnMajor`.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task. Currently, only one command queue is supported.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Matrix-matrix product of general rectangular matrices with float elements (Cont.)

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidValue` if either *offA*, *offB* or *offC* exceeds the size of the respective buffer object;
- otherwise, the same error codes as `clAmdBlasSgemm()`.

4.2.2 DgemmEx

Matrix-matrix product of general rectangular matrices with double elements

Function

```
clAmdBlasStatus clAmdBlasDgemmEx ( clAmdBlasOrder order, clAmdBlasTranspose
transA, clAmdBlasTranspose transB, size_t M, size_t N, size_t K, cl_double
alpha, const cl_mem A, size_t offA, size_t lda, const cl_mem B, size_t offB,
size_t ldb, cl_double beta, cl_mem C, size_t offC, size_t ldc, cl_uint
numCommandQueues, cl_command_queue * commandQueues, cl_uint
numEventsInWaitList, const cl_event * eventWaitList, cl_event * events )
```

Description

Matrix-matrix product of general rectangular matrices with double elements. Extended version, which takes an offset value for all matrix arguments.

Matrix-matrix products:

- $C \leftarrow \alpha AB + \beta C$
- $C \leftarrow \alpha AB^T + \beta C$
- $C \leftarrow \alpha A^T B + \beta C$
- $C \leftarrow \alpha A^T B^T + \beta C$

Parameters

in	*order*	Row/column order.
in	*transA*	How matrix *A* is to be transposed.
in	*transB*	How matrix *B* is to be transposed.
in	*M*	Number of rows in matrix *A*.
in	*N*	Number of columns in matrix *B*.
in	*K*	Number of columns in matrix *A* and rows in matrix *B*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offA*	Offset of the first element of the matrix *A* in the buffer object. Counted in elements.
in	*lda*	Leading dimension of matrix *A*. For a detailed description see `clAmdBlasSgemm()`.
in	*B*	Buffer object storing matrix *B*.
in	*offB*	Offset of the first element of the matrix *B* in the buffer object. Counted in elements.
in	*ldb*	Leading dimension of matrix *B*. For detailed description, see `clAmdBlasSgemm()`.
in	*beta*	The factor of matrix *C*.
in/ out	*C*	Buffer object storing matrix *C*.
in	*offC*	Offset of the first element of the matrix *C* in the buffer object. Counted in elements.
in	*ldc*	Leading dimension of matrix *C*. For detailed description, see `clAmdBlasSgemm()`.
in	*numCommandQueues*	Number of OpenCL command queues in which the task should be performed. Currently, only one command queue is supported.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Matrix-matrix product of general rectangular matrices with double elements (Cont.)

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- `clAmdBlasInvalidValue` if either *offA*, *offB* or *offC* exceeds the size of the respective buffer object;
- otherwise, the same error codes as the `clAmdBlasSgemm()` function.

4.2.3 CgemmEx

Matrix-matrix product of general rectangular matrices with float-complex elements

Function

```
clAmdBlasStatus clAmdBlasCgemmEx ( clAmdBlasOrder order, clAmdBlasTranspose
transA, clAmdBlasTranspose transB, size_t M, size_t N, size_t K, FloatComplex
alpha, const cl_mem A, size_t offA, size_t lda, const cl_mem B, size_t offB,
size_t ldb, FloatComplex beta, cl_mem C, size_t offC, size_t ldc, cl_uint
numCommandQueues, cl_command_queue * commandQueues, cl_uint
numEventsInWaitList, const cl_event * eventWaitList, cl_event * events )
```

Description

Matrix-matrix product of general rectangular matrices with float-complex elements. Extended version, which takes an offset value for all matrix arguments.
Matrix-matrix products are:

- $C \leftarrow \alpha AB + \beta C$
- $C \leftarrow \alpha AB^T + \beta C$
- $C \leftarrow \alpha A^T B + \beta C$
- $C \leftarrow \alpha A^T B^T + \beta C$

Parameters

in	*order*	Row/column order.
in	*transA*	How matrix *A* is to be transposed.
in	*transB*	How matrix *B* is to be transposed.
in	*M*	Number of rows in matrix *A*.
in	*N*	Number of columns in matrix *B*.
in	*K*	Number of columns in matrix *A* and rows in matrix *B*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offA*	Offset of the first element of the matrix *A* in the buffer object. Counted in elements.
in	*lda*	Leading dimension of matrix *A*. See `clAmdBlasSgemm()`.
in	*B*	Buffer object storing matrix *B*.
in	*offB*	Offset of the first element of the matrix *B* in the buffer object. Counted in elements.
in	*ldb*	Leading dimension of matrix *B*. See `clAmdBlasSgemm()`.
in	*beta*	The factor of matrix *C*.
in/ out	*C*	Buffer object storing matrix *C*.
in	*offC*	Offset of the first element of the matrix *C* in the buffer object. Counted in elements.
in	*ldc*	Leading dimension of matrix *C*. See `clAmdBlasSgemm()`.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Matrix-matrix product of general rectangular matrices with float-complex elements (Cont.)

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidValue` if either *offA*, *offB* or *offC* exceeds the size of the respective buffer object;
- otherwise, the same error codes as the `clAmdBlasSgemm()` function.

4.2.4 ZgemmEx

Matrix-matrix product of general rectangular matrices with double-complex elements

Function

```
clAmdBlasStatus clAmdBlasZgemmEx ( clAmdBlasOrder order, clAmdBlasTranspose
transA, clAmdBlasTranspose transB, size_t M, size_t N, size_t K, DoubleComplex
alpha, const cl_mem A, size_t offA, size_t lda, const cl_mem B, size_t offB,
size_t ldb, DoubleComplex beta, cl_mem C, size_t offC, size_t ldc, cl_uint
numCommandQueues, cl_command_queue * commandQueues, cl_uint
numEventsInWaitList, const cl_event * eventWaitList, cl_event * events )
```

Description

Matrix-matrix product of general rectangular matrices with double-complex elements. Extended version, which takes an offset value for all matrix arguments.

Matrix-matrix products:

- $C \leftarrow \alpha AB + \beta C$
- $C \leftarrow \alpha AB^T + \beta C$
- $C \leftarrow \alpha A^T B + \beta C$
- $C \leftarrow \alpha A^T B^T + \beta C$

Parameters

in	*order*	Row/column order.
in	*transA*	How matrix *A* is to be transposed.
in	*transB*	How matrix *B* is to be transposed.
in	*M*	Number of rows in matrix *A*.
in	*N*	Number of columns in matrix *B*.
in	*K*	Number of columns in matrix *A* and rows in matrix *B*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offA*	Offset of the first element of the matrix *A* in the buffer object. Counted in elements.
in	*lda*	Leading dimension of matrix *A*. For detailed description, see `clAmdBlasSgemm()`.
in	*B*	Buffer object storing matrix *B*.
in	*offB*	Offset of the first element of the matrix *B* in the buffer object. Counted in elements.
in	*ldb*	Leading dimension of matrix *B*. For detailed description, see `clAmdBlasSgemm()`.
in	*beta*	The factor of matrix *C*.
in/ out	*C*	Buffer object storing matrix *C*.
in	*offC*	Offset of the first element of the matrix *C* in the buffer object. Counted in elements.
in	*ldc*	Leading dimension of matrix *C*. For detailed description, see `clAmdBlasSgemm()`.
in	*numCommandQueues*	Number of OpenCL command queues in which the task should be performed.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Matrix-matrix product of general rectangular matrices with double-complex elements (Cont.)

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- `clAmdBlasInvalidValue` if either *offA*, *offB* or *offC* exceeds the size of the respective buffer object;
- otherwise, the same error codes as the `clAmdBlasSgemm()` function.

4.3 xTRMM -TRiangular Matrix-matrix Multiplication

4.3.1 Strmm

Multiplying a matrix by a triangular matrix with float elements

Function

clAmdBlasStatus clAmdBlasStrmm (clAmdBlasOrder *order*, clAmdBlasSide *side*, clAmdBlasUplo *uplo*, clAmdBlasTranspose *transA*, clAmdBlasDiag *diag*, size_t *M*, size_t *N*, cl_float *alpha*, const cl_mem *A*, size_t *lda*, cl_mem *B*, size_t *ldb*, cl_uint *numCommandQueues*, cl_command_queue * *commandQueues*, cl_uint *numEventsInWaitList*, const cl_event * *eventWaitList*, cl_event * *events*)

Description

This function has been deprecated.
Multiplying a matrix by a triangular matrix with float-complex elements.
Matrix-triangular matrix products are:

- $B \leftarrow \alpha AB$
- $B \leftarrow \alpha A^T B$
- $B \leftarrow \alpha BA$
- $B \leftarrow \alpha BA^T$

where *A* is an upper or lower triangular matrix.

Parameters

in	*order*	Row/column order.
in	*side*	The side of triangular matrix.
in	*uplo*	The triangle in matrix being referenced.
in	*transA*	How matrix *A* is to be transposed.
in	*diag*	Specify whether matrix is unit triangular.
in	*M*	Number of rows in matrix *B*.
in	*N*	Number of columns in matrix *B*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*lda*	Leading dimension of matrix *A*. See clAmdBlasStrmm().
in/ out	*B*	Buffer object storing matrix *B*.
in	*ldb*	Leading dimension of matrix *B*. See clAmdBlasStrmm().
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task. Currently, only one command queue is supported.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects in each command queue that identify a particular kernel execution instance.

Returns

0 on success; otherwise, following error codes.

- CL_INVALID_VALUE if invalid parameters are passed: *A*, *B*, or *C* object is invalid, or an image object rather than the buffer one; *M*, *N*, or *K* is zero, or a leading dimension is invalid, or the numCommandQueues parameter is not equal to 1.
- CL_OUT_OF_HOST_MEMORY if the library cannot allocate memory for internal structures.
- An error code from clEnqueueNDRangeKernel() function call.

4.3.2 Dtrmm

Multiplying a matrix by a triangular matrix with double elements

Function

`clAmdBlasStatus clAmdBlasDtrmm (clAmdBlasOrder` *`order`*`, clAmdBlasSide` *`side`*`, clAmdBlasUplo` *`uplo`*`, clAmdBlasTranspose` *`transA`*`, clAmdBlasDiag` *`diag`*`, size_t` *`M`*`, size_t` *`N`*`, cl_double` *`alpha`*`, const cl_mem` *`A`*`, size_t` *`lda`*`, cl_mem B, size_t` *`ldb`*`, cl_uint` *`numCommandQueues`*`, cl_command_queue *` *`commandQueues`*`, cl_uint` *`numEventsInWaitList`*`, const cl_event *` *`eventWaitList`*`, cl_event *` *`events`*`)`

Description

This function has been deprecated.
Multiplying a matrix by a triangular matrix with double elements.
Matrix-triangular matrix products are:

- $B \leftarrow \alpha AB$
- $B \leftarrow \alpha A^T B$
- $B \leftarrow \alpha BA$
- $B \leftarrow \alpha BA^T$

where *A* is an upper or lower triangular matrix.

Parameters

in	*order*	Row/column order.
in	*side*	The side of triangular matrix.
in	*uplo*	The triangle in matrix being referenced.
in	*transA*	How matrix *A* is to be transposed.
in	*diag*	Specify whether matrix is unit triangular.
in	*M*	Number of rows in matrix *B*.
in	*N*	Number of columns in matrix *B*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*lda*	Leading dimension of matrix *A*. See `clAmdBlasStrmm()`.
in/ out	*B*	Buffer object storing matrix *B*.
in	*ldb*	Leading dimension of matrix *B*. See `clAmdBlasStrmm()`.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task. Currently, only one command queue is supported.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects in each command queue that identify a particular kernel execution instance.

Returns

0 on success; otherwise, following error codes.

- `CL_INVALID_VALUE` if invalid parameters are passed: *A*, *B*, or *C* object is invalid, or an image object rather than the buffer one; *M*, *N*, or *K* is zero, or a leading dimension is invalid, or the `numCommandQueues` parameter is not equal to 1.
- `CL_OUT_OF_HOST_MEMORY` if the library cannot allocate memory for internal structures.
- An error code from `clEnqueueNDRangeKernel()` function call.

