

New in 6.1

New features

Concurrent debugging of multiple kernels is now enabled. Note that timing and thread switches behavior may vary greatly due to debugger overhead. It is recommended to debug kernels separately

[6122] Error / Warning breakpoints from standalone gDEDebugger were added to the Visual Studio package. These include Break on OpenCL Errors, Break on Memory Leaks, Break on Detected Errors, and more

Feature improvements and changes

gDEDebugger no longer requires online registration to obtain a license, allowing users inside LANs and behind firewalls a simple activation

Greatly improved overall kernel debugging performance, especially in scenarios where large sections of code are being skipped (Step over, run to breakpoint)

Kernel debugging robustness and reliability were greatly improved

Overall application stability was improved both for the client (Visual Studio) and server (debugged application) sides

All kernel optimizations are now disabled by gDEDebugger, greatly increasing debugging reliability and robustness. All private variables and types are now shown properly

The gDEDebugger tutorial is now packaged with the gDEDebugger installer. Access through gDEDebugger > Help > View Tutorial

[6895] Improved support of OpenGL 4.1

Viewing Images, buffers and textures is now enabled while kernel debugging. Note that OpenCL memory object values will be a snapshot of the original values and will not update until after the kernel run is finished

[7361, 7163] C# and VB.NET project types are now supported for debugging (requires setting the debugging option to "Start external program". Note that OpenCL and OpenGL must still be called from native code, though managed code call stack frames are supported

[7377] When a kernel source breakpoint is hit, if the last selected work item is not valid, a valid work item is chosen instead

[6967] The Kernel Function Breakpoints tab in the Breakpoints dialog now remembers the kernel names from the last debugging session

Fixed in 6.1

Visibility of Variables and Data

[6619, 7526, 7495] Unable to see contents of structs in Locals/Watch/MultiWatch views

[6616] gDEDebugger did not handle some vector data types correctly

[6575] Limitations causing missing or incorrect variable values during debugging

[7084] Vector and structure variable type members were not shown correctly in the MultiWatch views

[6659] When dereferencing pointers to compound types (vector, structs) in the watch view, the members were not expandable

[7447, 7449, 7364, 7237, 7233, 7446, 7176, 7282, 7177, 7462, 7448, 7309, 7374, 7226] Display issues for large data types

[6793] Variable display issues in kernels that use double-precision floating-point data

[7105] Global-scope variables in OpenCL program (defined outside functions) were not displayed in gDEDebugger

[7030] Variables would sometimes appear multiple times in the Locals view

[7063] Changing the selected work item wouldn't always update the Locals and Watch views properly

[6838] Texture export was not possible during kernel debugging

[7658] OpenCL event object indices were shown incorrectly in the API Calls History and Properties views

[7158, 7001, 7027, 7034, 7473, 7345] General variable display bugs

Debug and Program Control

[6727] Attempting to concurrently debug multiple kernels would cause unexpected behavior

[6720, 6755, 6722] Breaking on a function call lines was not possible. Step Over would sometimes step into the function called

[6453] Issuing Step Over commands could take a long time (especially when stepping over functions, loops, etc.)

[6785, 6808] Breakpoints were disabled in a confusing manner for some kernels.

[6451, 6452, 6718, 6862, 6522, 6618, 6592, 6568, 6475, 6527] Stepping through kernel code would sometimes cause the program counter to go back a few lines, as with debugging optimized code

[6715] Stepping into inline functions was not possible in some cases

[7095] Variables would sometimes disappear after passing a certain point in a function or code scope

[6524, 7293] When debugging a kernel with function calls, the variables from each call stack frame were shown in all stack frames, instead of only the appropriate ones

[6585] Stepping through switch statements would jump up 2 cases

[6835] When stepping over or out in kernel debugging, breakpoints in the stepped over code were ignored

[7097] A message saying the kernel cannot be debugged was sometimes displayed when attempting to enter kernel debugging

[7352, 6966, 7503] Disabling a kernel source breakpoint or kernel function breakpoint while kernel debugging is running could cause Visual Studio to hang

[7527] Structure and vector types included in structures would display incorrect values

[7474, 7033, 6880, 7260, 7217, 7195, 7218, 7134, 6884, 7342] General program control issues

OpenCL / OpenGL interoperability

[7167] Functions called internally by interop contexts would be displayed in the API Calls History view and would respond to breakpoints and step commands

[7274] General OpenCL / OpenGL interoperability issues

IDE / UI

[6665] The gDEDebugger Properties view would sometimes display information for the incorrect object

[6802] gDEDebugger OpenCL Engine / OpenGL Engine checkboxes did not block CL / GL functionality

[6674] Statistics view deprecated functions message was ambiguous

[6680] Stopping the debugged application could cause the solution window to be closed

[6682] MultiWatch view - adjusting the variable value active range did not work

[7011] Adding breakpoints through the Visual Studio breakpoints window would sometimes cause IDE crashes. gDEDebugger would also sometimes cause a crash when Visual Studio was exited

[7436, 6886, 7198, 7306] Debugging projects configured with relative paths from the solution would not work correctly and could cause crashes

[6840] The debug environment set in the project "Debugging" properties was ignored

[6685, 6661] Issues with the Statistics and Memory Chart views.

