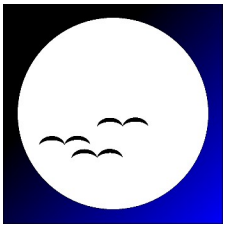


QSpice C-Block Components

Microsoft Visual Studio Configuration

Overview

For any serious QSpice C-Block component development, we need a proper development environment. In particular, we need a debugger to step into the DLL code. QSpice doesn't provide that.

This document walks through configuring the Microsoft Visual Studio 2022 Community Edition (free, see Resources below) to build and debug QSpice component DLLs.

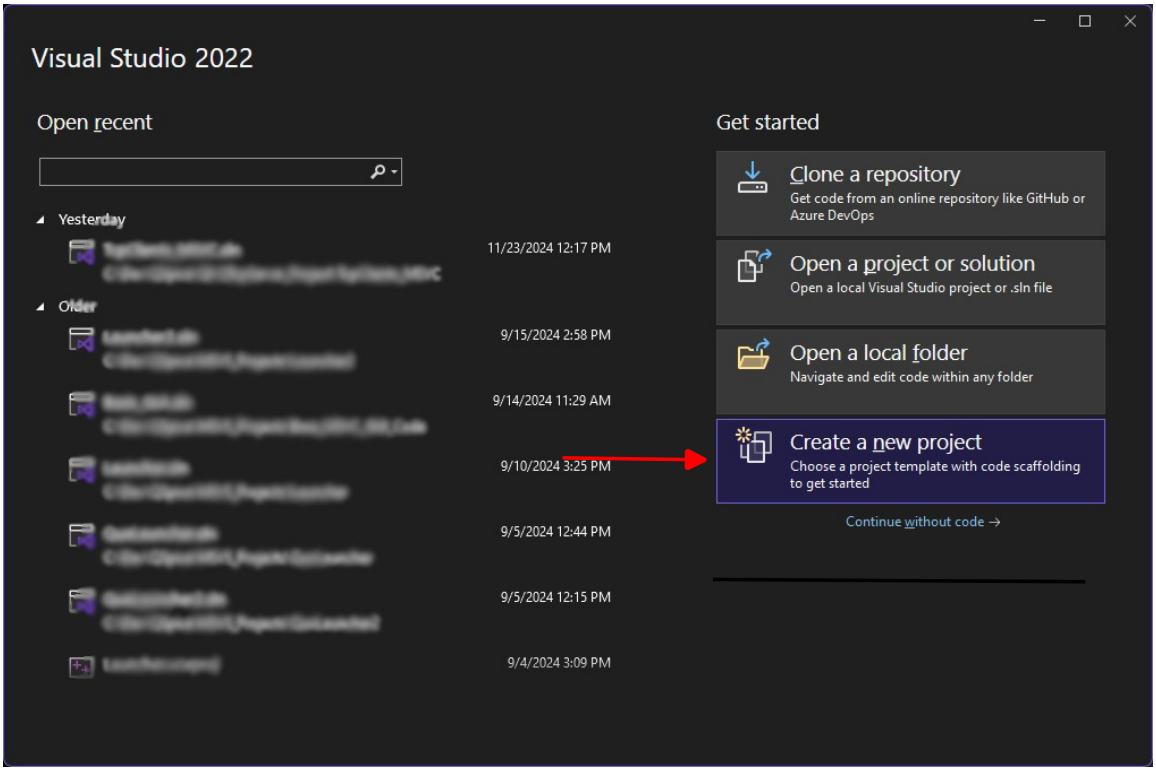
I'm covering only how to set up MSVC for a QSpice component project. I assume that:

- You have installed MSVC and configured it to compile C/C++ code.
- You are familiar with QSpice component development basics. In particular, I assume that you know how to create C-Block components in schematics and how to generate default C++ component code with the QSpice code-generator.
- You have QSpice installed in the default location (`C:\Program Files\QSpice`). If not, adjust accordingly.