4.3.3 Ctrmm

Multiplying a matrix by a triangular matrix with float complex elements

Function

`clAmdBlasStatus clAmdBlasCtrmm (clAmdBlasOrder` *`order`*`, clAmdBlasSide` *`side`*`, clAmdBlasUplo` *`uplo`*`, clAmdBlasTranspose` *`transA`*`, clAmdBlasDiag` *`diag`*`, size_t` *`M`*`, size_t` *`N`*`, FloatComplex` *`alpha`*`, const cl_mem` *`A`*`, size_t` *`lda`*`, cl_mem` *`B`*`, size_t` *`ldb`*`, cl_uint` *`numCommandQueues`*`, cl_command_queue * ` *`commandQueues`*`, cl_uint` *`numEventsInWaitList`*`, const cl_event * ` *`eventWaitList`*`, cl_event * ` *`events`*`)`

Description

This function has been deprecated.
Multiplying a matrix by a triangular matrix with float elements.
Matrix-triangular matrix products are:

- $B \leftarrow \alpha AB$
- $B \leftarrow \alpha A^T B$
- $B \leftarrow \alpha BA$
- $B \leftarrow \alpha BA^T$

where *A* is an upper or lower triangular matrix.

Parameters

in	*order*	Row/column order.
in	*side*	The side of triangular matrix.
in	*uplo*	The triangle in matrix being referenced.
in	*transA*	How matrix *A* is to be transposed.
in	*diag*	Specify whether matrix is unit triangular.
in	*M*	Number of rows in matrix *B*.
in	*N*	Number of columns in matrix *B*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix A.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *M* when the `side` parameter is set to `clAmdBlasLeft`, or less than *N* when it is set to `clAmdBlasRight`.
in/ out	*B*	Buffer object storing matrix *B*.
in	*ldb*	Leading dimension of matrix *B*. It cannot be less than *N* when the `order` parameter is set to `clAmdBlasRowMajor`, or less than *M* when it is set to `clAmdBlasColumnMajor`.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task. Currently, only one command queue is supported.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects in each command queue that identify a particular kernel execution instance.

Multiplying a matrix by a triangular matrix with float complex elements (Cont.)

Returns 0 on success; otherwise, following error codes.

- `CL_INVALID_VALUE` if invalid parameters are passed: *A*, *B*, or *C* object is invalid, or an image object rather than the buffer one; *M*, *N*, or *K* is zero, or a leading dimension is invalid, or the `numCommandQueues` parameter is not equal to 1.
- `CL_OUT_OF_HOST_MEMORY` if the library cannot allocate memory for internal structures.
- An error code from `clEnqueueNDRangeKernel()` function call.

Examples `example_strmm.c`.

4.3.4 Ztrmm

Multiplying a matrix by a triangular matrix with double complex elements

Function

clAmdBlasStatus clAmdBlasZtrmm (clAmdBlasOrder *order*, clAmdBlasSide *side*, clAmdBlasUplo *uplo*, clAmdBlasTranspose *transA*, clAmdBlasDiag *diag*, size_t *M*, size_t *N*, DoubleComplex *alpha*, const cl_mem *A*, size_t *lda*, cl_mem *B*, size_t *ldb*, cl_uint *numCommandQueues*, cl_command_queue * *commandQueues*, cl_uint *numEventsInWaitList*, const cl_event * *eventWaitList*, cl_event * *events*)

Description

This function has been deprecated.
Multiplying a matrix by a triangular matrix with double-complex elements.
Matrix-triangular matrix products are:

- $B \leftarrow \alpha AB$
- $B \leftarrow \alpha A^T B$
- $B \leftarrow \alpha BA$
- $B \leftarrow \alpha BA^T$

where *A* is an upper or lower triangular matrix.

Parameters

in	*order*	Row/column order.
in	*side*	The side of triangular matrix.
in	*uplo*	The triangle in matrix being referenced.
in	*transA*	How matrix *A* is to be transposed.
in	*diag*	Specify whether matrix is unit triangular.
in	*M*	Number of rows in matrix *B*.
in	*N*	Number of columns in matrix *B*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*lda*	Leading dimension of matrix *A*. See clAmdBlasStrmm().
in/ out	*B*	Buffer object storing matrix *B*.
in	*ldb*	Leading dimension of matrix *B*. See clAmdBlasStrmm().
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task. Currently, only one command queue is supported.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects in each command queue that identify a particular kernel execution instance.

Returns

0 on success; otherwise, following error codes.

- CL_INVALID_VALUE if invalid parameters are passed: *A*, *B*, or *C* object is invalid, or an image object rather than the buffer one; *M*, *N*, or *K* is zero, or a leading dimension is invalid, or the numCommandQueues parameter is not equal to 1.
- CL_OUT_OF_HOST_MEMORY if the library cannot allocate memory for internal structures.
- An error code from clEnqueueNDRangeKernel() function call.

4.4 xTRMMEX - TRiangular Matrix-matrix Multiplication, Extended

4.4.1 StrmmEx

Multiplying a matrix by a triangular matrix with float elements

Function

```
clAmdBlasStatus clAmdBlasStrmmEx ( clAmdBlasOrder order, clAmdBlasSide side,
clAmdBlasUplo uplo, clAmdBlasTranspose transA, clAmdBlasDiag diag, size_t M,
size_t N, cl_float alpha, const cl_mem A, size_t offA, size_t lda, cl_mem B,
size_t offB, size_t ldb, cl_uint numCommandQueues, cl_command_queue *
commandQueues, cl_uint numEventsInWaitList, const cl_event *
eventWaitList, cl_event * events )
```

Description

Multiplying a matrix by a triangular matrix with float elements. Extended version, which takes an offset value for all matrix arguments.
Matrix-triangular matrix products:

- $B \leftarrow \alpha AB$
- $B \leftarrow \alpha BA$
- $B \leftarrow \alpha A^T B$
- $B \leftarrow \alpha BA^T$

where *A* is an upper or lower triangular matrix.

Parameters

in	*order*	Row/column order.
in	*side*	The side of triangular matrix.
in	*uplo*	The triangle in matrix being referenced.
in	*transA*	How matrix *A* is to be transposed.
in	*diag*	Specify whether matrix is unit triangular.
in	*M*	Number of rows in matrix *B*.
in	*N*	Number of columns in matrix *B*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offA*	Offset of the first element of the matrix *A* in the buffer object. Counted in elements.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *M* when the *side* parameter is set to `clAmdBlasLeft`, or less than *N* when it is set to `clAmdBlasRight`.
in/ out	*B*	Buffer object storing matrix *B*.
in	*offB*	Offset of the first element of the matrix *B* in the buffer object. Counted in elements.
in	*ldb*	Leading dimension of matrix *B*. It cannot be less than *N* when the *order* parameter is set to `clAmdBlasRowMajor`, or not less than *M* when it is set to `clAmdBlasColumnMajor`.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects in each command queue that identify a particular kernel execution instance.

Multiplying a matrix by a triangular matrix with float elements (Cont.)

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidValue` if either *offA* or *offB* exceeds the size of the respective buffer object;
- otherwise, the same error codes as `clAmdBlasStrmm()`.

4.4.2 DtrmmEx

Multiplying a matrix by a triangular matrix with double elements

Function

```
clAmdBlasStatus clAmdBlasDtrmmEx ( clAmdBlasOrder order, clAmdBlasSide side,
clAmdBlasUplo uplo, clAmdBlasTranspose transA, clAmdBlasDiag diag, size_t M,
size_t N, cl_double alpha, const cl_mem A, size_t offA, size_t lda, cl_mem
B, size_t offB, size_t ldb, cl_uint numCommandQueues, cl_command_queue *
commandQueues, cl_uint numEventsInWaitList, const cl_event *
eventWaitList, cl_event * events )
```

Description

Multiplying a matrix by a triangular matrix with double elements. Extended version, which takes an offset value for all matrix arguments.
Matrix-triangular matrix products:

- $B \leftarrow \alpha AB$
- $B \leftarrow \alpha A^T B$
- $B \leftarrow \alpha BA$
- $B \leftarrow \alpha BA^T$

where *A* is an upper or lower triangular matrix.

Parameters

in	*order*	Row/column order.
in	*side*	The side of triangular matrix.
in	*uplo*	The triangle in matrix being referenced.
in	*transA*	How matrix *A* is to be transposed.
in	*diag*	Specify whether matrix is unit triangular.
in	*M*	Number of rows in matrix *B*.
in	*N*	Number of columns in matrix *B*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offA*	Offset of the first element of the matrix A in the buffer object. Counted in elements.
in	*lda*	Leading dimension of matrix *A*. For detailed description, see `clAmdBlasStrmm()`.
in/ out	*B*	Buffer object storing matrix *B*.
in	*offB*	Offset of the first element of the matrix *B* in the buffer object. Counted in elements.
in	*ldb*	Leading dimension of matrix *B*. For detailed description, see `clAmdBlasStrmm()`.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- `clAmdBlasInvalidValue` if either *offA* or *offB* exceeds the size of the respective buffer object;
- otherwise, the same error codes as the `clAmdBlasStrmm()` function.

4.4.3 CtrmmEx

Multiplying a matrix by a triangular matrix with float-complex elements

Function

```
clAmdBlasStatus clAmdBlasCtrmmEx ( clAmdBlasOrder order, clAmdBlasSide side,
clAmdBlasUplo uplo, clAmdBlasTranspose transA, clAmdBlasDiag diag, size_t M,
size_t N, FloatComplex alpha, const cl_mem A, size_t offA, size_t lda, cl_mem
B, size_t offB, size_t ldb, cl_uint numCommandQueues, cl_command_queue *
commandQueues, cl_uint numEventsInWaitList, const cl_event *
eventWaitList, cl_event * events )
```

Description

Multiplying a matrix by a triangular matrix with float-complex elements. Extended version, which takes an offset value for all matrix arguments.
Matrix-triangular matrix products:

- $B \leftarrow \alpha AB$
- $B \leftarrow \alpha A^T B$
- $B \leftarrow \alpha BA$
- $B \leftarrow \alpha BA^T$

where *A* is an upper or lower triangular matrix.

Parameters

in	*order*	Row/column order.
in	*side*	The side of triangular matrix.
in	*uplo*	The triangle in matrix being referenced.
in	*transA*	How matrix *A* is to be transposed.
in	*diag*	Specify whether matrix is unit triangular.
in	*M*	Number of rows in matrix *B*.
in	*N*	Number of columns in matrix *B*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix A.
in	*offA*	Offset of the first element of the matrix *A* in the buffer object. Counted in elements.
in	*lda*	Leading dimension of matrix *A*. For detailed description, see `clAmdBlasStrmm()`.
in/ out	*B*	Buffer object storing matrix *B*.
in	*offB*	Offset of the first element of the matrix *B* in the buffer object. Counted in elements.
in	*ldb*	Leading dimension of matrix *B*. For detailed description, see `clAmdBlasStrmm()`.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidValue` if either *offA* or *offB* exceeds the size of the respective buffer object;
- otherwise, the same error codes as `clAmdBlasStrmm()`.

4.4.4 ZtrmmEx

Multiplying a matrix by a triangular matrix with double-complex elements

Function

```
clAmdBlasStatus clAmdBlasZtrmmEx ( clAmdBlasOrder order, clAmdBlasSide side,
clAmdBlasUplo uplo, clAmdBlasTranspose transA, clAmdBlasDiag diag, size_t M,
size_t N, DoubleComplex alpha, const cl_mem A, size_t offA, size_t lda, cl_mem
B, size_t offB, size_t ldb, cl_uint numCommandQueues, cl_command_queue *
commandQueues, cl_uint numEventsInWaitList, const cl_event *
eventWaitList, cl_event * events )
```

Description

Multiplying a matrix by a triangular matrix with double-complex elements. Extended version, which takes an offset value for all matrix arguments.
Matrix-triangular matrix products:

- $B \leftarrow \alpha AB$
- $B \leftarrow \alpha A^T B$
- $B \leftarrow \alpha BA$
- $B \leftarrow \alpha BA^T$

where *A* is an upper or lower triangular matrix.

Parameters

in	*order*	Row/column order.
in	*side*	The side of triangular matrix.
in	*uplo*	The triangle in matrix being referenced.
in	*transA*	How matrix *A* is to be transposed.
in	*diag*	Specify whether matrix is unit triangular.
in	*M*	Number of rows in matrix *B*.
in	*N*	Number of columns in matrix *B*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offA*	Offset of the first element of the matrix A in the buffer object. Counted in elements.
in	*lda*	Leading dimension of matrix *A*. For detailed description, see `clAmdBlasStrmm()`.
in/ out	*B*	Buffer object storing matrix *B*.
in	*offB*	Offset of the first element of the matrix B in the buffer object. Counted in elements.
in	*ldb*	Leading dimension of matrix *B*. For detailed description, see `clAmdBlasStrmm()`.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- `clAmdBlasInvalidValue` if either *offA* or *offB* exceeds the size of the respective buffer object;
- otherwise, the same error codes as the `clAmdBlasStrmm()` function.

