The Embedded Muse 139

Editor: Jack Ganssle (jack@ganssle.com) January 9th, 2007

You may redistribute this newsletter for noncommercial purposes. For commercial use contact info@ganssle.com.

EDITOR: Jack Ganssle, jack@ganssle.com

CONTENTS:

Editor's Notes

I wish everyone a prosperous and peaceful 2007, and one that's free from schedule panics and overtime.

Did you know it IS possible to create accurate schedules? Or that most projects consume 50% of the development time in debug and test, and that it's not hard to slash that number drastically? Or that we know how to manage the quantitative relationship between complexity and bugs? Learn all this and much more at my Better Firmware Faster class, presented at your facility. See http://www.ganssle.com/brochure-onsite.pdf.

Thanks to everyone (491 of you) who filled out the salary survey last month. I've compiled the results, which are here: http://www.ganssle.com/salsurv2006.pdf . The results are interesting and sometimes surprising. Non-embedded developers do better than firmware people; in the USA the average age going up; and those who work a lot of overtime are generally happier than those who don't.

I've written recently about different approximations to speed floating point applications. Micro Digital sells a commercial library, which I recently looked over. The algorithms are apparently very different than most approaches and work in some sort of unpacked integer mode, while returning conventional IEEE 754 single or double-precision results. It's really fast. See http://www.smxrtos.com/ussw/gofast.htm .

Copyright 2003 by The Ganssle Group. All Rights Reserved. You may distribute this for non-commercial purposes. Contact us at info@ganssle.com for more information.

Engineering as a Process

Mike Cravens wrote an interesting and thoughtful email about a sort of meta-view of engineering:

"I believe that many people miss out on the significance of engineering as a process vs. a product, and this is most prevalent when the branch of engineering is software engineering.

"I spent my formative years following college working at the old TI Equipment Group, which does not exist any more. (Raytheon purchased the business).

"Early on, we were sent to a class given by a senior guy named Don on how to write an Engineering Change Notification.

"Once products were released, the process at TI EG said that we had to create an ECN any time anything was modified on the product or the process used to create and test it. The ECN had to be signed by all who used the product or component, which led green engineers along a path where they got to meet people that they would never otherwise talk to, including customer representatives.

"My group was writing embedded, real time software and designing hardware that tested the terrain following and targeting radar on a fighter plane which became the MRCA Tornado.

"My immediate boss was wise beyond his years, and when he created our drawing tree of parent and sibling documents for the fabrication and assembly of our special test sets, the top "patriarch" document was a process control specification.

"For software components, this included details such as exactly what version of what tools were needed to build a software component, as well as files such as link editor control files (today they would be Makefiles).

"All of the tools were archived, along with the actual operating software.

"More than a decade later, I was employed as a consultant in the telecom industry, and worked on an embedded design using the PIC micro controller as the on board maintenance processor for every card in a multi million dollar system.

"We had our issues with which versions of compilers and assemblers were needed to generate correct code. When I finished the development work, my boss asked me to go through the internal process needed to archive the results.

Copyright 2003 by The Ganssle Group. All Rights Reserved. You may distribute this for non-commercial purposes. Contact us at info@ganssle.com for more information.

"The compiler and debugger tools were many times larger and more complex than the code that I generated for the PIC. Without them, my code could never be duplicated or modified. They were documented and archived, along with my program.

"The first thing that I would do, if asked to modify such a system years later, would be to gather the original tools, take the original source, produce object, and do a binary difference with existing object. Only when the two matched would I move forward.

"You can carry this even further: You may need to archive the actual machine that the code was built on, and its operating system. Physically, or as a tested simulation.

"I was brought in as a consultant for one of the downstream users of an early video-on-demand companies, who supplied complete systems and programming to hotels and hospitals, even providing a broadband network infrastructure for free to sell their services.

"There was a need to add new educational programming services for a client market, or be displaced by a competitor.

"The company had not built their code from scratch in more than 10 years. In fact, they had decided to move to cross compilation rather than self hosting for a while, had bought a commuter and new tools, never tried the tools, and had subsequently sold the cross host machine for scrap.

"Our first task was to put together a development environment hosted on a "dead" OS, including compilers, linkers, and build control files, gather known source, and attempt to rebuild the shipping object from known source.

"This took several months, and was a real adventure. A year and a half down the road, job complete, we sold them a working development environment for what the hardware and software components cost us as surplus items. Even the disk drives were a special 256 byte per sector version which are no longer manufactured.

"We were able to re-host development of the application code we created, by building a simulated environment on a PC using Borland Pascal tools and an IDE which is itself now obsolete.

"I have no idea if they have kept that machine running, or in mothballs, but re-constructing it cost them a lot of money, and held the key to them being able to re-create and evolve their old product.

