The Embedded Muse 129

Editor: Jack Ganssle (jack@ganssle.com) April 19th, 2006

EDITOR: Jack Ganssle, jack@ganssle.com

CONTENTS:

- Editor's Notes
- Failure Story
- Computer History and Tidbits
- Disasters
- Jobs!
- Joke for the Week
- About The Embedded Muse

Editor's Notes

One of my talks at this month's Embedded Systems Conference got quite a bit of press coverage. EE Times ran this quote: "The use of C is really criminal; C will compile a telephone directory, practically. I guess we use C because we think debugging is fun."

The quote was, of course, taken out of context. In the talk I went on to say that the tools we use to build tremendously complex applications are inherently fragile. C gives us fantastic capabilities, but by itself it offers little to protect us from errors. ********************i=3; is a perfectly legitimate statement... but no human knows what it means. Only good processes, smart and engaged people, and the use of other, complementary, tools will tame the hydra of software complexity.

Firmware content doubles every 10 months to two years; working *harder* is not a strategy to build these sorts of systems. (For more on the futility of working harder see "Quality is Personal" by Harry Roberts.) Instead, we need better approaches.

That's what my "Better Firmware Faster" seminar is all about. There are still a few seats in Denver (April 26) and Dallas (April 28). For more info see http://www.ganssle.com/classes.htm.

For the last half-dozen issues many readers have sent in comments about their favorite tools. Charley Moore kindly assembled this information and then we web-ized it. Find it all at: http://www.ganssle.com/tools.htm . Please continue to submit info about tools you love and hate; I'll have those comments added to the site.

Niels Malotaux has a number of interesting (and free) booklets on his site http://malotaux.nl/nrm/English/index.htm . He's an advocate for Evolutionary Delivery, a process invented by Tom Gilb, which is probably the granddaddy of all of the agile methods. The site also lists upcoming classes on the subject.

Failure Story

As always, last week's Embedded Systems Conference was a blast. One of my talks was about embedded disasters and the lessons we can draw from them. One attendee submitted his own:

In 1999 I was with a contract design and manufacturing company. We were contracted by a company to update an existing device which measures the opacity of smoke emanating from the smoke stack of a diesel truck. These devices are used at weigh stations around the country to enforce local pollution standards.

The device consists of a base (about the size of a toaster) and a stack sensor head. The sensor head is basically a big U with an IR LED on one side and a photodetector on the other side. Before the head is mounted on the smoke stack, the unit samples the level on the photodetector and makes that 0%. After the head is mounted, the smoke traveling between the LED and photodetector blocks a certain percentage of the IR light. That percentage is reported as the opacity.

One of the things that made this an interesting development was that the unit is used outside in various levels of direct sunlight. That's one reason we calibrate the unit by redefining the 0% point before every test. During development I used a flashlight to simulate the sun. I would aim it directly at the photodetector, go through the calibration, and then insert a neutral density filter and ensure I got the right reading.

The day before the customer was supposed to come to our offices to pick up the prototype, it occurred to us that we should take it outside and actually try to measure the smoke opacity of a small diesel truck we used to ferry stuff between buildings. The unit didn't work at all! After calibration but before we had placed the head on the smoke stack, we were getting readings all over the place. If we dangled the head from its cable and allowed it to rotate 360 degrees, the readings would vary from 0% up to 100%. The engineering manager asked two of us, the hardware designer and I, to do whatever it took to get the thing working before the customer arrived the next morning.

We spent hours going over the code and hardware design that night. Finally around 1 AM I pointed out that the existing product didn't have this problem. We started pouring over the schematics for the old unit. After 30 minutes or so the hardware engineer had an epiphany. He suddenly pointed at a strange looking circuit on the original schematic and said, "That's what that's for!"

Turns out that the original product had a sample and hold circuit that essentially removed any pre-existing light level just before the LED was turned on. This kept the amplifier from getting saturated even in direct sunlight. The hardware designer had studied the original schematic before starting his design, but he couldn't figure out what that circuit did and he didn't think it was necessary so he left it out.

The two lessons I learned from this experience were:

1) Don't wait until the end of the project to field test your device. Field testing should be part of your normal development cycle (as much as possible).

2) Don't assume that those who came before you were idiots. If you see something strange in the code or on the schematic, there's probably a good reason it was done that way.

Disasters

Having spent years studying embedded disasters, I must note that the anonymous contributor of the above story hit on one of the major causes of software failures: failure to test properly. Here are the top five proximate causes of dramatic failures in embedded applications:

1) Inadequate testing. We leave much of the testing till the tail end of development. Then the project runs late. Guess what gets cut? Even fairly disciplined teams make big testing mistakes, like not ensuring they actually execute all of the code. How can you be sure you executed every possible permutation of those 5-deep nested IFs?

Studies show that testing typically exercises only half the code.