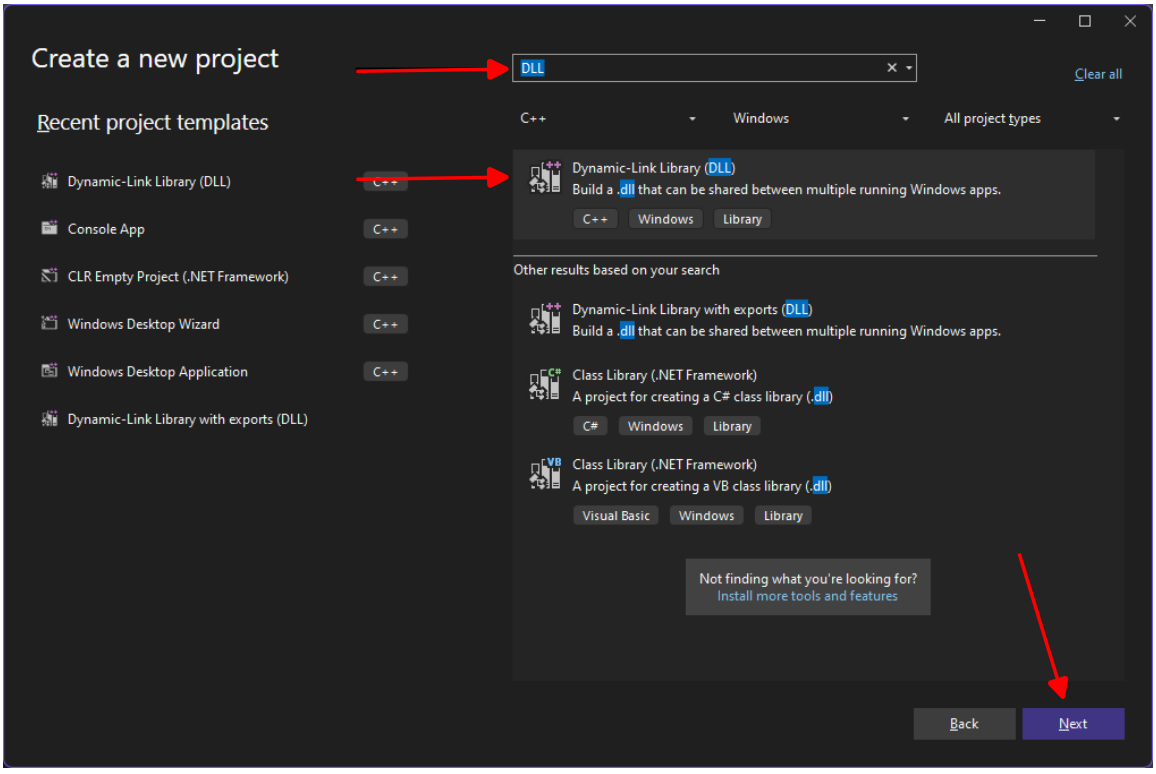
Note that there are many ways to set up/configure MSVS solutions/projects. I'm not a Visual Studio expert and what I'm presenting here is simply one configuration that works. Feel free to suggest improvements.

Step-By-Step

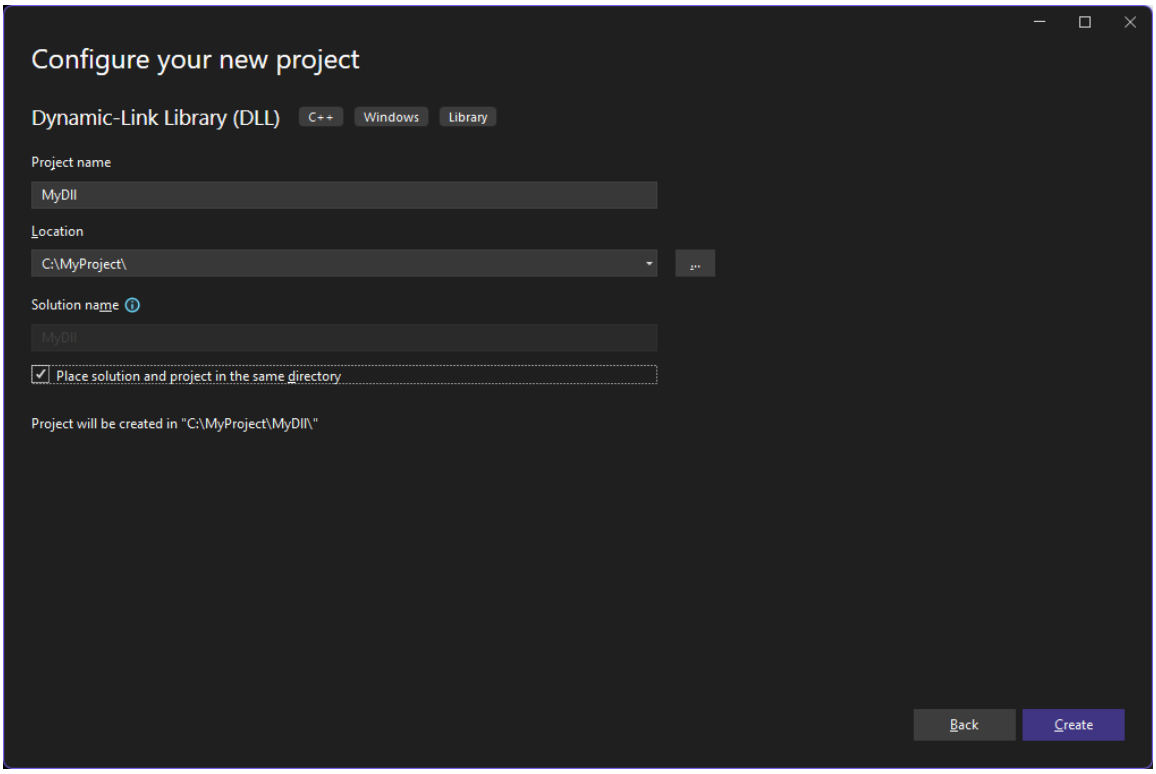
Step 1 : Open Visual Studio and create new project



