The Embedded Muse 125

Editor: Jack Ganssle (jack@ganssle.com) Feb. 22, 2006

EDITOR: Jack Ganssle, jack@ganssle.com

CONTENTS:

Editor’s Notes

Last issue I ran a note about transistor-free computing using, of all things, Legos! Bob Maxwell sent in the following link: http://meccano.us/difference_engines/rde_1/ , which is a purely mechanical difference engine, very much like Babbage’s.

Then I ran across this super-cool computer built using 415 relays as the active elements: http://www.cs.pdx.edu/~harry/Relay/index.html . There’s a short video of the thing operating linked from the site; unfortunately it’s silent. I bet the clacking noise is impressive.

Denver and Dallas – Learn to develop better firmware faster in my one-day seminar in Denver on April 26 and Dallas April 28. Details at: http://www.ganssle.com/classes.htm. With luck there might be some decent skiing in the Rockies that week!

TV chef Emeril is always “kicking things up a notch” by adding garlic and other zestiness to his recipes. How are your firm’s engineering recipes? Want to kick up your development processes by more than a notch? I can present my Better Firmware Faster class at your facility. See http://www.ganssle.com/classes.htm .

Book Review – How Computer Do Math

Clive “Max” Maxfield and Alvin Brown have written a wonderful book called “How Computers Do Math” about the essential workings of computers. All of Max’s writings are entertaining and offbeat (e.g., “Bebop to the Boolean Boogie”).

The book is aimed at people starting out in computers; we embedded experts know this stuff cold. But an interested 15 year old could get truly in-depth insight into the mysteries of computing from this volume.

It’s a very readable book laid out with easy-on-the-eyes formatting and a plethora of clear illustrations. The illustration of a LIFO stack just booms clarity. Chapters start with relevant and often amusing quotes; one of my favorites is Lewis Carroll’s “The four branches of arithmetic: ambition, distraction, uglification, and derision.”

Quickly page through the book and you’ll be puzzled by its organization. The first 55 pages (out of 450) comprise its ostensible meat. The rest are labs for each chapter, a series of problems the authors pose to illustrate important concepts. They nudge you through the solutions – there are no proofs left to the confused student.

The labs are very well-written accessible activities in which the authors take the reader along hand-in-hand. They’re a bit insidious: work through them and the reader will become a reasonably competent assembly-language programmer, without realizing he’s learning one of the more difficult aspects of programming. There’s a perverse genius in covertly slipping assembly language into one’s head without pain.

The authors’ sure hands guide one along each lab, with descriptions and demonstrations till the code that’s required is almost anticlimactic: “of *course* it must be like this!”

But how is one to do a lab? You need a computer, right? Well, sure, but the authors provide a DIY Calculator on CD, an interactive and sophisticated bit of code that runs on a PC. It sports the usual display and math functions, plus its own low-level programming language. And, it’s extensible. The companion website (http://www.diycalculator.com/aboutdiy.shtml) contains plenty of downloadable extension code, plus the calculator itself. Like open source advocates they hope the community will contribute to the set of routines.

The web site also has a fabulous background to the field of computing (http://www.diycalculator.com/cool.shtml#PhyVer) that, if you’re a history buff like me, will suck you in and surely doom the schedule of whatever product you’re working on now.

Where too many computer books have a dreary chapter about number systems, "How Computers Do Math" cover the subject in an entertaining and very complete fashion. From basic binary math they go on to show how one constructs an adder out of gates. Signed, unsigned, multiplication, rounding (9 different approaches!), BCD – it's all there, and it's all extremely comprehensible.

The book is published by John Wiley & Sons, Hoboken, NJ, copyright 2005, and sells for $26 on Amazon.

Pre-Cad

A number of people have been inquiring about an article I wrote long ago about the pre-CAD days. Here it is:

Long ago, back in the 70s, there were no personal computers or similar tools we take for granted. The life of a digital designer was very different indeed.

Computer folks used the same tools as civil engineers. Offices were dominated by the drawing board, a 6 by 3 foot flat panel propped at a "comfortable" angle. Engineers lived in high chairs strangely reminiscent of Dicken's Bob Crachet, hunched over the drawing board with mechanical pencil in hand, creating wonderful silicon inventions using the same paper and lead that writers and philosophers relied on for centuries.

Drafting was a skill all mechanical engineers mastered in college, but for some reason, at the University of Maryland, it was not considered important for electronics guys (guys - all guys. There were virtually no women in the field then). Some engineers were nothing short of gifted artists, creating schematics that filled the paper in a visually pleasing way. Others, myself included, could hardly draw a straight line even with the aid of a drafting machine.