4.5 xTRSM - TRiangular Matrix-matrix Solve

4.5.1 Strsm

Solving triangular systems of equations with multiple right-hand sides and float elements

Function

clAmdBlasStatus clAmdBlasStrsm (clAmdBlasOrder *order*, clAmdBlasSide *side*, clAmdBlasUplo *uplo*, clAmdBlasTranspose *transA*, clAmdBlasDiag *diag*, size_t *M*, size_t *N*, cl_float *alpha*, const cl_mem *A*, size_t *lda*, cl_mem *B*, size_t *ldb*, cl_uint *numCommandQueues*, cl_command_queue * *commandQueues*, cl_uint *numEventsInWaitList*, const cl_event * *eventWaitList*, cl_event * *events*)

Description

This function has been deprecated.

Solving triangular systems of equations with multiple right-hand sides and float-complex elements. Triangular systems of equations are:

- $B \leftarrow \alpha A^{-1}B$
- $B \leftarrow \alpha BA^{-1}$
- $B \leftarrow \alpha A^{-T}B$
- $B \leftarrow \alpha BA^{-T}$

where *A* is an upper or lower triangular matrix.

Parameters

in	*order*	Row/column order.
in	*side*	The side of triangular matrix.
in	*uplo*	The triangle in matrix being referenced.
in	*transA*	How matrix *A* is to be transposed.
in	*diag*	Specify whether matrix is unit triangular.
in	*M*	Number of rows in matrix *B*.
in	*N*	Number of columns in matrix *B*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *M* when the *side* parameter is set to clAmdBlasLeft, or less than *N* when it is set to clAmdBlasRight.
in/ out	*B*	Buffer object storing matrix *B*.
in	*ldb*	Leading dimension of matrix B. It cannot be less than *N* when the *order* parameter is set to clAmdBlasRowMajor, or less than *M* when it is set to clAmdBlasColumnMajor.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task. Currently, only one command queue is supported.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects in each command queue that identify a particular kernel execution instance.

Returns

0 on success; otherwise, following error codes.

- CL_INVALID_VALUE if invalid parameters are passed: *A*, *B*, or *C* object is invalid or an image object rather than the buffer one; *M*, *N*, or *K* is zero, or a leading dimension is invalid, or the numCommandQueues parameter is not equal to 1.
- CL_OUT_OF_HOST_MEMORY if the library cannot allocate memory for internal structures.
- An error code from clEnqueueNDRangeKernel() function call.

4.5.2 Dtrsm

Solving triangular systems of equations with multiple right-hand sides and double elements

Function

`cl_int clAmdBlasDtrsm (clAmdBlasOrder` *`order`*`, clAmdBlasSide` *`side`*`, clAmdBlasUplo` *`uplo`*`, clAmdBlasTranspose` *`transA`*`, clAmdBlasDiag` *`diag`*`, size_t` *`M`*`, size_t` *`N`*`, cl_double` *`alpha`*`, const cl_mem` *`A`*`, size_t` *`lda`*`, cl_mem` *`B`*`, size_t` *`ldb`*`, cl_uint` *`numCommandQueues`*`, cl_command_queue *` *`commandQueues`*`, cl_uint` *`numEventsInWaitList`*`, const cl_event *` *`eventWaitList`*`, cl_event *` *`events`*`)`

Description

This function has been deprecated.
Solving triangular systems of equations with multiple right-hand sides and double elements. Triangular systems of equations are:

- $B \leftarrow \alpha A^{-1} B$
- $B \leftarrow \alpha A^{-T} B$
- $B \leftarrow \alpha B A^{-1}$
- $B \leftarrow \alpha B A^{-T}$

where *A* is an upper or lower triangular matrix.

Parameters

in	*order*	Row/column order.
in	*side*	The side of triangular matrix.
in	*uplo*	The triangle in matrix being referenced.
in	*transA*	How matrix *A* is to be transposed.
in	*diag*	Specify whether matrix is unit triangular.
in	*M*	Number of rows in matrix *B*.
in	*N*	Number of columns in matrix *B*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*lda*	Leading dimension of matrix *A*. See clAmdBlasStrsm().
in/ out	*B*	Buffer object storing matrix *B*.
in	*ldb*	Leading dimension of matrix *B*. See clAmdBlasStrsm().
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task. Currently, only one command queue is supported.
in	*command-Queues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects in each command queue that identify a particular kernel execution instance.

Returns

0 on success; otherwise, following error codes.

- `CL_INVALID_VALUE` if invalid parameters are passed: *A*, *B*, or *C* object is invalid or an image object rather than the buffer one; *M*, *N*, or *K* is zero, or a leading dimension is invalid, or the `numCommandQueues` parameter is not equal to 1.
- `CL_OUT_OF_HOST_MEMORY` if the library cannot allocate memory for internal structures.
- An error code from `clEnqueueNDRangeKernel()` function call.

4.5.3 Ctrsm

Solving triangular systems of equations with multiple right-hand sides and float complex elements

Function

`cl_int clAmdBlasCtrsm (clAmdBlasOrder order, clAmdBlasSide side, clAmdBlasUplo uplo, clAmdBlasTranspose transA, clAmdBlasDiag diag, size_t M, size_t N, FloatComplex alpha, const cl_mem A, size_t lda, cl_mem B, size_t ldb, cl_uint numCommandQueues, cl_command_queue * commandQueues, cl_uint numEventsInWaitList, const cl_event * eventWaitList, cl_event * events)`

Description

This function has been deprecated.
Solving triangular systems of equations with multiple right-hand sides and float elements.
Triangular systems of equations are:

- $B \leftarrow \alpha A^{-1} B$
- $B \leftarrow \alpha A^{-T} B$
- $B \leftarrow \alpha B A^{-1}$
- $B \leftarrow \alpha B A^{-T}$

where *A* is an upper or lower triangular matrix.

Parameters

in	*order*	Row/column order.
in	*side*	The side of triangular matrix.
in	*uplo*	The triangle in matrix being referenced.
in	*transA*	How matrix *A* is to be transposed.
in	*diag*	Specify whether matrix is unit triangular.
in	*M*	Number of rows in matrix *B*.
in	*N*	Number of columns in matrix *B*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *N* when the `side` parameter is set to `clAmdBlasLeft`, or less than *M* when it is set to `clAmdBlasRight`.
in/ out	*B*	Buffer object storing matrix *B*.
in	*ldb*	Leading dimension of matrix *B*. It cannot be less than *N* when the `order` parameter is set to `clAmdBlasRowMajor`, or less than *M* when it is set to `clAmdBlasColumnMajor`.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task. Currently, only one command queue is supported.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects in each command queue that identify a particular kernel execution instance.

Solving triangular systems of equations with multiple right-hand sides and float complex elements (Cont.)

Returns 0 on success; otherwise, following error codes.

- `CL_INVALID_VALUE` if invalid parameters are passed: *A*, *B*, or *C* object is invalid or an image object rather than the buffer one; *M*, *N*, or *K* is zero, or a leading dimension is invalid, or the `numCommandQueues` parameter is not equal to 1.
- `CL_OUT_OF_HOST_MEMORY` if the library cannot allocate memory for internal structures.
- An error code from `clEnqueueNDRangeKernel()` function call.

Examples `example_strsm.c`.

4.5.4 Ztrsm

Solving triangular systems of equations with multiple right-hand sides and double complex elements

Function

`cl_int clAmdBlasZtrsm (clAmdBlasOrder` *`order`*`, clAmdBlasSide` *`side`*`, clAmdBlasUplo` *`uplo`*`, clAmdBlasTranspose` *`transA`*`, clAmdBlasDiag` *`diag`*`, size_t` *`M`*`, size_t` *`N`*`, DoubleComplex` *`alpha`*`, const cl_mem` *`A`*`, size_t` *`lda`*`, cl_mem` *`B`*`, size_t` *`ldb`*`, cl_uint` *`numCommandQueues`*`, cl_command_queue *` *`commandQueues`*`, cl_uint` *`numEventsInWaitList`*`, const cl_event *` *`eventWaitList`*`, cl_event *` *`events`*`)`

Description

This function has been deprecated.
Solving triangular systems of equations with multiple right-hand sides and double-complex elements.
Triangular systems of equations are:

- $B \leftarrow \alpha A^{-1} B$
- $B \leftarrow \alpha A^{-T} B$
- $B \leftarrow \alpha B A^{-1}$
- $B \leftarrow \alpha B A^{-T}$

where *A* is an upper or lower triangular matrix.

Parameters

in	*order*	Row/column order.
in	*side*	The side of triangular matrix.
in	*uplo*	The triangle in matrix being referenced.
in	*transA*	How matrix *A* is to be transposed.
in	*diag*	Specify whether matrix is unit triangular.
in	*M*	Number of rows in matrix *B*.
in	*N*	Number of columns in matrix *B*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*lda*	Leading dimension of matrix *A*. See `clAmdBlasStrsm()`.
in/ out	*B*	Buffer object storing matrix *B*.
in	*ldb*	Leading dimension of matrix *B*. See `clAmdBlasStrsm()`.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task. Currently, only one command queue is supported.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects in each command queue that identify a particular kernel execution instance.

Returns

0 on success; otherwise, following error codes.

- `CL_INVALID_VALUE` if invalid parameters are passed: *A*, *B*, or *C* object is invalid or an image object rather than the buffer one; *M*, *N*, or *K* is zero, or a leading dimension is invalid, or the `numCommandQueues` parameter is not equal to 1.
- `CL_OUT_OF_HOST_MEMORY` if the library cannot allocate memory for internal structures.
- An error code from `clEnqueueNDRangeKernel()` function call.

4.6 xTRSMEX - TRiangular Matrix-matrix Solve, Extended

4.6.1 StrsmEx

Solving triangular systems of equations with multiple right-hand sides and float elements

Function

```
clAmdBlasStatus clAmdBlasStrsmEx ( clAmdBlasOrder order, clAmdBlasSide side,
clAmdBlasUplo uplo, clAmdBlasTranspose transA, clAmdBlasDiag diag, size_t M,
size_t N, cl_float alpha, const cl_mem A, size_t offA, size_t lda, cl_mem B,
size_t offB, size_t ldb, cl_uint numCommandQueues, cl_command_queue *
commandQueues, cl_uint numEventsInWaitList, const cl_event *
eventWaitList, cl_event * events )
```

Description

Solving triangular systems of equations with multiple right-hand sides and float elements. Extended version, which takes an offset value for all matrix arguments.
Solving triangular systems of equations:

- $B \leftarrow \alpha A^{-1} B$
- $B \leftarrow \alpha B A^{-1}$
- $B \leftarrow \alpha A^{-T} B$
- $B \leftarrow \alpha B A^{-T}$

where *A* is an upper or lower triangular matrix.

Parameters

in	*order*	Row/column order.
in	*side*	The side of triangular matrix.
in	*uplo*	The triangle in matrix being referenced.
in	*transA*	How matrix *A* is to be transposed.
in	*diag*	Specify whether matrix is unit triangular.
in	*M*	Number of rows in matrix *B*.
in	*N*	Number of columns in matrix *B*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offA*	Offset of the first element of the matrix *A* in the buffer object. Counted in elements.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *M* when the *side* parameter is set to `clAmdBlasLeft`, or less than *N* when it is set to `clAmdBlasRight`.
in/ out	*B*	Buffer object storing matrix *B*.
in	*offB*	Offset of the first element of the matrix *B* in the buffer object. Counted in elements.
in	*ldb*	Leading dimension of matrix *B*. It cannot be less than *N* when the *order* parameter is set to `clAmdBlasRowMajor`, or less than *M* when it is set to `clAmdBlasColumnMajor`.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidValue` if either *offA* or *offB* exceeds the size of the respective buffer object;
- otherwise, the same error codes as `clAmdBlasStrsm()`.