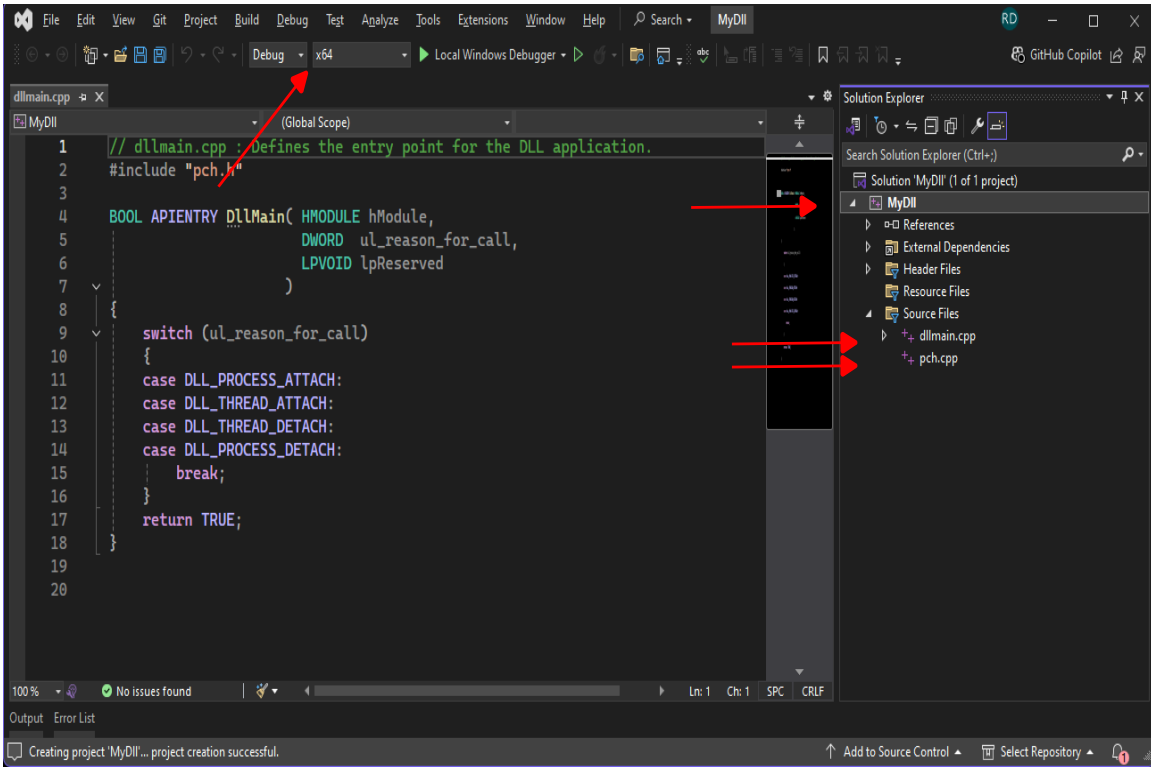
Step 2: Select the DLL template



Step 3: Set up project path, name, etc.



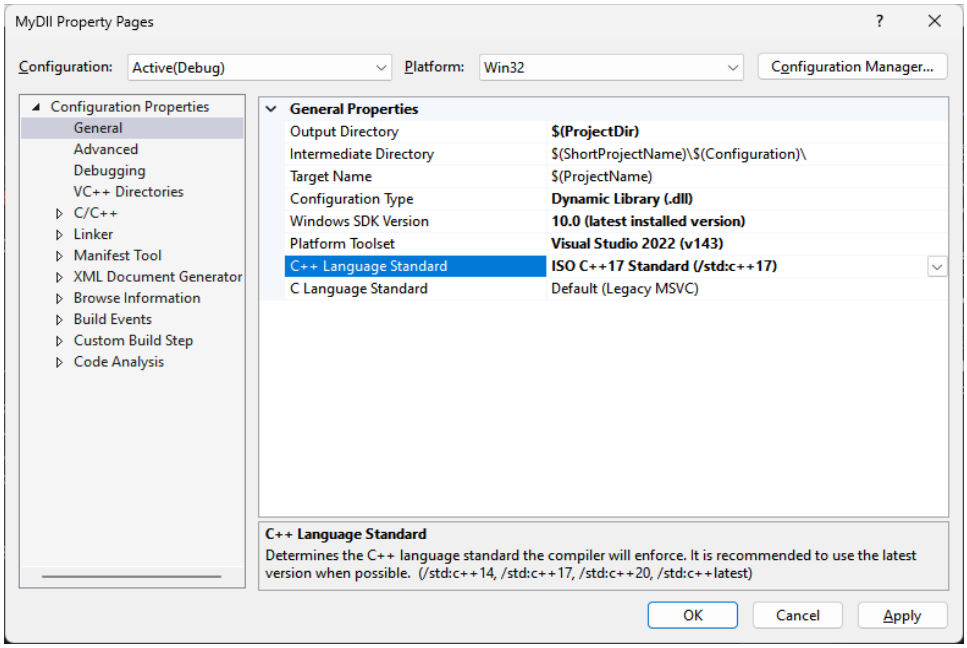
Step 4: Default MSVS DLL code generated – note stuff before changing...



