



QSpice C-Block DbgLog Class

Fast Logging of Debugging Messages To File

Document Revision: 2023.10.14

Original Release.

Document Revision: 2023.10.25

Revised to support the Microsoft Visual C/C++ compiler. DbgLog.h now redirects to DbgLog_DM.h (for DMC) and DbgLog_VC.h (for MSVC). Time format changed. Note: The demonstration code (DbgLogTest.cpp) was *not* revised to compile under MSVC.

Document Revision: 2023.11.14

Revised to support the MinGW compiler. DbgLog.h now redirects to DbgLog_MGW.h for MinGW. Note: The demonstration code (DbgLogTest.cpp) was *not* revised to compile under MinGW.

The Problem

I don't have my development environment set up to debug QSpice C-Block component DLLs. No stepping into the source code with breakpoints, variable inspection, etc. So, until I set up more capable tools, I'm back in the 1980's using the template-generated `display()` function to find errors.

However, there is a problem with the `display()` function: It has embedded delays of 60ms for each output. This is *clock time*, not simulation time. That's an additional *minute* for every 1000 messages in a simulation run. If I sprinkle the code with lots of debugging messages, this adds up in a hurry.

I'm an impatient guy. I make lots of mistakes. Time is precious.

Additionally, once all of those messages are dumped into the QSpice simulation Output window, scrolling through them is clumsy and inefficient. I could copy the entire output into an IDE editor and use the search features. Better. But that's an additional step. More wasted moments.

The Solution

My solution is a small class to send debugging messages to a log file. It's fast – at least it's much faster than `display()` since there's no 60ms delay built into each message output.

Once I've opened the debugging log, my IDE notices if it changes (i.e., I run another simulation) and loads the changed data. Sweet.

Using The DbgLog Class

The DbgLog class is simple to use:

1. Add the header file to and a class instance to your component code.

```
#include "dbglog.h"
DbgLog dbgLog("@qdebug.log", 100);
```

Change the log name if desired. The second parameter sets the maximum number of messages to log. Change it to whatever you want (-1 = no limit).

2. Where you would use the `QSpice display()` function to output debugging messages, change them to `LOG()` and/or `LOGT()` calls. (See below)
3. Compile and run the simulation.
4. Open the log and start the head-scratching ritual.

The log file will be created in the working directory. It looks something like this:

```
Simulation Time...@Line: Log Message...
0.0000000000000000000000083: Evaluation Function called, per-instance data initialized.
0.0000000000000000000000090: Evaluation Function called.
-----@0094: MaxExtStepSize() called.
0.000000000097656250000090: Evaluation Function called.
-----@0094: MaxExtStepSize() called.
0.00000000019531250000105: Trunc() called with *timestep=inf
0.0000000001953125000090: Evaluation Function called.
*** EOF ***
```

Each line is prefixed with the simulation time (`LOGT()`) or a spacer (`LOG()`) followed by “@” and the source code line number that generated the message. Last is, of course, your debugging message.

Note: The `LOGT()` version requires the time variable (“t”) to be present. If you create and call other functions, pass the variable into the function if you use `LOGT()` in the function. Otherwise, just use `LOG()`.

It's that simple. And fast.

Caveats/Notes

The code is designed to be easy to use. To keep things simple, the underlying `DbgLog` class is both declared and defined in the header file. This is bad programming style but the alternative would require users to set up a multi-compile-object project. As long as your component is in a single compilation unit (i.e., a single `*.cpp` file), this works.

I hope that you will find this useful. Please let me know of any problems or useful improvements.

```
--robert
```