4.6.2 DtrsmEx

Solving triangular systems of equations with multiple right-hand sides and double elements

Function

```
clAmdBlasStatus clAmdBlasDtrsmEx ( clAmdBlasOrder order,
clAmdBlasSide side, clAmdBlasUplo uplo, clAmdBlasTranspose transA,
clAmdBlasDiag diag, size_t M, size_t N, cl_double alpha, const cl_mem A,
size_t offA, size_t lda, cl_mem B, size_t offB, size_t ldb,
cl_uint numCommandQueues, cl_command_queue * commandQueues, cl_uint
numEventsInWaitList, const cl_event * eventWaitList, cl_event * events )
```

Description

Solving triangular systems of equations with multiple right-hand sides and double elements. Extended version, which takes an offset value for all matrix arguments.
Solving triangular systems of equations:

- $B \leftarrow \alpha A^{-1} B$
- $B \leftarrow \alpha A^{-T} B$
- $B \leftarrow \alpha B A^{-1}$
- $B \leftarrow \alpha B A^{-T}$

where *A* is an upper or lower triangular matrix.

Parameters

in	*order*	Row/column order.
in	*side*	The side of triangular matrix.
in	*uplo*	The triangle in matrix being referenced.
in	*transA*	How matrix *A* is to be transposed.
in	*diag*	Specify whether matrix is unit triangular.
in	*M*	Number of rows in matrix *B*.
in	*N*	Number of columns in matrix *B*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offA*	Offset of the first element of the matrix *A* in the buffer object. Counted in elements.
in	*lda*	Leading dimension of matrix *A*. For detailed description, see `clAmdBlasStrsm()`.
in/ out	*B*	Buffer object storing matrix *B*.
in	*offB*	Offset of the first element of the matrix *B* in the buffer object. Counted in elements.
in	*ldb*	Leading dimension of matrix *B*. For detailed description, see `clAmdBlasStrsm()`.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in	*command-Queues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- `clAmdBlasInvalidValue` if either *offA* or *offB* exceeds the size of the respective buffer object;
- otherwise, the same error codes as the `clAmdBlasStrsm()` function.

4.6.3 CtrsmEx

Solving triangular systems of equations with multiple right-hand sides and float-complex elements

Function

`clAmdBlasStatus clAmdBlasCtrsmEx ( clAmdBlasOrder` *`order`*`, clAmdBlasSide` *`side`*`, clAmdBlasUplo` *`uplo`*`, clAmdBlasTranspose` *`transA`*`, clAmdBlasDiag` *`diag`*`, size_t` *`M`*`, size_t` *`N`*`, FloatComplex` *`alpha`*`, const cl_mem` *`A`*`, size_t` *`offA`*`, size_t` *`lda`*`, cl_mem` *`B`*`, size_t` *`offB`*`, size_t` *`ldb`*`, cl_uint` *`numCommandQueues`*`, cl_command_queue *` *`commandQueues`*`, cl_uint` *`numEventsInWaitList`*`, const cl_event *` *`eventWaitList`*`, cl_event *` *`events`* `)`

Description

Solving triangular systems of equations with multiple right-hand sides and float-complex elements. Extended version, which takes an offset value for all matrix arguments.

Solving triangular systems of equations:

- $B \leftarrow \alpha A^{-1} B$
- $B \leftarrow \alpha B A^{-1}$
- $B \leftarrow \alpha A^{-T} B$
- $B \leftarrow \alpha B A^{-T}$

where *A* is an upper or lower triangular matrix.

Parameters

in	*order*	Row/column order.
in	*side*	The side of triangular matrix.
in	*uplo*	The triangle in matrix being referenced.
in	*transA*	How matrix *A* is to be transposed.
in	*diag*	Specify whether matrix is unit triangular.
in	*M*	Number of rows in matrix *B*.
in	*N*	Number of columns in matrix *B*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offA*	Offset of the first element of the matrix *A* in the buffer object. Counted in elements.
in	*lda*	Leading dimension of matrix *A*. For detailed description, see `clAmdBlasStrsm()`.
in/ out	*B*	Buffer object storing matrix *B*.
in	*offB*	Offset of the first element of the matrix *B* in the buffer object. Counted in elements.
in	*ldb*	Leading dimension of matrix *B*. For detailed description, see `clAmdBlasStrsm()`.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidValue` if either *offA* or *offB* exceeds the size of the respective buffer object;
- otherwise, the same error codes as `clAmdBlasStrsm()`.

4.6.4 ZtrsmEx

Solving triangular systems of equations with multiple right-hand sides and double-complex elements

Function

```
clAmdBlasStatus clAmdBlasZtrsmEx ( clAmdBlasOrder order, clAmdBlasSide side,
clAmdBlasUplo uplo, clAmdBlasTranspose transA, clAmdBlasDiag diag, size_t M,
size_t N, DoubleComplex alpha, const cl_mem A, size_t offA, size_t lda, cl_mem
B, size_t offB, size_t ldb, cl_uint numCommandQueues, cl_command_queue *
commandQueues, cl_uint numEventsInWaitList, const cl_event *
eventWaitList, cl_event * events )
```

Description

Solving triangular systems of equations with multiple right-hand sides and double-complex elements. Extended version, which takes an offset value for all matrix arguments.
Solving triangular systems of equations:

- $B \leftarrow \alpha A^{-1} B$
- $B \leftarrow \alpha A^{-T} B$
- $B \leftarrow \alpha B A^{-1}$
- $B \leftarrow \alpha B A^{-T}$

where *A* is an upper or lower triangular matrix.

Parameters

in	*order*	Row/column order.
in	*side*	The side of triangular matrix.
in	*uplo*	The triangle in matrix being referenced.
in	*transA*	How matrix *A* is to be transposed.
in	*diag*	Specify whether matrix is unit triangular.
in	*M*	Number of rows in matrix *B*.
in	*N*	Number of columns in matrix *B*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offA*	Offset of the first element of the matrix *A* in the buffer object. Counted in elements.
in	*lda*	Leading dimension of matrix *A*. For detailed description, see `clAmdBlasStrsm()`.
in/ out	*B*	Buffer object storing matrix *B*.
in	*offB*	Offset of the first element of the matrix *B* in the buffer object. Counted in elements.
in	*ldb*	Leading dimension of matrix *B*. For detailed description, see `clAmdBlasStrsm()`.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- `clAmdBlasInvalidValue` if either *offA* or *offB* exceeds the size of the respective buffer object;
- otherwise, the same error codes as the `clAmdBlasStrsm()` function.

4.7 xSYRK - SYmmetric Rank-K Update of a Matrix

4.7.1 Ssyrk

Rank-k update of a symmetric matrix with float elements

Function

```
clAmdBlasStatus clAmdBlasSsyrk ( clAmdBlasOrder order, clAmdBlasUplo uplo,
clAmdBlasTranspose transA, size_t N, size_t K, cl_float alpha, const cl_mem A,
size_t lda, cl_float beta, cl_mem C, size_t ldc, cl_uint numCommandQueues,
cl_command_queue * commandQueues, cl_uint numEventsInWaitList, const cl_event *
eventWaitList, cl_event * events )
```

Description

This function has been deprecated.
Rank-k update of a symmetric matrix with float elements.
Rank-k updates:

- $C \leftarrow \alpha AA^T + \beta C$
- $C \leftarrow \alpha A^TA + \beta C$

where *C* is a symmetric matrix.

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix *C* being referenced.
in	*transA*	How matrix *A* is to be transposed.
in	*N*	Number of rows and columns in matrix *C*.
in	*K*	Number of columns of the matrix *A* if it is not transposed; otherwise, number of rows.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing the matrix *A*.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *K* if *A* is in the row-major format; otherwise, less than *N*.
in	*beta*	The factor of the matrix *C*.
in/ out	*C*	Buffer object storing matrix *C*.
in	*ldc*	Leading dimension of matric *C*. It cannot be less than *N*.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Rank-k update of a symmetric matrix with float elements (Cont.)

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasNotInitialized` if `clAmdBlasSetup()` was not called;
- `clAmdBlasInvalidValue` if invalid parameters are passed:
  - either *M* or *N* is zero, or
  - the leading dimension is invalid;
- `clAmdBlasInvalidMemObject` if either *A* or *B* object is invalid, or an image object rather than the buffer one;
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures;
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid;
- `clAmdBlasInvalidContext` if a context a passed command queue belongs to was released.

Examples

`example_ssyrk.c`.

4.7.2 Dsyrk

Rank-k update of a symmetric matrix with double elements

Function

`clAmdBlasStatus clAmdBlasDsyrk ( clAmdBlasOrder order, clAmdBlasUplo` *`uplo`*`, clAmdBlasTranspose` *`transA`*`, size_t` *`N`*`, size_t` *`K`*`, cl_double` *`alpha`*`, const cl_mem` *`A`*`, size_t` *`lda`*`, cl_double` *`beta`*`, cl_mem` *`C`*`, size_t` *`ldc`*`, cl_uint` *`numCommandQueues`*`, cl_command_queue * ` *`commandQueues`*`, cl_uint` *`numEventsInWaitList`*`, const cl_event * ` *`eventWaitList`*`, cl_event * ` *`events`* `)`

Description

This function has been deprecated.
Rank-k update of a symmetric matrix with double elements.
Rank-k updates:

- $C \leftarrow \alpha AA^T + \beta C$
- $C \leftarrow \alpha A^TA + \beta C$

where *C* is a symmetric matrix.

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix *C* being referenced.
in	*transA*	How matrix *A* is to be transposed.
in	*N*	Number of rows and columns in matrix *C*.
in	*K*	Number of columns of the matrix *A* if it is not transposed; otherwise, number of rows.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing the matrix *A*.
in	*lda*	Leading dimension of matrix *A*. For detailed description, see `clAmdBlasSsyrk()`.
in	*beta*	The factor of the matrix *C*.
in/ out	*C*	Buffer object storing matrix *C*.
in	*ldc*	Leading dimension of matrix *C*. It cannot be less than *N*.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- For other error codes, see the `clAmdBlasSsyrk()` function.

4.7.3 Csyrk

Rank-k update of a symmetric matrix with complex float elements

Function

```
clAmdBlasStatus clAmdBlasCsyrk ( clAmdBlasOrder order,
clAmdBlasUplo uplo, clAmdBlasTranspose transA, size_t N, size_t K,
FloatComplex alpha, const cl_mem A, size_t lda, FloatComplex beta, cl_mem C,
size_t ldc, cl_uint numCommandQueues, cl_command_queue * commandQueues,
cl_uint numEventsInWaitList, const cl_event * eventWaitList,
cl_event * events )
```

Description

This function has been deprecated.
Rank-k update of a symmetric matrix with complex float elements.
Rank-k updates:

- $C \leftarrow \alpha AA^T + \beta C$
- $C \leftarrow \alpha A^T A + \beta C$

where *C* is a symmetric matrix.

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix *C* being referenced.
in	*transA*	How matrix *A* is to be transposed.
in	*N*	Number of rows and columns in matrix *C*.
in	*K*	Number of columns of the matrix *A* if it is not transposed; otherwise, number of rows.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing the matrix *A*.
in	*lda*	Leading dimension of matrix *A*. For detailed description, see `clAmdBlasSsyrk()`.
in	*beta*	The factor of the matrix *C*.
in/ out	*C*	Buffer object storing matrix *C*.
in	*ldc*	Leading dimension of matrix *C*. It cannot be less than *N*.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidValue` if *transA* is set to `clAmdBlasConjTrans`.
- otherwise, the same error codes as the `clAmdBlasSsyrk()` function.

4.7.4 Zsyrk

Rank-k update of a symmetric matrix with complex double elements

Function

```
clAmdBlasStatus clAmdBlasZsyrk ( clAmdBlasOrder order, clAmdBlasUplo uplo,
clAmdBlasTranspose transA, size_t N, size_t K, DoubleComplex alpha, const
cl_mem A, size_t lda, DoubleComplex beta, cl_mem C, size_t ldc, cl_uint
numCommandQueues, cl_command_queue * commandQueues, cl_uint
numEventsInWaitList, const cl_event * eventWaitList, cl_event * events )
```

Description

This function has been deprecated.
Rank-k update of a symmetric matrix with complex double elements.
Rank-k updates:

- $C \leftarrow \alpha AA^T + \beta C$
- $C \leftarrow \alpha A^T A + \beta C$

where *C* is a symmetric matrix.

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix *C* being referenced.
in	*transA*	How matrix *A* is to be transposed.
in	*N*	Number of rows and columns in matrix *C*.
in	*K*	Number of columns of the matrix *A* if it is not transposed; otherwise, number of rows.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing the matrix *A*.
in	*lda*	Leading dimension of matrix *A*. For detailed description, see `clAmdBlasSsyrk()`.
in	*beta*	The factor of the matrix *C*.
in/ out	*C*	Buffer object storing matrix *C*.
in	*ldc*	Leading dimension of matrix *C*. It cannot be less than *N*.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- `clAmdBlasInvalidValue` if *transA* is set to `clAmdBlasConjTrans`.
- otherwise, the same error codes as the `clAmdBlasSsyrk()` function.

