

Re: Computer programmers' habits in electronics

Source: <http://sci.tech-archive.net/Archive/sci.electronics.design/2005-12/msg03564.html>

- *From:* Dirk Bruere at Neopax <dirk.bruere@xxxxxxxxxx>
 - *Date:* Tue, 20 Dec 2005 19:40:49 +0000
-

PeteS wrote:

Dirk Bruere at Neopax wrote:

Jim Thompson wrote:

On Tue, 20 Dec 2005 17:53:16 GMT, Ignoramus10397
<ignoramus10397@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote:

A newbie question...

As a computer programmer, I am used to programming without drawing "design diagrams", "flow charts" and other bullshit. I just start coding and try to make sure that I have some working prototype most of the time, and that I do things nicely. Usually things work out okay and programs do their job quietly, as intended.

Not doing too much "design" also helps when the purpose of the program is not quite known from the beginning, as it usually happens.

I find it very difficult to change this mindset and do any sort of diagram drawings or some such when it comes to electrics or electronics. For example, I put together a pretty intricate phase converter in the last month, for instance, with two motors, some turn

Re: Computer programmers' habits in electronics

on logic, blah blah. That seemed to work.

What I am worried about is that if I try to do something involving more than say 20 wires, I would run into a wall and that electronics is not the same as computer programming.

So, I am curious if anyone can relate and tell me either just how mandatory drawing is, and how to get accustomed to it, or how they make things without detailed plans.

i

As a circuit designer I've always liked "block" diagramming of a system before I begin, so I don't create redundant (or useless) circuit chunks.

So I find it hard to fathom how you can write software without some similar organizing scheme.

It depends.

I normally start by writing all the drivers that will actually interface with bits of hardware. That doesn't really take much 'design' in the normal sense of the word, but it gives me something to flash lights with.

Then I start at the top end and sketch out what classes/functions I need to get down to driver level.

Right now I'm into a control system being written in C++. It will probably top 3000 lines when complete, but I still do not need a formal design methodology.

I have spent a great deal of time on the control protocol that interfaces with a PC. That determines the data structures and hence classes. It's all quite straightforward.

Now, if I was having to collaborate on a bigger prog then formal methods would probably be indispensable.

Re: Computer programmers' habits in electronics

--
Dirk

The Consensus:-
The political party for the new millenium
<http://www.theconsensus.org>

The last *major* piece of software I was involved in was the diagnostics for a video-on-demand system. The diagnostics alone totaled over 800,000 lines of code written by 10+ people over some years.

Without the spec (tightly written) and a 'black box' diagram (which I use both for hardware and software, however apparently trivial the project) we would never have got it done. I also designed the diagnostics for the next gen system (massively distributed processes) and helped design the actual system hardware, which helped enormously.

That system, incidentally, was a 'massively parallel system' with up to 320 parallel processors (I love the C construct 'where(x)' :)

We wrote both the host and target code. The host code included not only task control but our own drivers onto our own boards in UltraSparcs which connected to the target. Without the spec and a diagram of what does what, we would have been lost.

That's not to say I haven't done drivers by the seat of my pants, but generally I sketch out what the driver is supposed to do so I (or others) can write the interface to it while I am still debugging the low level driver.

I've done a lot of code and hardware since then, but no code on the scale of that system. That said, doing any non-trivial project without some semblance of documentation is generally bound to have lasting consequences (the 'why is that part there? syndrome').

Depends what you mean by documentation.

I document the code with plenty of comments because I know that I will likely have to return to it in a year or two when I've forgotten everything.

Any 'real' documentation can wait.

Re: Computer programmers' habits in electronics

--
Dirk

The Consensus:-
The political party for the new millenium
<http://www.theconsensus.org>

.