Lettering was a fine art. It's hard to imagine this now, when we select from a thousand fonts with a few mouse clicks, but neat signal names were essential. Some designers used lettering templates. I always found these too cumbersome, and so subjected all readers of my drawings to the same illegible scrawl that the nuns weekly punished me for in grade school.

Every branch of engineering had its unique set of templates. Digital designers used a handful of plastic stencils that contained the entire stock of resources we worked with: AND and OR gates, inverters and "not" circles, and, of course, various sized squares and rectangles for microprocessors, memories, and everything else. Large chips were rare then; the largest might have a whopping 64 pins, which wasn't too hard to draw.

We drew resistors, capacitors, and other components using similar stencils. With a little practice you could whip the pencil through the zigzag of a resistor template in no time, though a not-perfectly-sharp pencil always resulted in only a hazy image of the part.

Though Luddites scoff that computers let us make mistakes at new, unprecedented rates, in fact even with these crude tools we were quite competently creating errors faster than decent designs. In lieu of a "delete" key we used erasers. Erasers in every form imaginable. In fact, most of us made so many mistakes we used electric erasers, a motorized drill-like device that spun a cylinder of gum-like material. Though these quite effectively removed vast areas of pencil lead, careless use always produced a hole in the vellum. Then what? I suppose mil spec folks started over, recopying the entire drawing onto a fresh sheet of paper. Where I worked we glanced furtively over the shoulder, and rerouted signals around the hole.

Drawings produced by hand - by sloppy electronic folks - had much wider line widths and larger lettering than that we make on computer screens today. As a result the drawings were big. Though some folks used "C" size paper (17 by 22 inches) generally "D" (22 by 34) was the norm. Engineers, production people, and technicians all looked like quintessential architects, with stacks of these monsters on every flat surface. When troubleshooting a new design the first step was to position a big table next to the lab bench - just to hold the drawings.

We drew on vellum, a gauzy semi-transparent material that was pretty tough. It took repeated erasures well, though somehow I always managed to drill a couple of erasing holes through each drawing.

Today we use mostly "A" and "B" sized paper, as that's all laserprinters handle. You can fax or Xerox these drawings without trouble. An ordinary filing cabinet is the perfect storage place. Back then, every business had a "drawing room", dedicated to nothing more than storing (in cabinets called "flat files") and reproducing these huge representations of our tiny circuits. Most companies had people whose entire role was to file and copy these.

No copier then or now could handle a sheet of D sized paper. One reason we used vellum was its transparency. An engineer duplicating a drawing first put it on top of an equal-size piece of paper coated with a light-sensitive chemical, and ran it through the Ozalid machine. This beast beamed a very intense light through the vellum, exposing areas with no pencil marks. It then treated the paper with ammonia, which turned the unexposed areas blue. The ever-present smell of ammonia (and second hand cigarette smoke!) was simply a part of the engineering environment.

Once the ammonia line broke in the middle of the night. In the morning the entire building was uninhabitable. Fire department fans eventually sucked the fumes out, but we found strange chemical changes in our environment. One secretary's fake flowers changed to a wonderful purple shade. Many pictures hanging on walls now looked like 19th century daguerreotypes. And, of course, several thousand dollars of unused Ozalid paper were completely developed, bright blue reminders of the incident.

Now drawings have no value. Did you spill coffee on a schematic? Just print out a new copy. The information itself if immensely valuable, but its paper incarnation is sacrificial.

Before CAD systems the paper was the only representation of a drawing. The labor required to recreate one was so immense that it was simply inconceivable to lose or mar a schematic. It's funny to realize that one promise of computerization was the paperless office, yet what has happened is paper is now so prevalent that it's worthless. By contrast, in the olden days an original drawing was a holy relic. Copies were scarce because of the cost of duplication and the nuisance of storing the bulky papers.

The regulations said no drawing left the files unless it was being duplicated or was on an engineer's drafting board. Most of us flaunted these rules when the Ozalid machine (which required an hour to warm up) was off at night and we were engaged in a furious troubleshooting war. Living dangerously, with the original by the bench, we'd hope to be done by morning when the drawing police came in. More than one drawing went back in the files with food stains in the corners.

I suppose it goes without saying that there were no PCB software packages. Either our engineers or outside contractors, leaning over a light table for weeks on end, routed tracks by placing black tape on sheets of mylar. It was a game of chess: the best board designers developed a mental plan of attack which (they hoped) would let them get all of the tracks down without running into dead ends. How things have changed! Now it's great fun to watch the routing program automatically plopping traces down onto the virtual board, at a speed that takes my breath away. In the 70s a big board took weeks, even months, to design.