4.8 xSYRKEX - SYmmetric Rank-K update of a matrix, Extended

4.8.1 SsyrkEx

Rank-k update of a symmetric matrix with float elements

Function

```
clAmdBlasStatus clAmdBlasSsyrkEx ( clAmdBlasOrder order, clAmdBlasUplo uplo,
clAmdBlasTranspose transA, size_t N, size_t K, cl_float alpha, const cl_mem
A, size_t offA, size_t lda, cl_float beta, cl_mem C, size_t offC, size_t
ldc, cl_uint numCommandQueues, cl_command_queue * commandQueues, cl_uint
numEventsInWaitList, const cl_event * eventWaitList, cl_event * events )
```

Description

Rank-k update of a symmetric matrix with float elements. Extended version, which takes an offset value for all matrix arguments.

Rank-k updates:

- $C \leftarrow \alpha AA^T + \beta C$
- $C \leftarrow \alpha A^T A + \beta C$

where *C* is a symmetric matrix.

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix *C* being referenced.
in	*transA*	How matrix *A* is to be transposed.
in	*N*	Number of rows and columns in matrix *C*.
in	*K*	Number of columns of the matrix *A* if it is not transposed; otherwise, number of rows.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing the matrix *A*.
in	*offA*	Offset of the first element of the matrix *A* in the buffer object. Counted in elements.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *K* if *A* is in the row-major format; otherwise, less than *N*.
in	*beta*	The factor of the matrix *C*.
in/ out	*C*	Buffer object storing matrix *C*.
in	*offC*	Offset of the first element of the matrix *C* in the buffer object. Counted in elements.
in	*ldc*	Leading dimension of matrix *C*. It cannot be less than *N*.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidValue` if either *offA* or *offC* exceeds the size of the respective buffer object;
- otherwise, the same error codes as the `clAmdBlasSsyrk()` function.

4.8.2 DsyrkEx

Rank-k update of a symmetric matrix with double elements

Function

```
clAmdBlasStatus clAmdBlasDsyrkEx ( clAmdBlasOrder order, clAmdBlasUplo uplo,
clAmdBlasTranspose transA, size_t N, size_t K, cl_double alpha, const cl_mem
A, size_t offA, size_t lda, cl_double beta, cl_mem C, size_t offC, size_t
ldc, cl_uint numCommandQueues, cl_command_queue * commandQueues, cl_uint
numEventsInWaitList, const cl_event * eventWaitList, cl_event * events )
```

Description

Rank-k update of a symmetric matrix with double elements. Extended version, which takes an offset value for all matrix arguments.
Rank-k updates:

- $C \leftarrow \alpha AA^T + \beta C$
- $C \leftarrow \alpha A^T A + \beta C$

where *C* is a symmetric matrix.

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix *C* being referenced.
in	*transA*	How matrix *A* is to be transposed.
in	*N*	Number of rows and columns in matrix *C*.
in	*K*	Number of columns of the matrix *A* if it is not transposed; otherwise, number of rows.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing the matrix *A*.
in	*offA*	Offset of the first element of the matrix *A* in the buffer object. Counted in elements.
in	*lda*	Leading dimension of matrix *A*. For detailed description, see `clAmdBlasSsyrk()`.
in	*beta*	The factor of the matrix *C*.
in/ out	*C*	Buffer object storing matrix *C*.
in	*offC*	Offset of the first element of the matrix *C* in the buffer object. Counted in elements.
in	*ldc*	Leading dimension of matrix *C*. It cannot be less than *N*.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- `clAmdBlasInvalidValue` if either *offA* or *offC* exceeds the size of the respective buffer object;
- otherwise, the same error codes as the `clAmdBlasSsyrk()` function.

4.8.3 CsyrkEx

Rank-k update of a symmetric matrix with complex float elements

Function

```
clAmdBlasStatus clAmdBlasCsyrkEx ( clAmdBlasOrder order, clAmdBlasUplo uplo,
clAmdBlasTranspose transA, size_t N, size_t K, FloatComplex alpha, const
cl_mem A, size_t offA, size_t lda, FloatComplex beta, cl_mem C, size_t offC,
size_t ldc, cl_uint numCommandQueues, cl_command_queue * commandQueues,
cl_uint numEventsInWaitList, const cl_event * eventWaitList, cl_event *
events )
```

Description

Rank-k update of a symmetric matrix with complex float elements. Extended version, which takes an offset value for all matrix arguments.
Rank-k updates:

- $C \leftarrow \alpha AA^T + \beta C$
- $C \leftarrow \alpha A^T A + \beta C$

where *C* is a symmetric matrix.

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix *C* being referenced.
in	*transA*	How matrix *A* is to be transposed.
in	*N*	Number of rows and columns in matrix *C*.
in	*K*	Number of columns of the matrix *A* if it is not transposed; otherwise, number of rows.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing the matrix *A*.
in	*offA*	Offset of the first element of the matrix *A* in the buffer object. Counted in elements.
in	*lda*	Leading dimension of matrix *A*. For detailed description, see `clAmdBlasSsyrk()`.
in	*beta*	The factor of the matrix *C*.
in/ out	*C*	Buffer object storing matrix *C*.
in	*offC*	Offset of the first element of the matrix *C* in the buffer object. Counted in elements.
in	*ldc*	Leading dimension of matrix *C*. It cannot be less than *N*.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidValue` if either *offA* or *offC* exceeds the size of the respective buffer object;
- `clAmdBlasInvalidValue` if *transA* is set to `clAmdBlasConjTrans`.
- otherwise, the same error codes as the `clAmdBlasSsyrk()` function.

4.8.4 ZsyrkEx

Rank-k update of a symmetric matrix with complex double elements

Function

```
clAmdBlasStatus clAmdBlasZsyrkEx ( clAmdBlasOrder order, clAmdBlasUplo uplo,
clAmdBlasTranspose transA, size_t N, size_t K, DoubleComplex alpha, const
cl_mem A, size_t offA, size_t lda, DoubleComplex beta, cl_mem C, size_t offC,
size_t ldc, cl_uint numCommandQueues, cl_command_queue * commandQueues,
cl_uint numEventsInWaitList, const cl_event * eventWaitList, cl_event *
events )
```

Description

Rank-k update of a symmetric matrix with complex double elements. Extended version, which takes an offset value for all matrix arguments.
Rank-k updates:

- $C \leftarrow \alpha AA^T + \beta C$
- $C \leftarrow \alpha A^T A + \beta C$

where *C* is a symmetric matrix.

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix *C* being referenced.
in	*transA*	How matrix *A* is to be transposed.
in	*N*	Number of rows and columns in matrix *C*.
in	*K*	Number of columns of the matrix *A* if it is not transposed; otherwise, number of rows.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing the matrix *A*.
in	*offA*	Offset of the first element of the matrix *A* in the buffer object. Counted in elements.
in	*lda*	Leading dimension of matrix *A*. For detailed description, see `clAmdBlasSsyrk()`.
in	*beta*	The factor of the matrix *C*.
in/ out	*C*	Buffer object storing matrix *C*.
in	*offC*	Offset of the first element of the matrix *C* in the buffer object. Counted in elements.
in	*ldc*	Leading dimension of matrix *C*. It cannot be less than *N*.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- `clAmdBlasInvalidValue` if either *offA* or *offC* exceeds the size of the respective buffer object;
- `clAmdBlasInvalidValue` if *transA* is set to `clAmdBlasConjTrans`.
- otherwise, the same error codes as the `clAmdBlasSsyrk()` function.

4.9 xSYR2K - SYmmetric Rank-2K update to a Matrix

4.9.1 Ssyr2k

Rank-2k update of a symmetric matrix with float elements

Function

```
clAmdBlasStatus clAmdBlasSsyr2k ( clAmdBlasOrder order, clAmdBlasUplo uplo,
clAmdBlasTranspose transAB, size_t N, size_t K, cl_float alpha, const cl_mem
A, size_t lda, const cl_mem B, size_t ldb, cl_float beta, cl_mem C, size_t
ldc, cl_uint numCommandQueues, cl_command_queue * commandQueues, cl_uint
numEventsInWaitList, const cl_event * eventWaitList, cl_event * events )
```

Description

This function has been deprecated.
Rank-2k update of a symmetric matrix with float elements.
Rank-k updates:

- $C \leftarrow \alpha AB^T + \alpha BA^T + \beta C$
- $C \leftarrow \alpha A^T B + \alpha B^T A \beta C$

where *C* is a symmetric matrix.

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix *C* being referenced.
in	*transAB*	How matrices *A* and *B* is to be transposed.
in	*N*	Number of rows and columns in matrix *C*.
in	*K*	Number of columns of the matrices *A* and *B* if they are not transposed; otherwise, number of rows.
in	*alpha*	The factor of matrices *A* and *B*.
in	*A*	Buffer object storing matrix *A*.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *K* if *A* is in the row-major format; otherwise, less than *N*.
in	*B*	Buffer object storing matrix *B*.
in	*ldb*	Leading dimension of matrix *B*. It cannot be clAmdBlasColumnMajor Order than *K* if *B* is in the row-major format; otherwise, less than *N*.
in	*beta*	The factor of matrix *C*.
in/ out	*C*	Buffer object storing matrix *C*.
in	*ldc*	Leading dimension of matrix *C*. It cannot be less than *N*.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Rank-2k update of a symmetric matrix with float elements (Cont.)

Returns

- `clAmdBlasSuccess` on success.
- `clAmdBlasNotInitialized` if `clAmdBlasSetup()` was not called.
- `clAmdBlasInvalidValue` if invalid parameters are passed:
  - either *N* or *K* is zero, or
  - the leading dimension is invalid.
- `clAmdBlasInvalidMemObject` if either *A*, *B*, or *C* object is invalid, or an image object rather than the buffer one.
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures.
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid.
- `clAmdBlasInvalidContext` if a context a passed command queue belongs to was released.
- `clAmdBlasInvalidOperation` if kernel compilation relating to a previous call has not completed for any of the target devices.
- `clAmdBlasCompilerNotAvailable` if a compiler is not available.
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

Examples `example_ssyr2k.c`.

4.9.2 Dsyr2k

Rank-2k update of a symmetric matrix with double elements

Function

```
clAmdBlasStatus clAmdBlasDsyr2k ( clAmdBlasOrder order, clAmdBlasUplo uplo,
clAmdBlasTranspose transAB, size_t N, size_t K, cl_double alpha, const cl_mem
A, size_t lda, const cl_mem B, size_t ldb, cl_double beta, cl_mem C, size_t
ldc, cl_uint numCommandQueues, cl_command_queue * commandQueues, cl_uint
numEventsInWaitList, const cl_event * eventWaitList, cl_event * events )
```

Description

This function has been deprecated.
Rank-2k update of a symmetric matrix with double elements.
Rank-k updates:

- $C \leftarrow \alpha AB^T + \alpha BA^T + \beta C$
- $C \leftarrow \alpha A^T B + \alpha B^T A \beta C$

where *C* is a symmetric matrix.

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix *C* being referenced.
in	*transAB*	How matrices *A* and *B* is to be transposed.
in	*N*	Number of rows and columns in matrix *C*.
in	*K*	Number of columns of the matrices *A* and *B* if they are not transposed; otherwise, number of rows.
in	*alpha*	The factor of matrices *A* and *B*.
in	*A*	Buffer object storing matrix *A*.
in	*lda*	Leading dimension of matrix *A*. See `clAmdBlasSsyr2k()`.
in	*B*	Buffer object storing matrix *B*.
in	*ldb*	Leading dimension of matrix *B*. See `clAmdBlasSsyr2k()`.
in	*beta*	The factor of matrix *C*.
in/ out	*C*	Buffer object storing matrix *C*.
in	*ldc*	Leading dimension of matrix *C*. It cannot be less than *N*.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- For other error codes, see the `clAmdBlasSsyr2k()` function.