At this point, your project should look like the above.

Step 5: Change General properties to match then click Apply

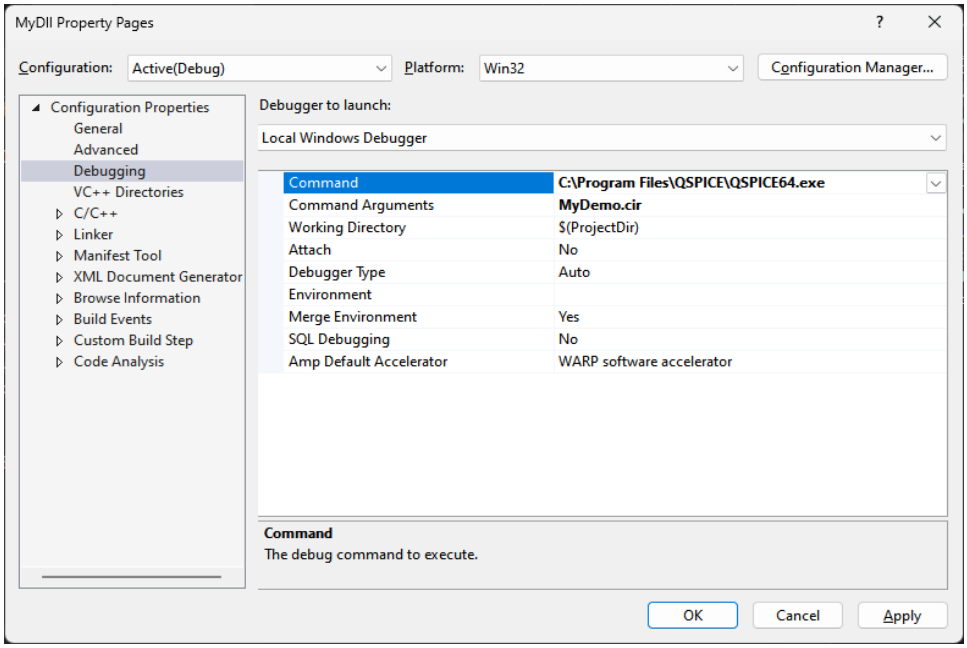
Right-click on the MyDLL project item in the Solution Explorer pane on the right. Select Properties from the pop-up menu (bottom item).



I probably should have selected “All configurations” instead of “Active (Debug).” Same for all of the below. (I’m too lazy to do all of the image captures again.)

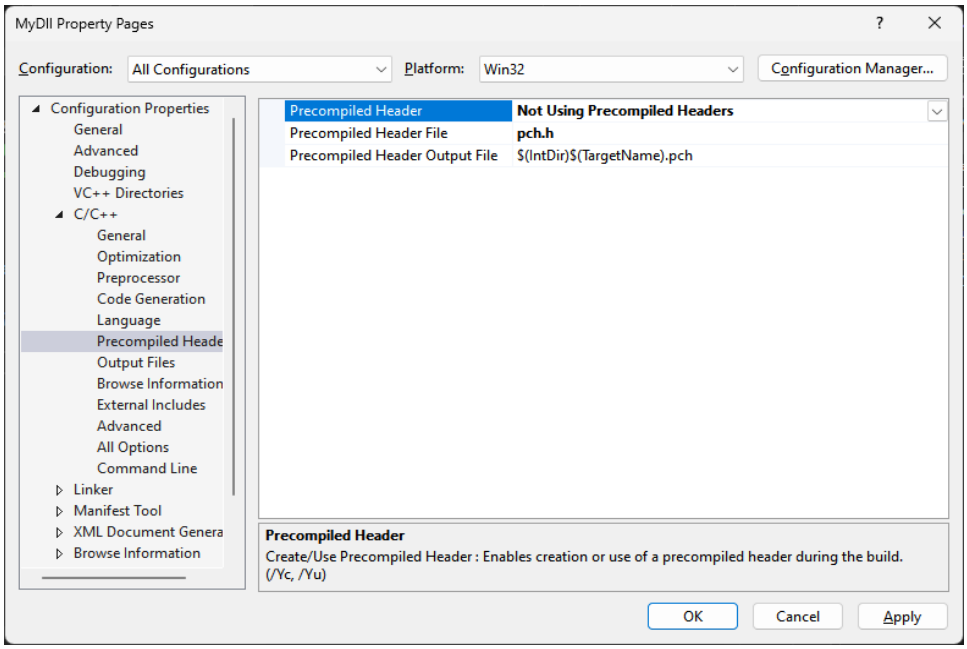
You can choose a different ISO standard if preferred.

Step 6: Change Debugging properties to match and then click Apply



Note that, when we run the debugger, it will open `QSPICE64.exe` and pass `MyDemo.cir` as an argument. You can use `QSPICE80.exe` if you require the higher-precision math. We’ll discuss the `*.cir` bit shortly.

Step 7: Change C/C++ | Precompiled Headers properties to match then click Apply and Close



If you want to use pre-compiled headers, you can but we're not doing that in this guide.

