Qnx Momentics IDE Debugger FAQ#

Q: Why does the debugger not stop on breakpoints in my shared library code (or does not "step in" to functions from my library)?#

This generally occurs because gdb cannot load symbols for this library. To check this, open the Modules view. If your library appears in the view without a bug icon, its symbols are not loaded.

Q: Why does the debugger not load symbols for my shared library?#

First, it needs to find this library in your shared library path on the host. You usually have to explicitly specify this in the Shared Libraries tab in the Debug tab of the launch configuration. Second, the library file name must be the same as the so name with "lib" prefix. You can check the so name if you open Properties view for the library (.so file) or open it in the Binary editor. For example, if your so name is aaa.so.1 and your library name is libaaa.so, gdb will be unable to match the two, because of the extra version number. To avoid this problem when debugging, do not use so version number when you generate the so name for your library. Third, the library has to be compiled with debug information.

Q: I tried to launch a debug session from IDE and I get an error "Target is not responding (time out)". Why am I unable to debug?#

Most likely when you created an image for the target board you did not include `pdebug` program in /usr/bin. This binary is required to be on the target for remote debugging. Also make sure that the `qconn` process has permissions to run it.

Q: I tried to launch a debug session from IDE but it does not break on main, even looks like program is running. After short time IDE shows error "Launch time out" and I cannot debug.#

If you have a lot of sources in your project this may cause some gdb misbehavior while IDE trying to set search paths. If you compile on the same machine as you run it (same host) you can use this workaround: in launch configuration open Source tab, delete "Default" source lookup entry and add "Absolute Path". That should solve it.

Q: Can I use the gdb command line console, when IDE is missing functionality provided by gdb?#

Yes, gdb console is provided and will redirect user input to the gdb command interpreter during a debugging session. To access the console, open Console view from Windows->Show View... and click on "gdb target" of current debugging session in Debug view. Optionally, you may click on Display Selected Console button (looks like a blue monitor) on a Console view and pick gdb console from the drop down list. To execute gdb commands, simply type the command in the console. For example, `show version`. An example of such functionality would be setting address breakpoints and catchpoints.

Q: How can I attach the debugger to a running process?#

First you need to create a launch configuration, select Run->Debug..., then select Attach to Process and click New button (top left). Fill in the configuration with project and binary, select target and press Debug. It will prompt you to pick a process running on selected target. Pick a process and click Ok to create the Debug session. Now you can use regular debugger commands: resume, suspend, step, etc... You can reuse same launch configuration for future runs, just re-select process id (of course the binary also has to match).

Q: Why does the debugger show a different value for my variable than printed in the console?#

Most likely you are trying to debug optimized code. This variable has been optimized as a register and the actual memory region never changes. To fix this, turn off optimization when debugging by passing option -O0 to the gcc compiler.

Q: Why does the console output of my program not match when I step through the code with the debugger?#

The console is buffered. If you are using stderr or stdout and you want to see your output immediately you have to add "\n" or flush the output. Input is always buffered, you have to press enter from console for the application to get any of the data, even if you are reading one byte.

Q: Why does my program output looks different when I am debugging vs. running it on Windows?#

When you debug on Windows, IDE cannot allocate another console for your program output so it gets mixed in with debugger data. Since it uses a specific debugger protocol, some lines of your program output can be parsed as debugger commands and won't be printed back. You can mitigate this issue by enabling verbose console for gdb in the Debug tab of the launch configuration.

Q: How can I debug a child process after fork?#

By default, when a program forks, gdb will continue to debug the parent process and the child process will run unimpeded. If you want to follow the child process instead of the parent process, use the command `set follow-fork-mode child` from gdb console. If you want to follow both processes, you can use the following workaround. Put a call to sleep (or send SIGSTOP to itself) in the code which the child process executes after the fork. It may be useful to sleep only if a certain environment variable is set, or a certain file exists, so that the delay need not occur when you don't want to run gdb on the child. While the child is sleeping, launch another debugger session that attaches to a running process and pick the child process from the list.

Q: Why does the debugger show garbage in backtrace or stepping does not seems to match with my source code?#

When using remote debugging it often happens that the binary and its libraries on the host side do not match with the binary and its target side. When using version 6.7 or greater, gdb will warn you if it thinks that some libraries don't match by printing a warning to the console. If you don't use IDE to upload your binaries and libraries, you have to take care to keep then in synch on every launch. You can check the exact files gdb is using on the host side by exploring your libraries in the Modules view.

Q: I am trying to debug simple program and it SIGSEGVs before entering main. It runs fine outside the debugger. What can I do?#

- This is usually a symptom of a library host/target mismatch. For example when you run Momentics 6.3.2 on the target and Momentics 6.4.0 on the host. When using gdb version 6.7 or greater (available from Foundry 27, independent of Momentics) it will print a warning about a library mismatch to the console.
- This can also happen if you applied libc patch on the host and not on the target (and vice versa).
- In rare cases, C++ programs can crash before entering main in static initializers of your own code or library code. In this case you can use the debugger stack trace to examine the problem.
- If you running gdb from command line the order which you upload the executable and library is important. The shared library should be uploaded first, then the executable.

- Also crash can happen if you have mismatched C++ runtime library linked or more than one linked. For example one library use one runtime and your executable uses another one.

Q: I was trying to attach to running process and I got a message "unable to access memory at ...". Why?#

This is again a symptom of binary/library mismatch. See "How to fix library mismatch".

Q: How to fix library mismatch?#

First of all it depends if this is system library such as libc.so.3 or your library. Mismatch means library that debugger loaded symbols from on the host (determined by Debug Shared Library Path) is compiled from different sources than one that is used at runtime. If this is your library - make sure you are a) copying libraries to right place on target, where your binary actually loading them (determined by LD_LIBRARY_PATH at runtime) b) the names are correct (i.e. _g usually would not work), c) debugger paths are set to correct location.

If it is libc this is a lot more difficult a) You cannot debug with 6.4 tools on 6.3.2 target for example - this would be obvious mismatch b) correct libc should be included in the target image. Just copying it over would not solve the problem because default libc is usually in read-only section, like /proc/boot, c) if (b) is not possible worse case scenario you can link libc statically -Bstatic -lc

Q: If something goes wrong but debugger diagnostics is meaningless or absent how to troubleshoot?#

Option 1
: Enable gdb verbose log (Launch config->Debugger->Verbose console mode). It will print communication between gdb and IDE and this may help you to understand the problem (Console is available when you debug the program, click on gdb target from Debug view of switch to gdb console from console view drop down).

Option 2
: Look at IDE log file, <workspace>/.metadata/.log. It contains exceptions recorded during execution. Sometimes it has meaningful information.

Option 3
: If option 1 and 2 did not help, post a message on IDE forum

---

Back