4.9.3 Csyr2k

Rank-2k update of a symmetric matrix with complex float elements

Function

```
clAmdBlasStatus clAmdBlasCsyr2k ( clAmdBlasOrder order, clAmdBlasUplo uplo,
clAmdBlasTranspose transAB, size_t N, size_t K, FloatComplex alpha, const
cl_mem A, size_t lda, const cl_mem B, size_t ldb, FloatComplex beta, cl_mem
C, size_t ldc, cl_uint numCommandQueues, cl_command_queue * commandQueues,
cl_uint numEventsInWaitList, const cl_event * eventWaitList, cl_event *
events )
```

Description

This function has been deprecated.
Rank-2k update of a symmetric matrix with complex float elements.
Rank-k updates:

- $C \leftarrow \alpha AB^T + \alpha BA^T + \beta C$
- $C \leftarrow \alpha A^T B + \alpha B^T A \beta C$

where *C* is a symmetric matrix.

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix *C* being referenced.
in	*transAB*	How matrices *A* and *B* is to be transposed.
in	*N*	Number of rows and columns in matrix *C*.
in	*K*	Number of columns of the matrices *A* and *B* if they are not transposed; otherwise, number of rows.
in	*alpha*	The factor of matrices *A* and *B*.
in	*A*	Buffer object storing matrix *A*.
in	*lda*	Leading dimension of matrix *A*. For detailed description, see `clAmdBlasSsyr2k()`.
in	*B*	Buffer object storing matrix *B*.
in	*ldb*	Leading dimension of matrix *B*. For detailed description, see `clAmdBlasSsyr2k()`.
in	*beta*	The factor of matrix *C*.
in/ out	*C*	Buffer object storing matrix *C*.
in	*ldc*	Leading dimension of matrix *C*. It cannot be less than *N*.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidValue` if *transAB* is set to `clAmdBlasConjTrans`.
- otherwise, the same error codes as the `clAmdBlasSsyr2k()` function.

4.9.4 Zsyr2k

Rank-2k update of a symmetric matrix with complex double elements

Function

```
clAmdBlasStatus clAmdBlasZsyr2k ( clAmdBlasOrder order, clAmdBlasUplo uplo,
clAmdBlasTranspose transAB, size_t N, size_t K, DoubleComplex alpha, const
cl_mem A, size_t lda, const cl_mem B, size_t ldb, DoubleComplex beta, cl_mem
C, size_t ldc, cl_uint numCommandQueues, cl_command_queue * commandQueues,
cl_uint numEventsInWaitList, const cl_event * eventWaitList, cl_event *
events )
```

Description

This function has been deprecated.
Rank-2k update of a symmetric matrix with complex double elements.
Rank-k updates:

- $C \leftarrow \alpha AB^T + \alpha BA^T + \beta C$
- $C \leftarrow \alpha A^T B + \alpha B^T A \beta C$

where *C* is a symmetric matrix.

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix *C* being referenced.
in	*transAB*	How matrices *A* and *B* is to be transposed.
in	*N*	Number of rows and columns in matrix *C*.
in	*K*	Number of columns of the matrices *A* and *B* if they are not transposed; otherwise, number of rows.
in	*alpha*	The factor of matrices *A* and *B*.
in	*A*	Buffer object storing matrix *A*.
in	*lda*	Leading dimension of matrix *A*. For detailed description, see `clAmdBlasSsyr2k()`.
in	*B*	Buffer object storing matrix *B*.
in	*ldb*	Leading dimension of matrix *B*. For detailed description, see `clAmdBlasSsyr2k()`.
in	*beta*	The factor of matrix *C*.
in/ out	*C*	Buffer object storing matrix *C*.
in	*ldc*	Leading dimension of matrix *C*. It cannot be less than *N*.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- `clAmdBlasInvalidValue` if *transAB* is set to `clAmdBlasConjTrans`.
- otherwise, the same error codes as the `clAmdBlasSsyr2k()` function.

4.10 xSYR2KEX - SYmmetric Rank-2K update to a matrix, Extended

4.10.1 Ssyr2kEx

Rank-2k update of a symmetric matrix with float elements

Function

```
clAmdBlasStatus clAmdBlasSsyr2kEx ( clAmdBlasOrder order, clAmdBlasUplo uplo,
clAmdBlasTranspose transAB, size_t N, size_t K, cl_float alpha, const cl_mem
A, size_t offA, size_t lda, const cl_mem B, size_t offB, size_t ldb, cl_float
beta, cl_mem C, size_t offC, size_t ldc, cl_uint numCommandQueues,
cl_command_queue * commandQueues, cl_uint numEventsInWaitList, const
cl_event * eventWaitList, cl_event * events )
```

Description

Rank-2k update of a symmetric matrix with float elements. Extended version, which takes an offset value for all matrix arguments.

Rank-k updates:

- $C \leftarrow \alpha AB^T + \alpha BA^T + \beta C$
- $C \leftarrow \alpha A^T B + \alpha B^T A \beta C$

where *C* is a symmetric matrix.

Rank-2k update of a symmetric matrix with float elements (Cont.)

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix *C* being referenced.
in	*transAB*	How matrices *A* and *B* is to be transposed.
in	*N*	Number of rows and columns in matrix *C*.
in	*K*	Number of columns of the matrices *A* and *B* if they are not transposed; otherwise, number of rows.
in	*alpha*	The factor of matrices *A* and *B*.
in	*A*	Buffer object storing matrix *A*.
in	*offA*	Offset of the first element of the matrix *A* in the buffer object. Counted in elements.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *K* if *A* is in the row-major format; otherwise, less than *N*.
in	*B*	Buffer object storing matrix *B*.
in	*offB*	Offset of the first element of the matrix *B* in the buffer object. Counted in elements.
in	*ldb*	Leading dimension of matrix *B*. It cannot be less than *K* if *B* is in the row-major format; otherwise, less than *N*.
in	*beta*	The factor of matrix *C*.
in/ out	*C*	Buffer object storing matrix *C*.
in	*offC*	Offset of the first element of the matrix *C* in the buffer object. Counted in elements.
in	*ldc*	Leading dimension of matrix *C*. It cannot be less than *N*.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidValue` if either *offA*, *offB* or *offC* exceeds the size of the respective buffer object;
- otherwise, the same error codes as the `clAmdBlasSsyr2k()` function.

4.10.2 Dsyr2kEx

Rank-2k update of a symmetric matrix with double elements

Function

```
clAmdBlasStatus clAmdBlasDsyr2kEx ( clAmdBlasOrder order, clAmdBlasUplo uplo,
clAmdBlasTranspose transAB, size_t N, size_t K, cl_double alpha, const cl_mem
A, size_t offA, size_t lda, const cl_mem B, size_t offB, size_t ldb, cl_double
beta, cl_mem C, size_t offC, size_t ldc, cl_uint numCommandQueues,
cl_command_queue * commandQueues, cl_uint snumEventsInWaitList, const
cl_event * eventWaitList, cl_event * events )
```

Description

Rank-2k update of a symmetric matrix with double elements. Extended version, which takes an offset value for all matrix arguments.
Rank-k updates:

- $C \leftarrow \alpha AB^T + \alpha BA^T + \beta C$
- $C \leftarrow \alpha A^T B + \alpha B^T A \beta C$

where *C* is a symmetric matrix.

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix *C* being referenced.
in	*transAB*	How matrices *A* and *B* is to be transposed.
in	*N*	Number of rows and columns in matrix *C*.
in	*K*	Number of columns of the matrices *A* and *B* if they are not transposed; otherwise, number of rows.
in	*alpha*	The factor of matrices *A* and *B*.
in	*A*	Buffer object storing matrix *A*.
in	*offA*	Offset of the first element of the matrix *A* in the buffer object. Counted in elements.
in	*lda*	Leading dimension of matrix *A*. For detailed description, see `clAmdBlasSsyr2k()`.
in	*B*	Buffer object storing matrix *B*.
in	*offB*	Offset of the first element of the matrix *B* in the buffer object. Counted in elements.
in	*ldb*	Leading dimension of matrix *B*. For detailed description, see `clAmdBlasSsyr2k()`.
in	*beta*	The factor of matrix *C*.
in/ out	*C*	Buffer object storing matrix *C*.
in	*offC*	Offset of the first element of the matrix *C* in the buffer object. Counted in elements.
in	*ldc*	Leading dimension of matrix *C*. It cannot be less than *N*.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Rank-2k update of a symmetric matrix with double elements (Cont.)

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- `clAmdBlasInvalidValue` if either *offA*, *offB* or *offC* exceeds the size of the respective buffer object;
- otherwise, the same error codes as the `clAmdBlasSsyr2k()` function.

4.10.3 Csyr2kEx

Rank-2k update of a symmetric matrix with complex float elements

Function

```
clAmdBlasStatus clAmdBlasCsyr2kEx ( clAmdBlasOrder order, clAmdBlasUplo uplo,
clAmdBlasTranspose transAB, size_t N, size_t K, FloatComplex alpha, const
cl_mem A, size_t offA, size_t lda, const cl_mem B, size_t offB, size_t ldb,
FloatComplex beta, cl_mem C, size_t offC, size_t ldc, cl_uint
numCommandQueues, cl_command_queue * commandQueues, cl_uint
numEventsInWaitList, const cl_event * eventWaitList, cl_event * events )
```

Description

Rank-2k update of a symmetric matrix with complex float elements. Extended version, which takes an offset value for all matrix arguments.
Rank-k updates:

- $C \leftarrow \alpha AB^T + \alpha BA^T + \beta C$
- $C \leftarrow \alpha A^T B + \alpha B^T A \beta C$

where *C* is a symmetric matrix.

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix *C* being referenced.
in	*transAB*	How matrices *A* and *B* is to be transposed.
in	*N*	Number of rows and columns in matrix *C*.
in	*K*	Number of columns of the matrices *A* and *B* if they are not transposed; otherwise, number of rows.
in	*alpha*	The factor of matrices *A* and *B*.
in	*A*	Buffer object storing matrix *A*.
in	*offA*	Offset of the first element of the matrix *A* in the buffer object. Counted in elements.
in	*lda*	Leading dimension of matrix *A*. For detailed description, see `clAmdBlasSsyr2k()`.
in	*B*	Buffer object storing matrix *B*.
in	*offB*	Offset of the first element of the matrix *B* in the buffer object. Counted in elements.
in	*ldb*	Leading dimension of matrix *B*. For detailed description, see `clAmdBlasSsyr2k()`.
in	*beta*	The factor of matrix *C*.
in/ out	*C*	Buffer object storing matrix *C*.
in	*offC*	Offset of the first element of the matrix *C* in the buffer object. Counted in elements.
in	*ldc*	Leading dimension of matrix *C*. It cannot be less than *N*.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Rank-2k update of a symmetric matrix with complex float elements (Cont.)

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidValue` if either *offA*, *offB* or *offC* exceeds the size of the respective buffer object;
- `clAmdBlasInvalidValue` if *transAB* is set to `clAmdBlasConjTrans`.
- otherwise, the same error codes as the `clAmdBlasSsyr2k()` function.

4.10.4 Zsyr2kEx

Rank-2k update of a symmetric matrix with complex double elements

Function

```
clAmdBlasStatus clAmdBlasZsyr2kEx ( clAmdBlasOrder order, clAmdBlasUplo uplo,
clAmdBlasTranspose transAB, size_t N, size_t K, DoubleComplex alpha, const
cl_mem A, size_t offA, size_t lda, const cl_mem B, size_t offB, size_t ldb,
DoubleComplex beta, cl_mem C, size_t offC, size_t ldc, cl_uint
numCommandQueues, cl_command_queue * commandQueues, cl_uint
numEventsInWaitList, const cl_event * eventWaitList, cl_event * events )
```

Description

Rank-2k update of a symmetric matrix with complex double elements. Extended version, which takes an offset value for all matrix arguments.

Rank-k updates:

- $C \leftarrow \alpha AB^T + \alpha BA^T + \beta C$
- $C \leftarrow \alpha A^T B + \alpha B^T A \beta C$

where *C* is a symmetric matrix.

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix *C* being referenced.
in	*transAB*	How matrices *A* and *B* is to be transposed.
in	*N*	Number of rows and columns in matrix *C*.
in	*K*	Number of columns of the matrices *A* and *B* if they are not transposed; otherwise, number of rows.
in	*alpha*	The factor of matrices *A* and *B*.
in	*A*	Buffer object storing matrix *A*.
in	*offA*	Offset of the first element of the matrix *A* in the buffer object. Counted in elements.
in	*lda*	Leading dimension of matrix *A*. For detailed description, see `clAmdBlasSsyr2k()`.
in	*B*	Buffer object storing matrix *B*.
in	*offB*	Offset of the first element of the matrix *B* in the buffer object. Counted in elements.
in	*ldb*	Leading dimension of matrix *B*. For detailed description, see `clAmdBlasSsyr2k()`.
in	*beta*	The factor of matrix *C*.
in/ out	*C*	Buffer object storing matrix *C*.
in	*offC*	Offset of the first element of the matrix *C* in the buffer object. Counted in elements.
in	*ldc*	Leading dimension of matrix *C*. It cannot be less than *N*.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Rank-2k update of a symmetric matrix with complex double elements (Cont.)