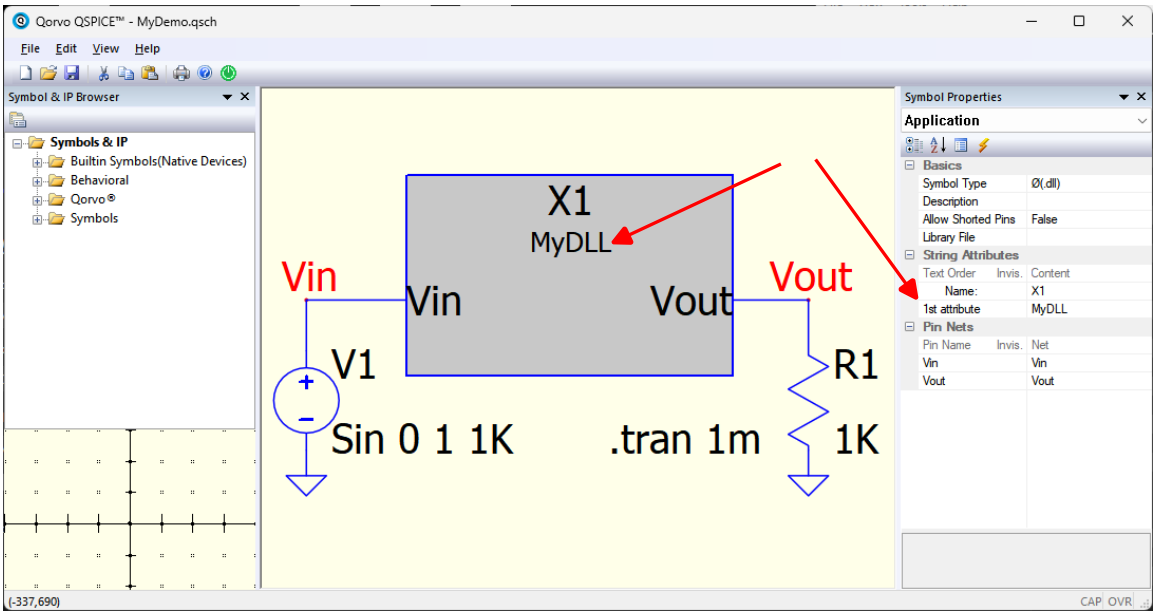
Step 8: Create the QSpice schematic, component code, and netlist

Now we need to:

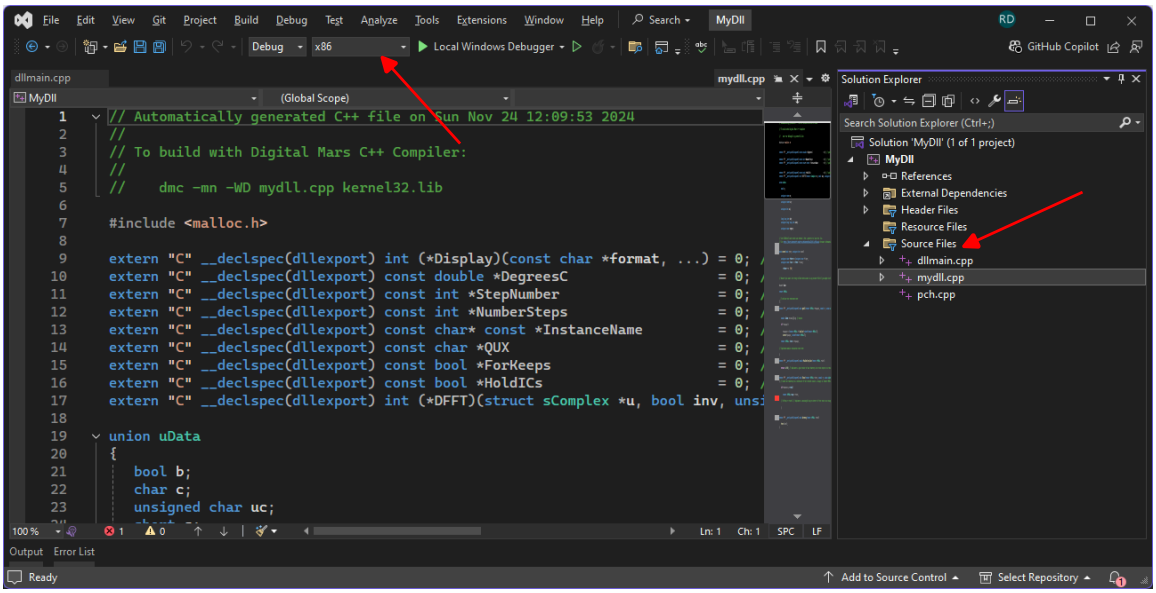
- Copy the provided QSpice schematic (`MyDemo.qsch`) into the MyDLL project folder.
- Generate the initial `MyDLL` component code, `MyDLL.cpp` with QSpice. Select all code-generation options.
- Generate the `MyDemo.cir` netlist. In QSpice, View | Netlist. Then, from the Netlist window, File | Save As, `MyDemo.cir`. Again, I'll come back to this in a bit.