"We turned the whole thing over to a roadie-like guy in a moving truck which smelled

Copyright 2003 by The Ganssle Group. All Rights Reserved. You may distribute this for non-commercial purposes. Contact us at info@ganssle.com for more information.

like outdoor concerts. I hope he found his way back with the gear in one piece.

“These are unrelated projects, but the common theme is: engineering creates a process, which is used to create products.

“You can throw away a product and keep the process and recover. The reverse is not true.

“As "the Don" told us "You're getting paid to produce the documentation. The product is just the test case".”

Tool Guardians

Bill Gatliff, the very well-known Linux guru and trainer (see billgatliff.com), responded to my discussion about Tool Guardians:

“The solution you're looking for in your "Tool Guardians" editorial can be found in Free and Open Source Software. No, I'm not kidding! Here's my experience.

“During the development phase for a project I try to track the latest GCC release. When a new compiler version is announced, I run my code through it and see what breaks. There are always a few issues, but most of the time they're due to sloppy code--- stuff that's unclear to the compiler as to what the developer was asking for. Fast-and-loose casts, data representation assumptions, etc. are all invitations for trouble and there's no better way to find them other than through code reviews and tweaking the settings/versions of your compiler from time to time. If your code is ambiguous, it will behave differently a.k.a. "break" in different contexts because it's genuinely broken. It really needs to be fixed.

“(A common theme in Dan Saks' ESC/ESP articles over the years is that you have to carefully follow the syntax rules of your language if you want your compiler to help you out. Sage advice).

“There are the occasional compiler bugs, sure. They get reported and fixed in due time, just like with any other software product. And if the problem is so bad that I can't use that compiler version, I just roll back to the previous version until things get sorted out. I've had to do that on only one occasion in almost a decade of embedded development with GCC.

“When a project ships, I save the source code for the compiler as part of the project source code. Then at some point in the future, I have the option of either re-installing that

Copyright 2003 by The Ganssle Group. All Rights Reserved. You may distribute this for non-commercial purposes. Contact us at info@ganssle.com for more information.

compiler, or bumping up to the currently-released version when I'm doing follow-up maintenance on the project's application. To date, it hasn't seemed to matter if I'm running an ancient version of GCC on a platform that didn't exist at the time the product was originally shipped: I've tested it several times, and the new application image will compare byte-for-byte with the original version.

"Of course, there's nothing preventing you from continuing to track the latest compiler version even after a product ships. The code tweaks that you have to make as a result--- if any--- don't have to go anywhere until you do another software release of the product. Where I have the budget, I'll do that as an investment towards maintenance down the road. With all the automation opportunities that command-line tools like gcc and gdb offer, the burden to do a lightweight quality-assurance test from time to time is pretty minimal.

"I follow this approach for all my tools, not just GCC.

"There are lots of good compilers and other development tools out there. But the ones you can get the source code for are the ones that really let you be your own Tool Guardian. Those are the ones I prefer every time."

Jakob Engblom also responded:
> Take for instance a PCB layout program, a staple of every
> engineering department. I was bitten by an upgrade to
> one of these products, and discovered to my horror that
> all of the libraries were now obsolete.
> What a joy it is to maintain both old and new versions of the
> software.

"One way to solve this that I saw a friend of mine use is to use VmWare or similar products (Xen; Virtual PC; Parallels) to have multiple OS installs, each with its own tools version installed.

"This was in the context of (embedded development, believe it or not) needing to have some three or four different versions of Visual C++ in use at the same time for different clients. Basically, each client was using a particular version, and since different version's project files are not exactly compatible, they needed to use the right VC++ versioin for each project... And since VC++ does not like being installed together with another version of VC++, virtual machines was the only way to go.

"With a virtual machine, you can also freeze that old OS version needed."

Copyright 2003 by The Ganssle Group. All Rights Reserved. You may distribute this for non-commercial purposes. Contact us at info@ganssle.com for more information.

Paul Tiplady commented:
“Tool guardians are a good idea. At my last company, where we were building engine and chassis control systems, we kept the executables for all the tools in the version control system, and built everything from a home-grown 'recipe' file. It specified the filename and revision of everything needed to build the application, including the tools. The only two tools not fully protected in this way were the version control system and the tool that read the recipe file to get all the other tools out of version control.

“I guess this gets way more complicated with Windows and shared DLLs and the like, but we were running under DOS, which made life much simpler in many ways (or maybe I'm just old-fashioned).

“Furthermore, every time we did a build, the build tool put the recipe file away in the version control system too. This meant we could guarantee to rebuild anything we'd ever built, whether it worked or not, and be sure to end up with the same object code every time. And it didn't take up that much disk space, because the version control tool stored incremental changes, and we didn't change much from one build to the next.