Returns

- `clAmdBlasSuccess` on success;
- *clAmdBlasInvalidDevice* if a target device does not support floating point arithmetic with double precision;
- `clAmdBlasInvalidValue` if either *offA*, *offB* or *offC* exceeds the size of the respective buffer object;
- *clAmdBlasInvalidValue* if *transAB* is set to `clAmdBlasConjTrans`.
- otherwise, the same error codes as the `clAmdBlasSsyr2k()` function.

4.11 xSYMM - SYmmetric Matrix-matrix Multiply

4.11.1 Ssymm

Matrix-matrix product of symmetric rectangular matrices with float elements

Function

```
clAmdBlasStatus clAmdBlasSsymm ( clAmdBlasOrder order, clAmdBlasSide side,
clAmdBlasUplo uplo, size_t M, size_t N, cl_float alpha, const cl_mem A, size_t
offa, size_t lda, const cl_mem B, size_t offb, size_t ldb, cl_float beta,
cl_mem C, size_t offc, size_t ldc, cl_uint numCommandQueues, cl_command_queue
* commandQueues, cl_uint numEventsInWaitList, const cl_event *
eventWaitList, cl_event * events )
```

Description

Matrix-matrix product of symmetric rectangular matrices with float elements.
Matrix-matrix products:

- $C \leftarrow \alpha AB + \beta C$
- $C \leftarrow \alpha BA + \beta C$

Parameters

in	*order*	Row/column order.
in	*side*	The side of triangular matrix.
in	*uplo*	The triangle in matrix being referenced.
in	*M*	Number of rows in matrices *B* and *C*.
in	*N*	Number of columns in matrices *B* and *C*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offa*	Offset of the first element of the matrix *A* in the buffer object. Counted in elements.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *M* when the *side* parameter is set to `clAmdBlasLeft`, or less than *N* when the parameter is set to `clAmdBlasRight`.
in	*B*	Buffer object storing matrix *B*.
in	*offb*	Offset of the first element of the matrix *B* in the buffer object. Counted in elements.
in	*ldb*	Leading dimension of matrix *B*. It cannot be less than *N* when the *order* parameter is set to `clAmdBlasRowMajor`, or less than *M* when it is set to `clAmdBlasColumnMajor`.
in	*beta*	The factor of matrix *C*.
in/ out	*C*	Buffer object storing matrix *C*.
in	*offc*	Offset of the first element of the matrix *C* in the buffer object. Counted in elements.
in	*ldc*	Leading dimension of matrix *C*. It cannot be less than *N* when the *order* parameter is set to `clAmdBlasRowMajor`, or less than *M* when it is set to `clAmdBlasColumnMajor`.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Matrix-matrix product of symmetric rectangular matrices with float elements (Cont.)

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasNotInitialized` if `clAmdBlasSetup()` was not called;
- `clAmdBlasInvalidValue` if invalid parameters are passed:
  - *M* or *N* is zero, or
  - any of the leading dimensions is invalid;
  - the matrix sizes lead to accessing outsize of any of the buffers;
- `clAmdBlasInvalidMemObject` if *A*, *B*, or *C* object is invalid, or an image object rather than the buffer one;
- `clAmdBlasOutOfResources` if you use image-based function implementation and no suitable scratch image available;
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures;
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid;
- `clAmdBlasInvalidContext` if a context a passed command queue belongs to was released;
- `clAmdBlasInvalidOperation` if kernel compilation relating to a previous call has not completed for any of the target devices;
- `clAmdBlasCompilerNotAvailable` if a compiler is not available;
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

Examples

`example_ssymm.c`.

4.11.2 Dsymm

Matrix-matrix product of symmetric rectangular matrices with double elements

Function

```
clAmdBlasStatus clAmdBlasDsymm ( clAmdBlasOrder order, clAmdBlasSide side,
clAmdBlasUplo uplo, size_t M, size_t N, cl_double alpha, const cl_mem A,
size_t offa, size_t lda, const cl_mem B, size_t offb, size_t ldb, cl_double
beta, cl_mem C, size_t offc, size_t ldc, cl_uint numCommandQueues,
cl_command_queue * commandQueues, cl_uint numEventsInWaitList, const
cl_event * eventWaitList, cl_event * events )
```

Description

Matrix-matrix product of symmetric rectangular matrices with double elements.
Matrix-matrix products:

- $C \leftarrow \alpha AB + \beta C$
- $C \leftarrow \alpha BA + \beta C$

Parameters

in	*order*	Row/column order.
in	*side*	The side of triangular matrix.
in	*uplo*	The triangle in matrix being referenced.
in	*M*	Number of rows in matrices *B* and *C*.
in	*N*	Number of columns in matrices *B* and *C*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offa*	Offset of the first element of the matrix *A* in the buffer object. Counted in elements.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *M* when the *side* parameter is set to `clAmdBlasLeft`, or less than *N* when the parameter is set to `clAmdBlasRight`.
in	*B*	Buffer object storing matrix *B*.
in	*offb*	Offset of the first element of the matrix *B* in the buffer object. Counted in elements.
in	*ldb*	Leading dimension of matrix *B*. It cannot be less than *N* when the *order* parameter is set to `clAmdBlasRowMajor`, or less than *M* when it is set to `clAmdBlasColumnMajor`.
in	*beta*	The factor of matrix *C*.
in/ out	*C*	Buffer object storing matrix *C*.
in	*offc*	Offset of the first element of the matrix *C* in the buffer object. Counted in elements.
in	*ldc*	Leading dimension of matrix *C*. It cannot be less than *N* when the *order* parameter is set to `clAmdBlasRowMajor`, or less than *M* when it is set to `clAmdBlasColumnMajor`.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Matrix-matrix product of symmetric rectangular matrices with double elements (Cont.)

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- otherwise, the same error codes as the `clAmdBlasSsymm()` function.

4.11.3 Csymm

Matrix-matrix product of symmetric rectangular matrices with float-complex elements

Function

```
clAmdBlasStatus clAmdBlasCsymm ( clAmdBlasOrder order, clAmdBlasSide side,
clAmdBlasUplo uplo, size_t M, size_t N, cl_float2 alpha, const cl_mem A,
size_t offa, size_t lda, const cl_mem B, size_t offb, size_t ldb, cl_float2
beta, cl_mem C, size_t offc, size_t ldc, cl_uint numCommandQueues,
cl_command_queue * commandQueues, cl_uint numEventsInWaitList, const
cl_event * eventWaitList, cl_event * events )
```

Description

Matrix-matrix product of symmetric rectangular matrices with float-complex elements.
Matrix-matrix products:

- $C \leftarrow \alpha AB + \beta C$
- $C \leftarrow \alpha BA + \beta C$

Parameters

in	*order*	Row/column order.
in	*side*	The side of triangular matrix.
in	*uplo*	The triangle in matrix being referenced.
in	*M*	Number of rows in matrices *B* and *C*.
in	*N*	Number of columns in matrices *B* and *C*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offa*	Offset of the first element of the matrix *A* in the buffer object. Counted in elements.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *M* when the *side* parameter is set to `clAmdBlasLeft`, or less than *N* when the parameter is set to `clAmdBlasRight`.
in	*B*	Buffer object storing matrix *B*.
in	*offb*	Offset of the first element of the matrix *B* in the buffer object. Counted in elements.
in	*ldb*	Leading dimension of matrix *B*. It cannot be less than *N* when the *order* parameter is set to `clAmdBlasRowMajor`, or less than *M* when it is set to `clAmdBlasColumnMajor`.
in	*beta*	The factor of matrix *C*.
in/ out	*C*	Buffer object storing matrix *C*.
in	*offc*	Offset of the first element of the matrix *C* in the buffer object. Counted in elements.
in	*ldc*	Leading dimension of matrix *C*. It cannot be less than *N* when the *order* parameter is set to `clAmdBlasRowMajor`, or less than *M* when it is set to `clAmdBlasColumnMajor`.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

The same result as the `clAmdBlasSsymm()` function.

4.11.4 Zsymm

Matrix-matrix product of symmetric rectangular matrices with double-complex elements

Function

```
clAmdBlasStatus clAmdBlasZsymm ( clAmdBlasOrder order, clAmdBlasSide side,
clAmdBlasUplo uplo, size_t M, size_t N, cl_double2 alpha, const cl_mem A,
size_t offa, size_t lda, const cl_mem B, size_t offb, size_t ldb, cl_double2
beta, cl_mem C, size_t offc, size_t ldc, cl_uint numCommandQueues,
cl_command_queue * commandQueues, cl_uint numEventsInWaitList, const
cl_event * eventWaitList, cl_event * events )
```

Description

Matrix-matrix product of symmetric rectangular matrices with double-complex elements. Products are:

- $C \leftarrow \alpha AB + \beta C$
- $C \leftarrow \alpha BA + \beta C$

Parameters

in	*order*	Row/column order.
in	*side*	The side of triangular matrix.
in	*uplo*	The triangle in matrix being referenced.
in	*M*	Number of rows in matrices *B* and *C*.
in	*N*	Number of columns in matrices *B* and *C*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offa*	Offset of the first element of the matrix *A* in the buffer object. Counted in elements.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *M* when the *side* parameter is set to `clAmdBlasLeft`, or less than *N* when the parameter is set to `clAmdBlasRight`.
in	*B*	Buffer object storing matrix *B*.
in	*offb*	Offset of the first element of the matrix *B* in the buffer object. Counted in elements.
in	*ldb*	Leading dimension of matrix *B*. It cannot be less than *N* when the *order* parameter is set to `clAmdBlasRowMajor`, or less than *M* when it is set to `clAmdBlasColumnMajor`.
in	*beta*	The factor of matrix *C*.
in/ out	*C*	Buffer object storing matrix *C*.
in	*offc*	Offset of the first element of the matrix *C* in the buffer object. Counted in elements.
in	*ldc*	Leading dimension of matrix *C*. It cannot be less than *N* when the *order* parameter is set to `clAmdBlasRowMajor`, or less than *M* when it is set to `clAmdBlasColumnMajor`.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Matrix-matrix product of symmetric rectangular matrices with double-complex elements

Returns	The same result as the `clAmdBlasDsymm()` function.

4.12 xHEMM - HErmitian Matrix-matrix Multiply

4.12.1 Chemm

Matrix-matrix product of hermitian rectangular matrices with float-complex elements

Function

```
clAmdBlasStatus clAmdBlasChemm ( clAmdBlasOrder order,
clAmdBlasSide side, clAmdBlasUplo uplo, size_t M, size_t N, cl_float2
alpha, const cl_mem A, size_t offa, size_t lda, const cl_mem B, size_t offb,
size_t ldb, cl_float2 beta, cl_mem C, size_t offc, size_t ldc, cl_uint
numCommandQueues, cl_command_queue * commandQueues, cl_uint
numEventsInWaitList, const cl_event * eventWaitList, cl_event * events )
```

Description

Matrix-matrix product of hermitian rectangular matrices with float-complex elements.
Matrix-matrix products:

- $C \leftarrow \alpha AB + \beta C$
- $C \leftarrow \alpha BA + \beta C$

Parameters

in	*order*	Row/column order.
in	*side*	The side of triangular matrix.
in	*uplo*	The triangle in matrix being referenced.
in	*M*	Number of rows in matrices *B* and *C*.
in	*N*	Number of columns in matrices *B* and *C*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offa*	Offset of the first element of the matrix *A* in the buffer object. Counted in elements.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *M* when the *side* parameter is set to `clAmdBlasLeft`, or less than *N* when the parameter is set to `clAmdBlasRight`.
in	*B*	Buffer object storing matrix *B*.
in	*offb*	Offset of the first element of the matrix *B* in the buffer object. Counted in elements.
in	*ldb*	Leading dimension of matrix *B*. It cannot be less than *N* when the *order* parameter is set to `clAmdBlasRowMajor`, or less than *M* when it is set to `clAmdBlasColumnMajor`.
in	*beta*	The factor of matrix *C*.
in/ out	*C*	Buffer object storing matrix *C*.
in	*offc*	Offset of the first element of the matrix *C* in the buffer object. Counted in elements.
in	*ldc*	Leading dimension of matrix *C*. It cannot be less than *N* when the *order* parameter is set to `clAmdBlasRowMajor`, or less than *M* when it is set to `clAmdBlasColumnMajor`.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Matrix-matrix product of hermitian rectangular matrices with float-complex elements (Cont.)