[6845, 6964] Kernels with function attributes and kernels mentioned in comments were not correctly pointed to by double-clicking in the gDEDebugger Explorer

[7307, 7403, 7497] Attempting to use "Attach to Process" or "Show Disassembly" with the gDEDebugger debug engine could cause Visual Studio to crash

[6901] Adding breakpoints with no solution open could cause Visual Studio to crash

[6832] The modules view would display incorrect module load timestamps

[6955, 6929, 7505, 7091, 7006, 6924, 7606, 7602, 7262, 6825, 6984, 6963, 6953, 6968, 7272, 7187, 6947, 6951] General UI and IDE fixes

General / Stability

[6836] Releasing OpenCL objects in certain order could cause the gDEBugger OpenCL server to crash

[7349] Applications that enqueue many small queue commands would sometimes cause IDE hangs

[7059] Viewing texture functions in the API Calls History view could cause Visual Studio to crash

[7031, 7290, 7101] The gDEBugger OpenCL and OpenGL servers would sometimes crash when checking for memory leaks on application exit

[7349, 7506] Attempting to debug kernels in a program created with `clCreateProgramWithBinary` could cause the gDEBugger OpenCL server to crash

[7573] The gDEBugger Send Error Report dialog would sometimes be shown in inappropriate scenarios

[7482, 7481, 7475, 7525, 7496, 7480] Stability issues with multi-GPU configurations

[7387, 7388, 7386, 7371, 6944] Stability issues with E-series APUs

[7093, 7359, 7400, 7376, 7617, 7535, 6912, 7657, 7683, 7684, 7039, 7657, 7469, 7632, 7451, 7356, 7211, 7169, 7009, 7254, 7247, 6882, 6873, 6057] General / stability issues in the OpenCL server

[7652, 7663, 6868, 6815, 7433, 7032, 7379, 7339, 7125, 6881, 7115, 7166, 7104, 7320] General / stability issues in the OpenGL server

[7675, 7529, 7666, 7656, 7502, 7209, 7210, 7392, 7098, 7194, 7214] General / stability issues in the gDEBugger servers

[6883, 7685, 7677, 6888, 7415, 6919, 7577, 7570, 7045, 7338, 7113, 7172, 7253, 7219, 7300, 7340, 7119, 7046, 7223, 7133, 7332, 6872, 7185, 6957, 6942, 6978, 7452, 6859, 6934, 6991, 7331, 6877, 6926, 6681, 6762, 6839, 6795, 6637, 7151] General / stability issues in the gDEBugger client (Visual Studio package)

[7528, 7443, 7189, 7486, 6627] Issues related to the gDEBugger Teapot sample application

Fixed several memory leaks in the gDEBugger client and server

Known Issues

Behavior to watch for:

gDEDebugger can hang on display driver timeout.

Introducing a debugger places increased performance demands upon a system, possibly causing applications being debugged with the gDEDebugger Visual Studio extension to encounter display driver resets. This is a function of the TDR mechanism in Windows 7 (Timeout Detection and Recovery). When this happens, the gDEDebugger session may appear to hang; this is due to the complete reset of the display driver by the operating system, with accompanying loss of context. It will also manifest as an event of type Display in the system event log, with event id: 4101.

More details and information regarding possible workarounds are available at:

<http://msdn.microsoft.com/en-us/windows/hardware/gg487368>

High Priority defects:

[6663] gDEDebugger does not currently support shared OpenGL/OpenCL memory objects

[7060] If the OpenCL Extension `cl_amd_printf` is enabled and `printf` is used in a kernel, gDEDebugger will not compile the kernel code

[7094] Some kernels may abnormally terminate kernel debugging. The kernel code will still be executed, but it will not be debuggable past a certain point

[7341] Compiler issues can cause non-sequential stepping in a loop with an early termination condition

[7221] Due to issues in how the OpenCL compiler handles struct arguments, kernels with struct arguments may not step correctly through the kernel source

[7310] When stepping through kernel sources with a do-while loop, when stepping from the previous statement to the do statement, the next statement is set to the end of the do block rather than the beginning, and the next step returns the next statement pointer to the correct location

[7353] gDEDebugger incorrectly handles declarations of the form "`__local [int or float] <variable name>`" incorrectly. The Locals window always shows these variables to be pointers to variables, rather than the variables themselves

[7382] If a kernel argument is declared `__volatile`, gDEDebugger does not show the variable in the Locals view

OpenCL kernels that use local memory unsafely (wrong use of barriers or conflicting writes) can result in undefined debugger behavior

Atomic operations that return a value are not currently supported. Affected extensions include `cl_khr_int64_base_atomics`, `cl_khr_int64_extended_atomics`, `cl_khr_global_int32_base_atomics`, `cl_khr_global_int32_extended_atomics`, `cl_khr_local_int32_base_atomics`, `cl_khr_local_int32_extended_atomics`, `cl_ext_atomic_counters_32` and `cl_ext_atomic_counters_64`

Low Priority defects

[6665] gDEDebugger properties initial values defaults to current object selected by gDEDebugger Explorer
When gDEDebugger an OpenCL or OpenGL API breakpoint, the properties displayed are for whichever object is currently selected in the gDEDebugger Explorer view, rather than for the API function. To display this information, open the gDEDebugger Function Calls History, scroll to the bottom, and select the last item (which should be the API function of the break)

[6666] Next and Previous buttons report "Cannot find string marker" in gDEDebugger Function Call History view. String marker is an extension supplied by gDEDebugger that enables the user to add strings to the API calls history log and HTML log file. If the debugged application does not use this extension, the search does not find any string marker. This message will be improved in a future release

[6713] During Kernel Debugging, printing variables via command window hangs

[6788] When debugging with gDEDebugger, the Command Window and Disassembly view are not supported

[6982] When there is more than one GL Context, the gDEDebugger Memory window will not display the Object Creation Call Stack for GL Context 1

[7085] When a kernel argument is declared as `__read_only` or `__write_only`, the argument will not show up in the Visual Studio Locals, Autos, and Watch views, or in the gDEDebugger MultiWatch window