“The switch from DOS to Windows was slow, because we had to re-run a whole series of representative builds under windows before we were happy that our friends at Micro$oft hadn't pulled the rug from under our feet, but that's a different story...”

Bob Paddock’s thoughts were short and too the point:
"Software Upgrades Are Tantamount to Protection Rackets, Says Cutter Consortium" – Full article here: http://software.tekrati.com/research/News.asp?id=8212

Steve deRosier wrote:
“Your recent comments on the needed for "Tool Guardians" struck a couple of nerves with me. I maintain a number of Linux applications for our "embedded Linux" product (Opus7: http://www.pianodisc.com/products/details.aspx?id=39795) that are written in C++. We use the gcc that is standard with our Linux workstations to compile this. During an OS upgrade, the gcc version changed...to one that implemented the new proper C++ standard, breaking a large amount of syntax. Easy enough to fix, but a large consumption of both coding and testing time.

“The second story is actually much grimmer. We recently lost 2/3 of our facility in a major fire. This included the entire engineering department. We have a product that requires an old MS DOS machine to create update disks (the low-level HW code won't work in modern Windows). This machine was destroyed in the fire. We do have the executable, but no machine to run it on. Even worse, we can't recompile and change the code to work on Linux, because the source for the program was sitting on old floppies

Copyright 2003 by The Ganssle Group. All Rights Reserved. You may distribute this for non-commercial purposes. Contact us at info@ganssle.com for more information.

somewhere in the department and never made it into our much newer CVS system. Much of our legacy code was destroyed as no one ever had time to copy it from the old floppies and place it in our CVS system.

"For those people who must support old products and have developed a proper software tools environment more recently: take the time to get the old code into your repository. Especially the source of those little "unimportant" tools and utilities that your department depends upon but takes for granted since they aren't shipping products."

This is very wise advice. Just today I heard from another developer whose office burned a month ago… and they still have not been able to get back into the building. Can you imagine the effect on a business from this?

Jobs!

Let me know if you're hiring firmware or embedded designers. No recruiters please, and I reserve the right to edit ads to fit the format and intents of this newsletter.

Gigle Semiconductor is a new fabless semiconductor company developing system-on-chip ICs for the inhome multimedia market. We are looking for bright, enthusiastic and dedicated candidates, with an ability to learn fast and to work as part of a team in the area of Embedded Firmware Design. Your location will be Barcelona, Spain, and your role will be to cover the specification and development phases of real time communication protocols using C and assembly for our system on chip development.

Experience in the following areas will be an asset:
- Real Time Embedded operating systems
- Networking protocols
- MAC and QoS protocols

Send your application to hr@gigle.biz

Schweitzer Engineering Laboratories (SEL) seeks a professional, innovative and meticulous individual for our Lead Software Engineer position. If you are looking for an opportunity to design new products using the latest development tools working on a dynamic team in a results oriented environment, then this may be the position for you! SEL has a reputation for quality, reliability, integrity, and service. SEL is located in Eastern Washington where you'll enjoy an unmatched quality of life. Please submit your resume through our website at www.selinc.com/careers.

Copyright 2003 by The Ganssle Group. All Rights Reserved. You may distribute this for non-commercial purposes. Contact us at info@ganssle.com for more information.

Agere Systems' Munich Mobility Office has several embedded SW development positions open, and we are willing to relocate the right candidates. Given all of the layoffs in Germany by biggies like Siemens over the last few years, it's surprisingly hard to get German nationals interested in (as my German office-mate puts it) "an American company that you've never heard of". Agere does have a very good reputation in the mobile space - we are Samsung's number one supplier of 2.5G chipsets. Here are some positions:

Senior (7+ years exp.) Software Engineer, System Framework - MOB000000F6 - http://tinyurl.com/y8uw94

Software Engineer (3+ years exp.), System Framework - MOB000000F3 - http://tinyurl.com/yxn8f5

Joke for the Week

.
Find a lot of funny and profound quotes about computers here:
http://www.eskimo.com/~hottub/software/programming_quotes.html

Two examples:
There are only two industries that refer to their customers as "users".
- Edward Tufte

I have always wished that my computer would be as easy to use as my telephone. My wish has come true. I no longer know how to use my telephone.
- Bjarne Stroustrup

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*". ". BUT - please use YOUR email address in place of "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

Copyright 2003 by The Ganssle Group. All Rights Reserved. You may distribute this for non-commercial purposes. Contact us at info@ganssle.com for more information.

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.

Copyright 2003 by The Ganssle Group. All Rights Reserved. You may distribute this for non-commercial purposes. Contact us at info@ganssle.com for more information.