Note: The component DLL name (the 1st String Attribute) must match the MSVS project name (`MyDLL`). That is, when MSVS generates the DLL, the DLL name will be based on the project name. QSpice will try to load a DLL based on the schematic component DLL name. If they don't match, QSpice won't find the `*.dll`.

Confusing? Yes. Can we name things differently and get things to work? Probably. But that's more complication that I have the patience to deal with in this guide.



Step 9: Change the MSVS project target files and build type

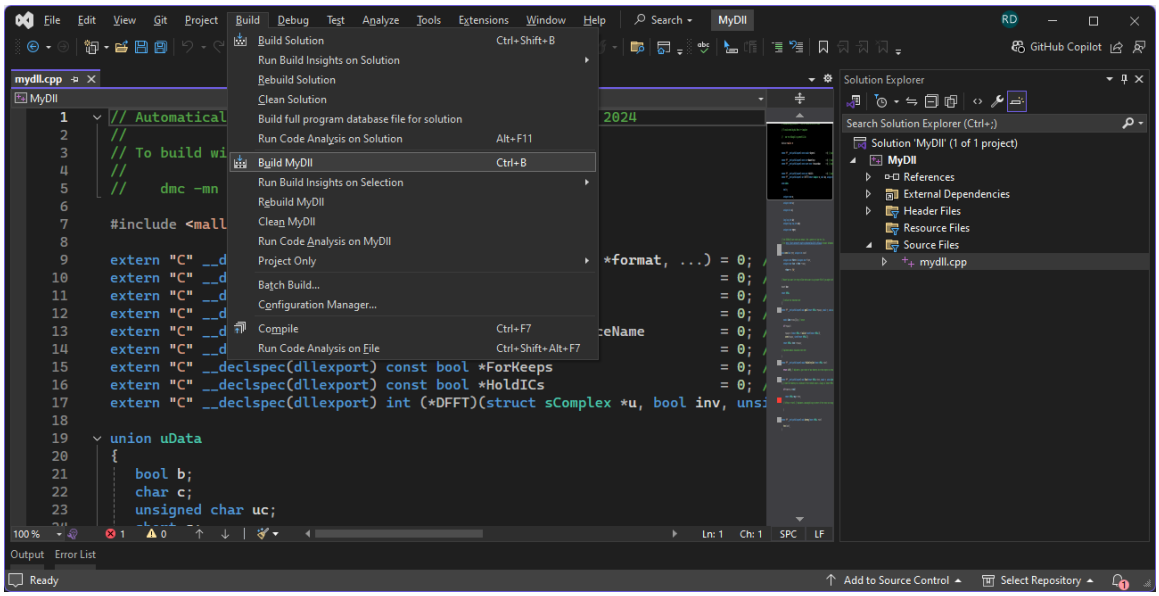


Change the build type from `x64` to `x86` (top).

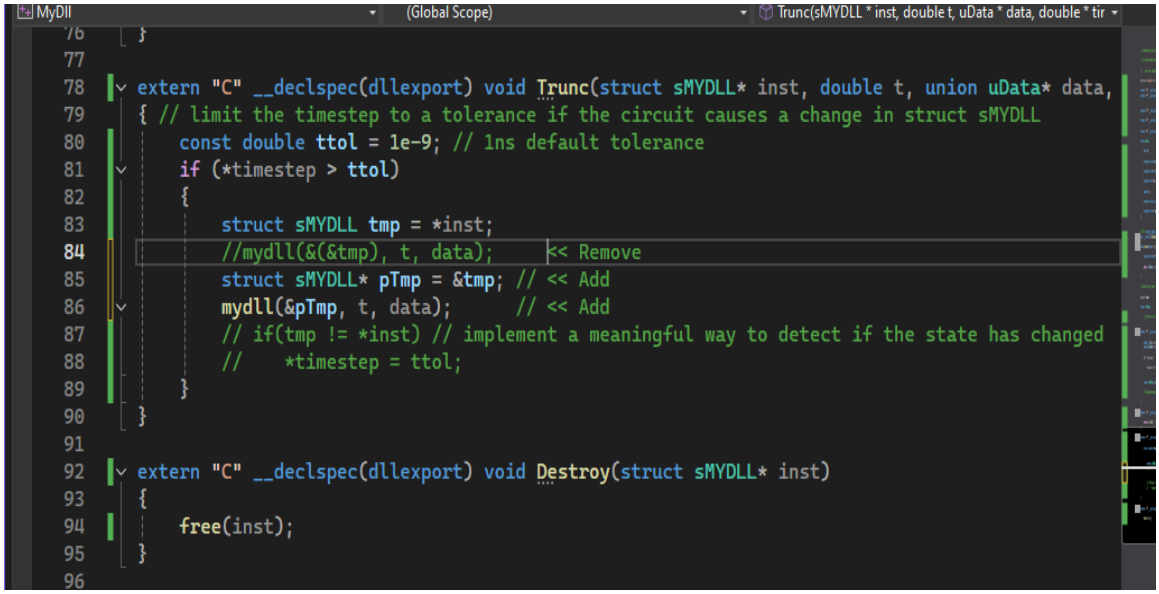
In the Solution Explorer, remove `dllmain.cpp` and `pch.cpp`. Right-click on each, select Remove. Remove `pch.h` from the Header Files (not shown, same process).