2) Using testing instead of other methods, like code inspections and static analysis. Inspections, analysis (i.e., Lint and even more sophisticated automated checks) and testing all work synergistically together.

3) Lousy exception handlers. It's hard – really hard – to implement great exception and error handlers. They're even harder to test.

4) Tired people. 60, 70 and 80 hour weeks for months on end lead to one thing: bugs. When you consider that a single bit error out of millions can wreak utter havoc in a software system, it's becomes suddenly clear how much thought needs to go into each of those bits.

5) Don't cheat the tools. Can you believe that expensive missions have been launched… with the wrong version of the code? 40% of us use no version control at all so such errors aren't uncommon.

And do believe the tools, until they are proven wrong. The number of folks who get rid of compiler warning messages by filtering them all out is amazing. Yet a warning is nothing more than the compiler, which knows far more about the language than most of us, saying "hey, you're scaring me, man."

Computer History and Tidbits

If you're in Mountain View be sure to visit the Computer History Museum (http://www.computerhistory.org/). Their hours are limited so do check the site.

Don't bring your kids. It's not an interactive exhibit. This building houses serious artifacts that will appeal primarily to people who love computers. If you grew up with the microprocessor revolution you'll be interested in the collection of personal computers from the Altair 8800 on forward. Old-timers like me (50s and older) will be nostalgic over big iron that you used in days of yore. Then there's the stuff we only read about.

Though the exhibits are labeled don't wander around by yourself till you've been given a tour by one of the museum's docents.

One of those is Steve Leibson, well known writer, former EDN head honcho, and now technology evangelist for Tensilica. He gave me a fabulous two hour tour, explaining the fascinating and arcane history behind the artifacts.

The exhibit goes to the dawn of computing time, back to the sectors and abaci used by the ancients. Slide rules and ingenious mechanical contraptions that pre-dated Babbage are on display, as are the mechanical and electro-mechanical calculators engineers used as recently as the 60s.

There's the 10 MB disk drive… which is nearly three feet in diameter. Yet here on my desk is a one inch 6 GB drive Western Digital gave me. The 3 footer is encased in a thick machined aluminum housing which is nearly a work of art. The 6 GB almost looks like a trinket, yet stores 500 times as much data. A couple of dozen will fit in your pocket.

An original rack from the Eniac towers inches from visitors. It's just beautiful – clean, the tubes look new, it seems ready to run a program.

What was the first "personal" computer? The Altair? Maybe not. In the mid-50s the LGP-30 appeared, a machine that needed no A/C, used conventional power, and was cheap: about $40,000. Now that's 1950s dollars, which is a lot more today, but it made computing affordable for mid-sized companies.

I'd never heard of the thing till seeing the unit at this museum. Steve explained that it has only 113 tubes and 1350 diodes. It has neither memory nor even registers per se; a 4k rotating drum memory contained the registers, mass storage and program/data store.

Wow.

Fascinated, I later searched and found a little (in German) about one of its logic boards at http://computermuseum.informatik.uni-stuttgart.de/dev/lgp30/LGP-30_7.html .

Its successor was the LGP-21, which had only 460 transistors and 300 diodes! How did they build a computer with so few components? Well, the schematics are here: http://wps.com/projects/LGP-21/Documentation/LGP-21-Maintenance-Training-Manual/index.html .

Steve explained that Stan Frankel was the genius behind the LGP-30 and many other computer innovations. Stan who? His fame has been eclipsed by others and he's unfortunately a footnote to history. See Steve's write-up about this amazing person at http://www.hp9825.com/html/stan_frankel.html . Surf around the rest of the site for a tremendous amount of information about computer history.

Then there's the Museum's Cray Alley where you'll learn to pick out Seymour Cray's signature handiwork and will admire the sheer beauty of his creations.

Though the Museum isn't particularly large, it's packed with the artifacts described above and far more, many of which were considered obsolete junk when their usefulness expired.

Most of us trash old computers and technology when we move on to a newer, faster, more up-to-date model. I wonder if Eniac was itself rescued from some dumpster by a far-seeing conservator.

Jobs!

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

Crestron Electronics is actively seeking experienced and fresh-out embedded developers. We are located in New Jersey, close to Manhattan, between the Tappan Zee and George Washington bridges. We are a private company with over 700 employees and are growing 20-25% per year! There is a special section on our website – see http://www.crestron.com/features/careers/engineering/default.asp .

Joke for the Week

Late at Night Before Beta Test

Whose lines these are I think I know.
He's on another project though;
He should not mind my pausing here
to clarify the logic flow.

... though Management does NOT appear
to sanction major changes here.
"Fix it not 'fore it shall break"
and Beta Test is looming near.

Still, best case good ideas enmeshed
with side effects will fail the test
of year two thousand (coming soon).
Change just two lines, and then I'll rest.

Half vast possibilities run deep
but I have promises to keep
And lines to code before I sleep,
And lines to code before I sleep.

by Robot Frost (with apologies to Robert)

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.