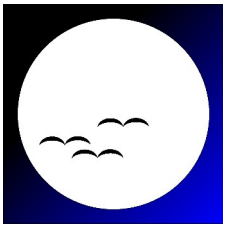

QTcpServer Project Overview

A Client/Server Framework for QSpice C-Block Components

Author's Note – 2024.12.31

This is one of those projects that started with a simple problem and ended with an unexpectedly complicated solution. As I was finalizing the documentation for this project, a QSpice enhancement provided a simpler, more straight-forward solution for the original issue[1].

So, you might ask why I am sharing this project now that the problem is solved more easily?

Well, this project is a TCP-based client/server framework. It's easy to implement C-Block component code that connects to and sends requests to the server. And the server code could do ***anything***. *When I started, I had bigger things in mind.*

We could, for example, implement a micro-controller emulator, maybe rework an existing open source project to embed it in the server. Or the server could connect to existing online micro-controller emulator. Or the server could connect to a USB micro-controller hardware debugger. For all of these, the C-Block component would send input pin states to the server, the server would return output pin states, and the component would set those output pin states in the simulation. The server does the heavy lifting and the client component code remains simple. Call me a dreamer.

Anyway, while the project doesn't solve any major problems in its current form, maybe it will prove a useful starting point for something more interesting. And that's why I'm sharing it.

Now back to the incomplete documentation….

Background

A few months ago, QSpice forum member @KSKelvin contacted me with an issue. He was trying to launch a QSpice Waveform viewer from within a C-Block component. He could successfully open the viewer but the simulation wouldn't actually end until the viewer was closed.

That seemed to be a pretty simple problem. We just need to launch a separate process for the viewer. There are many WinAPI calls to do that – `system()`, `exec()`, `spawn()`, `ShellExecute()`,

1 The 2024.12.28 QSpice release added a new ".system" command. Per the help documentation, "*The .system allows you to execute commands once the simulation completes. These will be executed as if you had typed them at the DOS C:\> prompt.*" Thus, you can execute any program or batch command after a simulation runs. If you want to get fancy and let a C-Block control what is executed, you could add "*.system MyBatchFile.bat*" to the schematic, write/modify the contents of the batch file from within C-Block code, and the batch file will be executed when the simulation completes.

`CreateProcess()` – and I tried them all. No joy. The simulation wouldn't completely end until the viewer window was closed or the window would be closed when the simulation ended. Unsatisfactory.

The only solution seemed to be to have a separate process to do the actual launching of the viewer. So I wrote a server. The QSpice component DLL client asks the server to launch the viewer and it runs in the server process. Problem solved.

But, honestly, this server solution is overkill if launching a Waveform viewer was all that it could do. Think of this as a client/server framework that could be expanded to do much more….

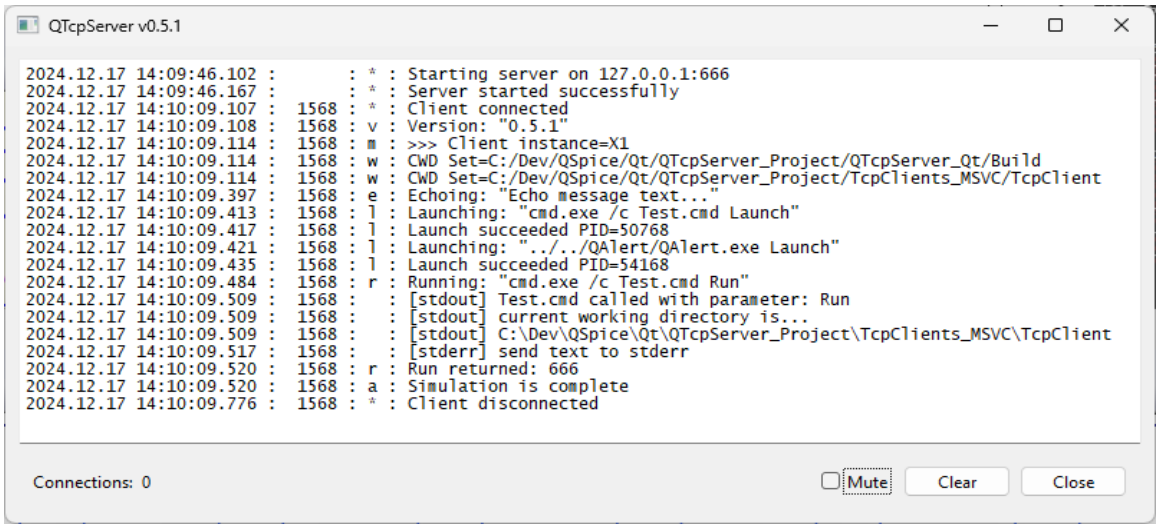

Figure 1: Sample QTcpServer GUI

Overview

The QTcpServer Project is a TCP-based client/server framework for QSpice C-Block components. The key word here is *framework*. The goal is to make it easy to use and modify for any particular purpose. The client-side code, in particular, is quite easy to use through the included high-level API.

The Server Program

When the server program is started, it:

- Opens a GUI window to display an activity log window.
- Starts the TCP server on a specified port.
- Listens for clients to connect.
- Processes client requests and returns results to the client.

The current implementation supports a limited number of client requests:

Request	Function
Alert	Pops up a notification window and beeps until the notification is dismissed. (The beep can be muted.) Useful for long-running simulations.

Request	Function
Echo	Simply echos text back to the client. Useful for testing with a TTY program.
Launch*	Runs a program (executable or batch file) and immediately returns. Useful when the program return code isn't needed.
Message	Adds a message to the server activity log.
Run*	Similar to Launch Request but waits for the program to complete and returns the program return code. Simulation execution is blocked until the program completes.
Version	Returns the server version. Useful if client requires a particular server version.
Change Working Directory (CWD)	Sets the current working directory context for launching/running programs.

* The Launch and Run commands capture `stdout` and `stderr` output in the server log window.

The Client Framework

QSpice component client code connects to the server using the Windows Sockets API. The QTcpServer framework includes low-level server messaging (`TcpSocket.h/.cpp`) and a high-level API (`TcpClient.h/.cpp`).

The high-level API makes it simple for component code to connect to the server and make requests. For example, to run a batch file called "MyScript.bat" that is located in the simulation directory, the component code might contain something similar to this:

```
#include "TcpClient.h"
…
// create a client connection to server on port 666 of this machine
TcpClient *pClient = new TcpClient("localhost", "666");
…
// set CWD to simulation directory
pClient->setCwd();

// run batch file, display result
int returnCode;
pClient->runCmd("cmd.exe /c MyScript.bat", &returnCode));
Display("MyScript.bat returned %d", returnCode);
```

In reality, it's a bit more complicated since I've omitted error-checking. But you get the idea. See the included example schematics/code for details.

Resources

The project was developed with the Qt6 Open Source toolkit and Microsoft Visual Studio 2022 Community Edition. It does NOT support the Digital Mars compiler (DMC) shipped with QSpice.

Qt Development Toolkit

Qt version 6 tools are available here: https://www.qt.io. The Open Source Development version is here: https://www.qt.io/download-open-source.

Microsoft Visual Studio Toolkit

The MSVS tools are available here: https://visualstudio.microsoft.com/downloads.

Terminal Software

Server functionality can be tested without a C-Block component using a terminal program. I use PuTTY. It is available here: https://www.putty.org.

The QTcpServer Project

The current documentation and code for this project is available here: https://github.com/robdunn4/QSpice.

Conclusion

I hope that you find this project to be useful or, at least, interesting. Please let me know if you find problems or suggestions for improving the code or this documentation.

--robert