Add the `mydll.cpp` target. Right-click on Source Files, Add | Existing Item, and then select `mydll.cpp`.

Step 10: Build the debug DLL

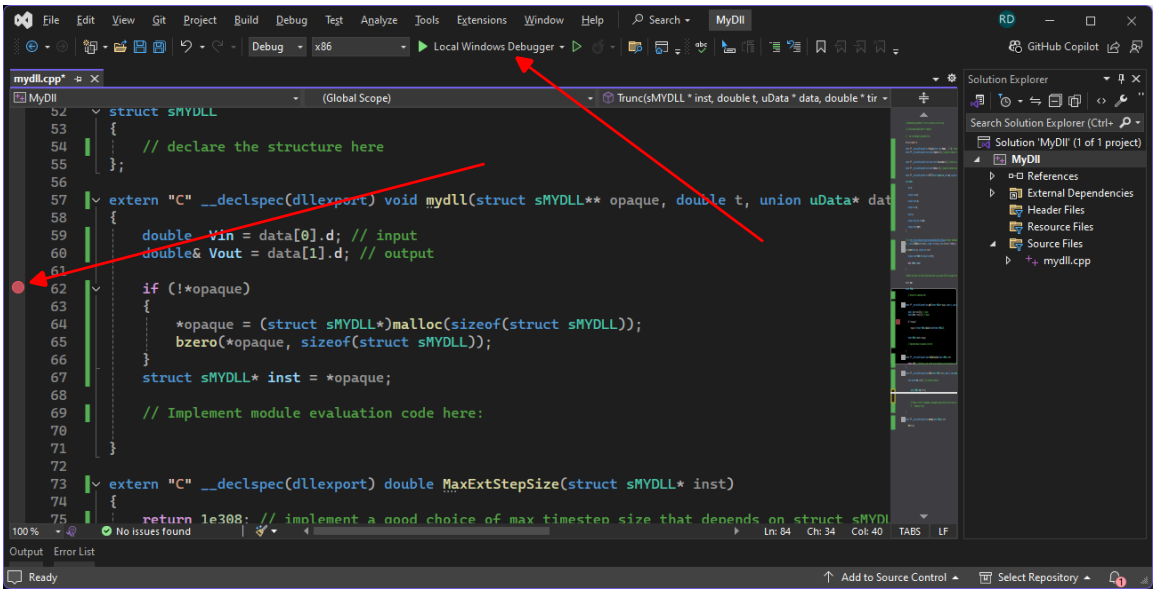


From the top menu, Build | Build MyDll. As mentioned, you'll get an error in the `Trunc()` function. Here's the fix:

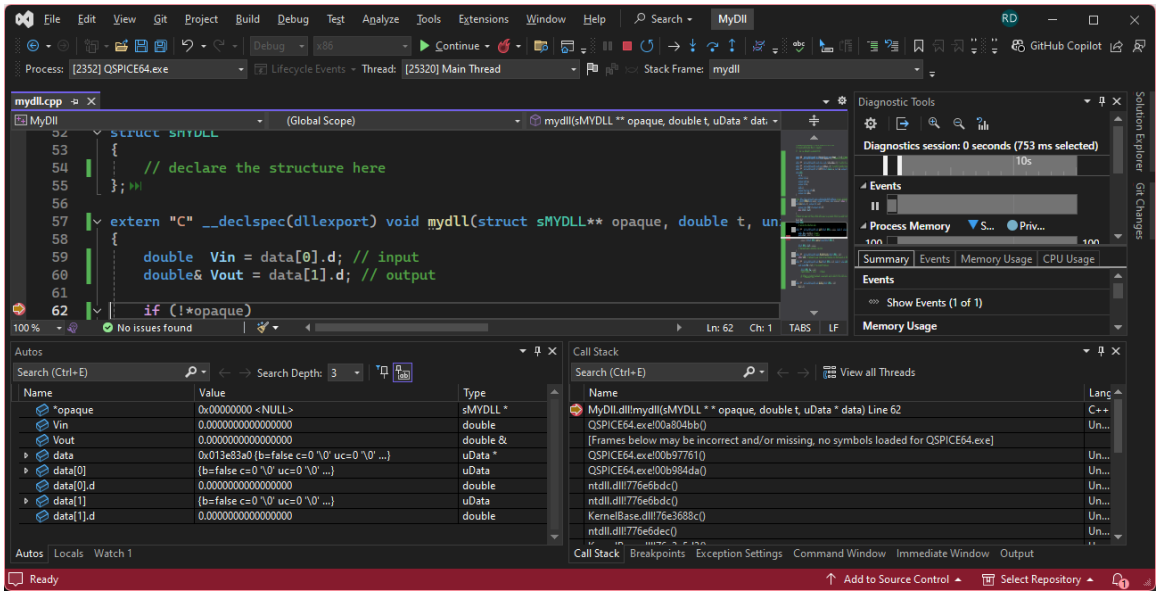


Do another build. Hopefully, there are no errors.