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasNotInitialized` if `clAmdBlasSetup()` was not called;
- `clAmdBlasInvalidValue` if invalid parameters are passed:
  - *M* or *N* is zero, or
  - any of the leading dimensions is invalid;
  - the matrix sizes lead to accessing outsize of any of the buffers;
- `clAmdBlasInvalidMemObject` if *A*, *B*, or *C* object is invalid, or an image object rather than the buffer one;
- `clAmdBlasOutOfResources` if you use image-based function implementation and no suitable scratch image available;
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures;
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid;
- `clAmdBlasInvalidContext` if a context a passed command queue belongs to was released;
- `clAmdBlasInvalidOperation` if kernel compilation relating to a previous call has not completed for any of the target devices;
- `clAmdBlasCompilerNotAvailable` if a compiler is not available;
- `clAmdBlasBuildProgramFailure` if there is a failure to build a program executable.

Examples

`example_chemm.cpp`.

4.12.2 Zhemm

Matrix-matrix product of hermitian rectangular matrices with double-complex elements

Function

```
clAmdBlasStatus clAmdBlasZhemm ( clAmdBlasOrder order, clAmdBlasSide side,
clAmdBlasUplo uplo, size_t M, size_t N, cl_double2 alpha, const cl_mem A,
size_t offa, size_t lda, const cl_mem B, size_t offb, size_t ldb, cl_double2
beta, cl_mem C, size_t offc, size_t ldc, cl_uint numCommandQueues,
cl_command_queue * commandQueues, cl_uint numEventsInWaitList, const
cl_event * eventWaitList, cl_event * events )
```

Description

Matrix-matrix product of hermitian rectangular matrices with double-complex elements. Matrix-matrix products:

- $C \leftarrow \alpha AB + \beta C$
- $C \leftarrow \alpha BA + \beta C$

Parameters

in	*order*	Row/column order.
in	*side*	The side of triangular matrix.
in	*uplo*	The triangle in matrix being referenced.
in	*M*	Number of rows in matrices *B* and *C*.
in	*N*	Number of columns in matrices *B* and *C*.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offa*	Offset of the first element of the matrix *A* in the buffer object. Counted in elements.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *M* when the *side* parameter is set to `clAmdBlasLeft`, or less than *N* when the parameter is set to `clAmdBlasRight`.
in	*B*	Buffer object storing matrix *B*.
in	*offb*	Offset of the first element of the matrix *B* in the buffer object. Counted in elements.
in	*ldb*	Leading dimension of matrix *B*. It cannot be less than *N* when the *order* parameter is set to `clAmdBlasRowMajor`, or less than *M* when it is set to `clAmdBlasColumnMajor`.
in	*beta*	The factor of matrix *C*.
in/ out	*C*	Buffer object storing matrix *C*.
in	*offc*	Offset of the first element of the matrix *C* in the buffer object. Counted in elements.
in	*ldc*	Leading dimension of matrix *C*. It cannot be less than *N* when the *order* parameter is set to `clAmdBlasRowMajor`, or less than *M* when it is set to `clAmdBlasColumnMajor`.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Matrix-matrix product of hermitian rectangular matrices with double-complex elements

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- otherwise, the same error codes as the `clAmdBlasChemm()` function.

4.13 xHERK - HErmitian Rank-K update to a matrix

4.13.1 Cherk

Rank-k update of a hermitian matrix with float-complex elements

Function

```
clAmdBlasStatus clAmdBlasCherk ( clAmdBlasOrder order, clAmdBlasUplo uplo,
clAmdBlasTranspose transA, size_t N, size_t K, float alpha, const cl_mem A,
size_t offa, size_t lda, float beta, cl_mem C, size_t offc, size_t ldc,
cl_uint numCommandQueues, cl_command_queue * commandQueues, cl_uint
numEventsInWaitList, const cl_event * eventWaitList, cl_event * events )
```

Description

Rank-k update of a hermitian matrix with float-complex elements.
Rank-k updates:

- $C \leftarrow \alpha AA^H + \beta C$
- $C \leftarrow \alpha A^H A + \beta C$

where *C* is a hermitian matrix.

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix *C* being referenced.
in	*transA*	How matrix *A* is to be transposed.
in	*N*	Number of rows and columns in matrix *C*.
in	*K*	Number of columns of the matrix *A* if they are not transposed; otherwise, number of rows.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offa*	Offset in number of elements for the first element in matrix *A*.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *K* if *A* is in the row-major format; otherwise, less than *N*.
in	*beta*	The factor of matrix *C*.
in/ out	*C*	Buffer object storing matrix *C*.
in	*offc*	Offset in number of elements for the first element in matrix *C*.
in	*ldc*	Leading dimension of matrix *C*. It cannot be less than *N*.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Rank-k update of a hermitian matrix with float-complex elements (Cont.)

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasNotInitialized` if `clAmdBlasSetup()` was not called;
- `clAmdBlasInvalidValue` if invalid parameters are passed:
  - either *N* or *K* is zero, or
  - any of the leading dimensions is invalid;
  - the matrix sizes lead to accessing outsize of any of the buffers;
- `clAmdBlasInvalidMemObject` if either *A* or *C* object is invalid, or an image object rather than the buffer one;
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures;
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid;
- `clAmdBlasInvalidContext` if a context a passed command queue belongs to was released.

Examples

`example_cherk.cpp`.

4.13.2 Zherk

Rank-k update of a hermitian matrix with double-complex elements

Function

```
clAmdBlasStatus clAmdBlasZherk ( clAmdBlasOrder order, clAmdBlasUplo uplo,
clAmdBlasTranspose transA, size_t N, size_t K, double alpha, const cl_mem A,
size_t offa, size_t lda, double beta, cl_mem C, size_t offc, size_t ldc,
cl_uint numCommandQueues, cl_command_queue * commandQueues, cl_uint
numEventsInWaitList, const cl_event * eventWaitList, cl_event * events )
```

Description

Rank-k update of a hermitian matrix with double-complex elements.
Rank-k updates:

- $C \leftarrow \alpha AA^H + \beta C$
- $C \leftarrow \alpha A^H A + \beta C$

where *C* is a hermitian matrix.

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix *C* being referenced.
in	*transA*	How matrix *A* is to be transposed.
in	*N*	Number of rows and columns in matrix *C*.
in	*K*	Number of columns of the matrix *A* if they are not transposed; otherwise, number of rows.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offa*	Offset in number of elements for the first element in matrix *A*.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *K* if *A* is in the row-major format; otherwise, less than *N*.
in	*beta*	The factor of matrix *C*.
in/ out	*C*	Buffer object storing matrix *C*.
in	*offc*	Offset in number of elements for the first element in matrix *C*.
in	*ldc*	Leading dimension of matrix *C*. It cannot be less than *N*.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Returns

- `clAmdBlasSuccess` on success.
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- otherwise, the same error codes as the `clAmdBlasCherk()` function.

4.14 xHER2K - HErmitian Rank-2K update to a matrix

4.14.1 Cher2k

Rank-2k update of a hermitian matrix with float-complex elements

Function

```
clAmdBlasStatus clAmdBlasCher2k ( clAmdBlasOrder order, clAmdBlasUplo
uplo, clAmdBlasTranspose trans, size_t N, size_t K, FloatComplex alpha,
const cl_mem A, size_t offa, size_t lda, const cl_mem B, size_t offb,
size_t ldb, cl_float beta, cl_mem C, size_t offc, size_t ldc, cl_uint
numCommandQueues, cl_command_queue *commandQueues, cl_uint numEventsIn-
WaitList, const cl_event *eventWaitList, cl_event *events);
```

Description

Rank-2k update of a hermitian matrix with float-complex elements.
Rank-k updates:

- $C \leftarrow \alpha AB^H + \overline{\alpha}BA^H + \beta\ C$
- $C \leftarrow \alpha A^H B + \overline{\alpha}B^H A + \beta\ C$

where *C* is a hermitian matrix.

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix *C* being referenced.
in	*transA*	How matrix *A* is to be transposed.
in	*N*	Number of rows and columns in matrix *C*.
in	*K*	Number of columns of the matrix *A* if they are not transposed; otherwise, number of rows.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offa*	Offset in number of elements for the first element in matrix *A*.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *K* if *A* is in the row-major format; otherwise, less than *N*. Vice-versa for transpose case.
in	*B*	Buffer object storing the matrix *B*.
in	*offb*	Offset in number of elements for the first element in matrix *B*.
in	*ldb*	Leading dimension of matrix *B*. It cannot be less than *K* if *B* is in the row-major format; otherwise, less than *N*. Vice-versa for transpose case
in	*beta*	The factor of matrix *C.*
in/ out	*C*	Buffer object storing matrix *C*.
in	*offc*	Offset in number of elements for the first element in matrix *C*.
in	*ldc*	Leading dimension of matrix *C*. It cannot be less than *N*.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in/ out	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Rank-2k update of a hermitian matrix with float-complex elements (Cont.)

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasNotInitialized` if `clAmdBlasSetup`() was not called;
- `clAmdBlasInvalidValue` if invalid parameters are passed:

- either *N* or *K* is zero, or
- any of the leading dimensions is invalid;
- the matrix sizes lead to accessing outsize of any of the buffers;

- `clAmdBlasInvalidMemObject` if either *A*, *B*, or *C* object is invalid, or an image object rather than the buffer one;
- `clAmdBlasOutOfHostMemory` if the library can't allocate memory for internal structures;
- `clAmdBlasInvalidCommandQueue` if the passed command queue is invalid;
- `clAmdBlasInvalidContext` if a context a passed command queue belongs to was released.

4.14.2 Zher2k

Rank-k update of a hermitian matrix with double-complex elements

Function

```
clAmdBlasStatus clAmdBlasZher2k( clAmdBlasOrder order, clAmdBlasUplo
uplo, clAmdBlasTranspose trans, size_t N, size_t K, DoubleComplex
alpha, const cl_mem A, size_t offa, size_t lda, const cl_mem B, size_t
offb, size_t ldb, cl_double beta, cl_mem C, size_t offc, size_t ldc,
cl_uint numCommandQueues, cl_command_queue *commandQueues, cl_uint
numEventsInWaitList, const cl_event *eventWaitList, cl_event *events);
```

Description

Rank-k update of a hermitian matrix with double-complex elements.
Rank-k updates:

- $C \leftarrow \alpha AB^H + \overline{\alpha}BA^H + \beta C$
- $C \leftarrow \alpha A^H HB + \overline{\alpha}B^H A + \beta C$

where *C* is a hermitian matrix.

Parameters

in	*order*	Row/column order.
in	*uplo*	The triangle in matrix *C* being referenced.
in	*transA*	How matrix *A* is to be transposed.
in	*N*	Number of rows and columns in matrix *C*.
in	*K*	Number of columns of the matrix *A* if they are not transposed; otherwise, number of rows.
in	*alpha*	The factor of matrix *A*.
in	*A*	Buffer object storing matrix *A*.
in	*offa*	Offset in number of elements for the first element in matrix *A*.
in	*lda*	Leading dimension of matrix *A*. It cannot be less than *K* if *A* is in the row-major format; otherwise, less than *N*. Vice-versa for transpose case.
in	*B*	Buffer object storing the matrix *B*.
in	*offb*	Offset in number of elements for the first element in matrix *B*.
in	*ldb*	Leading dimension of matrix *B*. It cannot be less than K if B is in the row-major format; otherwise, less than *N*. Vice-versa for the transpose case.
in	*beta*	The factor of matrix *C.*
in/ out	*C*	Buffer object storing matrix *C*.
in	*offc*	Offset in number of elements for the first element in matrix *C*.
in	*ldc*	Leading dimension of matrix *C*. It cannot be less than *N*.
in	*numCommandQueues*	Number of OpenCL command queues in which to do the task.
in/ out	*commandQueues*	OpenCL command queues.
in	*numEventsInWaitList*	Number of events in the event wait list.
in	*eventWaitList*	Event wait list.
out	*events*	Event objects per each command queue that identify a particular kernel execution instance.

Rank-k update of a hermitian matrix with double-complex elements (Cont.)

Returns

- `clAmdBlasSuccess` on success;
- `clAmdBlasInvalidDevice` if a target device does not support floating point arithmetic with double precision;
- otherwise, he same error codes as the `clAmdBlasCher2k()` function.