The Embedded Muse 160

Editor: Jack Ganssle (jack@ganssle.com) May 9, 2008

EDITOR: Jack Ganssle, jack@ganssle.com

CONTENTS:

Editor's Notes

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 this and far more at my Better Firmware Faster class, presented at YOUR facility. See http://www.ganssle.com/classes.htm .

Vinzent Hoefler sent along this interesting link about Ada:
http://www.gcn.com/print/27_8/46116-1.html .

Clyde Shappee was working on floating point algorithms and noted how the Patriot Missile failed in the first Gulf war. Some of us still shudder when thinking about this software problem; others haven't heard of it. A quick summary is here:
http://www.ima.umn.edu/~arnold/disasters/patriot.html .

The April issue of IEEE Computer looks at those North American colleges that award doctorial degrees. Only half as many people enrolled in computer science majors in 2007

compared to 2000. Total enrollments slipped 18% in the single year between 2005/6 and 2006/7. That sounds pretty alarming, so I hunted around for more data and came across this NSF site: http://www.nsf.gov/statistics/nsf07307/pdf/nsf07307.pdf . NSF has looked at degrees awarded by all colleges, not just PhD-granting institutions. The data runs from 1966 through 2004, and science & engineering bachelor degrees have represented between 31 and 35% of all degrees awarded in the US. The number was 32% in 2004, which is pretty typical over that nearly 40 year period. Engineering accounted for 5-8% of all bachelors (5% in 2004), while math and computer science varied from 2-6%, hovering around 5% in the 2000s.

The two data sets are by no means matched; not only do they differ in terms of PhD granting institutions, Computer's data is for all of North America and the NSF's is just for the US. And NSF lumped math and computer science together.

Which leads to a question I'm often asked: which universities excel in teaching embedded systems development? I'll publish responses so we can all get some perspective.

****

****

Firmware – Best Practices

In sync with this college theme, Richard Wall recently contacted me about a class he's developing. His email follows, and I'd like to hear ideas from you folks, and of course will pass them along in the next Muse.

As I come to the end of another semester of teaching microcontrollers, I am faced again with the challenge of attempting to teach students to write quality (in the sense of Robert Pirsig) embedded software. I am looking for the few and the good "best practices" that can result in functional, reliable, and maintainable software based systems. My search has yielded two such lists: "10 Tips Make Embedded-System Code Easy to Maintain" by Timothy Stapko, Digi International, Ecnmag.com - January 01, 2008 and "The Power of

10: Rules for Developing Safety- Critical Code"
(http://www.spinroot.com/gerard/pdf/Power_of_Ten.pdf) From my own experience and search on the subject, I have comprised my own list that I can start junior level electrical and computer engineers down the right path. Some of these are borrowed for other lists.

1. Quality Code: How to get there
- Recognize that writing quality software is an acquired disciplined art.
- Writing quality code requires practice, diligence and assessment (outside review).
- Make writing quality code part of your personal process.
- Teach how to write quality code. (I ponder this ling and hard - any ideas?)
- Value quality code. (corporate and individual)

2. Impediments to quality code
- Time pressure
- Lack of training / experience
- Motivation - lack of professionalism or just plain lazy

3. Quality code the implementation
- Well structured
  a) Consistency + Neatness
  b) Use accepted (mandated) practices
    Simple structure
      1. Simple instructions
      2. One instruction per line
      3. Minimal use of macros (I have a horror story about multi-instruction macros)
      4. Effective modularization: Limit LOC per function to one screen of code. Use task portioning and functional encapsulation (I have seen some embedded articles that discourage too many functions because they require both time and memory.)
    Clean interfaces
      1. Pass minimal data (reduce memory requirements and increase speed)
      2. Expose only variables that are necessary: minimal use of global variables (IO port and configuration registers as well as variable use to communicate with ISRs are notable exceptions.)
      3. Minimize dependencies confine processor dependent code to specific functions
    Functionality
      1. Minimize amount of code: less LOC is usually better
      2. Test frequently completely, and thoroughly
      3. Start simple - add complexity in layers
      4. Zero tolerance for errors (no “know bugs” list)

I need a list that is easily implementable, easily enforced, and easily learned. Do you have any suggestions on things to include in my list or sources of such lists to leverage

from?

VDC Survey Results

To all the Embedded Muse readers who participated in VDCs 2008 Embedded Systems Engineering Survey, thank you! Although the survey is still open at this point, we wanted to share some of the preliminary results around salary and employment data with you.

You can access this information at:
www.vdc-corp.com/ganssle/salary_data.pdf

The survey will remain open over the rest of this month. We look forward to analyzing the remainder of the data and e-mailing a much more substantial summary of the results of the survey to all of those who participated shortly. Thanks again for your contributions.