Step 11: Set a breakpoint and try a debug run



Click on the leftmost column of an executable line of code to set a breakpoint. Then click Local Windows Debugger. If all went as expected, the debugger window will open stopped at the breakpoint. It looks like this:



With a bit of luck, everything worked. If you find a problem with my instructions, please let me know.

About That *.cir Netlist Stuff

To state the obvious, when we run the debugger, we're executing the simulator (`QSPICE64.exe`) for the `MyDemo.cir` netlist. *If you modify the schematic, you **must** regenerate the *.cir file before running the debugger again.*

Manually generating the netlist is, of course, annoying and easily forgotten. It is possible to generate the netlist from the command line with the following incantation:

```
"C:\Program Files\QSPICE\QUX.exe" -Netlist MyDemo.qsch
```

Note: "`-Netlist`" is case-sensitive. "`-netlist`" doesn't work.

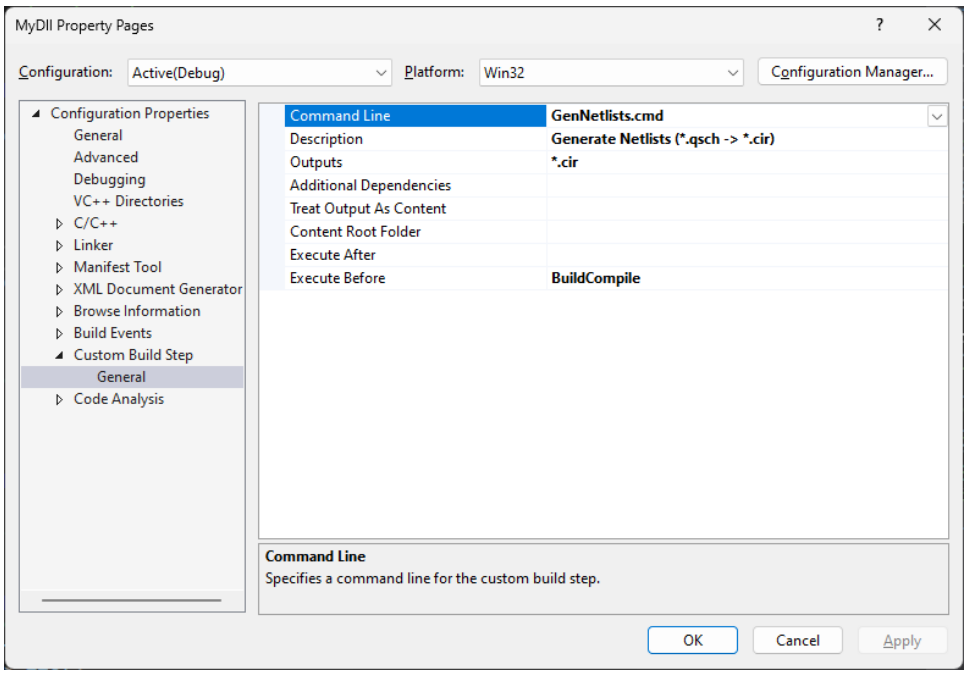
We could, in theory, create a custom dependency in MSVS that regenerates the `*.cir` netlist if the `*.qsch` is newer. I've not gotten that far. Feel free to solve this and send me the solution. In fact, that would be super.

Update: More About That *.cir Netlist Stuff

OK, I have a solution, albeit possibly imperfect….

I've created a batch file, `GenNetlists.cmd`, that does the above for all `*.qsch` files in the project folder. Save it in the project folder. (If you installed QSpice somewhere other than the default location, edit the batch file accordingly.)

We need to add the batch file as a custom build step. In the Solution Explorer, right-click on the project (MyDLL) and select Properties. In the Custom Build Step | General settings, make these changes:



The `*.cir` file(s) will now be (re)created every time that MSVS (re)builds the component DLL. As far as I can tell, this happens when the debugger is launched even if the DLL isn't changed. That is, it may needlessly recreate the netlist but it will always be up-to-date when debugging.

I called this solution "possibly imperfect" above. By that I mean that:

- The `*.cir` file is recreated even if the `*.qsch` hasn't changed. This is quite fast so it's not really a problem.
- The solution is a batch file. Surely this could be moved into MSVS and proper dependencies configured.

On the other hand, we don’t have the “forgot to create/update the `*.cir` file manually in QSpice” before debugging issue….

Anyway, if you find a more elegant solution, please let me know and I’ll update this guide.

Resources

The most current version of this document and the related files are available on my GitHub QSpice repository. The repo includes many more documents that should be useful for QSpice component developers.

You can get the free Community Edition of Microsoft Visual Studio 2022 here: https://visualstudio.microsoft.com/downloads/.

Conclusion

I hope that you find the information useful. Please let me know if you find problems or have suggestions for improving this documentation.

--robert