They routed on mylar, as paper isn't dimensionally stable when subjected to humidity variations, and the PCB was made via a photographic reduction of the drawing. My dad, a mechanical engineer from way back, tells me that in the 50s they had the same problem with paper when designing airplanes. Since this predated even mylar, they drew on starched linen in ink. Apparently the linen was stable, but it couldn't tolerate water. One drop of sweat dissolved the starch, ruining the drawing. And this was before Grumman, his employer, had air conditioning.

Me, I'm thankful for modern CAD. I'd never go back to those tedious days of yore.

Yet More on Tools

The tool discussion that started a couple of issues ago is still generating a lot of response. Thanks, all!

Several people suggested Ethereal. David Hinerman wrote: "On the topic of tools: I highly recommend Ethereal (http://www.ethereal.com) for Ethernet packet sniffing. We just got done adding an Ethernet interface to several of our products and this was indispensable. It's also free. It's available for Windows, Linux, and Unix."

Henk Dijkstra contributed: "My tip would be srecord http://srecord.sourceforge.net by Peter Miller, to manipulate target files for embedded systems.

"It has default support for numerous fileformats, and you can easily add new formats to it. By applying filters to the input files, you can also manipulate it if needed (like adding CRC/Checksum or crop/fill/exclude parts and much more).

"Another useful item for adding lightweight threads to an (existing) application is: Protothreads by Adam Dunkels: http://www.sics.se/~adam/pt/ . It consists of a few include files defining MACRO's that are used to implement threads as switch/case constructs. I have used it to easily add functionality in existing complex systems with high timing constraints."

Shaun Kaplan wrote: "In the last issue, under the tools discussion, Gennady Palitsky wrote in about ContentSaver. I too was looking for a similar tool and last week went flipping through the Firefox extensions and found one called Scrapbook (http://amb.vis.ne.jp/mozilla/scrapbook/). I've only had it a few days and I've never tried ContentSaver so I can't give a comparison but so far I'm very happy with it."

Jobs!

Let me know if you're hiring firmware or embedded designers. No recruiters please.

Nordic Semiconductor in Oslo has a few opportunities available. Seehttp://www.nordicsemi.no/index.cfm?obj=menu&act=displayMenu&men=11 .

Illinois-based C.E. Niehoff & Co has these openings:

Electronic Engineer - Industry leading and long-established manufacturer of automotive electrical equipment is looking for an Electronic engineer for their expanding Chicago North Shore operation. Successful candidate will be responsible for design and development of analog and digital power electronic circuits and controllers, particularly alternator voltage regulators. Candidates must have BS or MS degree in engineering. Min two years experience preferred including CAD and circuit design tools. Salary based on experience. Additional benefits include flexible environment, 401k, ESOP, health insurance and tuition reimbursement. Fax resume to (847)8647530.

Electrical Engineer: Also looking for an Electrical engineer for their expanding Chicago North Shore operation. Successful candidate will be responsible for design and development of automotive AC and DC generators, inverters and controllers. Candidates must have BS or MS degree in engineering. Min two years experience preferred including ProE or equivalent. Salary based on experience. Additional benefits include 401k, ESOP, health insurance and tuition reimbursement. Fax resume to (847)8647530.

Ember is looking for an Embedded SW Engineer and a Networking SW Developer to work in Boston. The Embedded SW Engineer will work on developing Ember's platform software. Candidates must have 5 years experience in embedded software development and a strong background in C and assembly language of at least one microcontroller such as the AVR, MSP430, 8051 or HC08.

The Networking Developer will work on our embedded wireless networking protocol stack. Candidates should have 5 years experience in software development with a strong background in C. You must be able to demonstrate the ability to implement simple solutions to complex distributed communications problems in highly resource-constrained environments.

Visit www.ember.com for detailed job descriptions. Send a resume to careers@ember.com

Joke for the Week

Stephan Beer sent in this one: "I'm really sorry, but I cannot create a new password. The stock is empty."

About The Embedded Muse

The Embedded Muse is an occasional newsletter sent via email by Jack Ganssle. Send complaints, comments, and contributions to him at jack@ganssle.com.

To subscribe, send a message to majordomo@ganssle.com, with the words “subscribe embedded *your-email-address”* in the body. To unsubscribe, change the message to “unsubscribe embedded *your-email-address*”.

The Embedded Muse is supported by The Ganssle Group, whose mission is to help embedded folks get better products to market faster. We offer seminars at your site offering hard-hitting ideas - and action - you can take now to ***improve firmware quality and decrease development time***. Contact us at info@ganssle.com for more information.