If you havent participated in the survey yet, and would like to, it is still available at:
www.vdc-corp.com/08esdt?RID=G

Tools and Tips

Robert Bristow-Johnson sent: “Because division is so expensive in a DSP or a RISC (usually it costs one instruction cycle for each *bit* in your quotient), I have always tried to make polynomial approximations without the need to divide by another expression (another polynomial). I came up with some pretty good results using a MATLAB program I wrote that is an implementation of the Remes Exchange Algorithms for finding coefficients that minimize the maximum weighted error.

“If you want and have time, please review some of the code, it's short and reasonably well commented, I think. in the file __functions.c is the code that has the coefficients for 2^x, log2(x), sqrt(x), sin(pi*x), and so on. It was written for a job I did where they had a floating-point processor, but no standard math lib for C (so they could add, subtract, multiply and divide, but little else). It *is* the case that for arctan(x), I *did* have to divide (the optimized polynomial goes into the denominator), but the rest of the functions needed no divide.

“I have the same functions implemented in DSP code for fixed point, but that code does not belong to me (but to a previous client), so I can't send it to you. But the coefficients are the same, one has to deal with the arithmetic shifting and ranging differently, but once the x value is between 0 and 1 (or between 1 and 2 for log2(x)), it's all the same. I am

also including the MATLAB code I use to determine the optimized coefficients. I just hope that exposure to someone else's tweaking might be both interesting and useful to you."

The code is available here: http://www.ganssle.com/misc/approx.zip

Greg Williams wrote: "Actually, it's been a while since I've used Eclipse for C development. It looks like the current version of Eclipse finally supports C and C++ text formatting and does it pretty well, with help of the CDT (C Development Tools). Therefore, I think that SimpleIndent has worn out it's usefulness. The built-in formatter for C/C++ in Eclipse is pretty full featured and has nice configuration via the Eclipse Preferences
GUI.

"Although, the AStyle formatter may still be good to mention as another option. It's freely available via SourceForge at: http://astyle.sourceforge.net/

"The page does a pretty good overview of it's features. With its command-line interface, it is really easy to patch in to any editor."

Isn't it amazing what is on-line? Anto Jurkovic found this: "There is very useful master thesis from Dominic Rath with the title: Embedded Systems Trace Solutions. It talks about debugging techniques, and debugging implementations. The link is: http://www.hs-augsburg.de/~hhoegl/da/ma-3/master_thesis.pdf ."

More Response to Great Engineers

Stephen Pelc wrote:

> John Taylor had some good thoughts in response to this from Steve
> Litt in the last Muse:

> I contend that the guru is using a troubleshooting
> process. Sure, it looks to *us* like he sniffs, licks his
> finger and touches a node, and immediately discovers the
> problem, but what he really did was use a process to
> troubleshoot, including sophisticated use of divide and
> conquer. Either that or he's seen the symptom before and
> remembers the corresponding fix.

“The response is good, but ... the topic of debugging really, really pushes a button in me! IMHO, debugging requires two primary comprehensions:
1) Don't let them in
2) When they are there, fixing them is just formal scientific method:
  a) Observation
  b) Hypothesis
  c) Experiment with a yes/no answer
  d) Repeat until experiment succeeds
  e) Fix
and go round for the next bug. Debugging is a process.

“In this view, debugging consists of two nested loops. How fast you can debug is dependent of how fast you go round the inner loop a to d above.

“I've run classes on this topic for years and years, but it's truly amazing how few people get it. If you cannot observe accurately (John Taylor's point), you can't debug efficiently.

“There are corollaries that we've known for decades, but still don't apply, in particular:
1) Documented code is much easier to maintain. And if write this stuff as part of the implementation, your initial bug level goes way down. We're big fans of literate programmming.
2) Because bugs interact, it is much cheaper to fix the bugs as soon as they are seen than to leave them to the next "ample spare time" moment.

“But then, we're a Forth shop, and the main reason for using an interactive language on the target is for observation. And because our VFX compilers are optimising compilers using the same techniques as most C compilers, we don't have a performance penalty for the interactivity. And we use literate programming.”

Free Stuff

Want to learn about DSP programming? Microchip has generously donated an MPLAB starter kit for the DsPIC digital signal controller. Send us a prediction about what embedded development will be like in 10 years, and one of those entries will be selected at random to get the DsPIC kit.

Contest rules: one winner only, who will be selected at random on or about May 16 from all of the entries. Send your entry to marybeth@ganssle.com . Sorry, we can’t acknowledge entries. I’ll post the winner’s name (but no other contact info) in the next Muse and any funny or interesting commentary.

Newnes is offering a 20% discount till May 31 to Muse readers who want a copy of my new book. See http://www.ganssle.com/misc/adesch1.pdf for the discount code and the book's first chapter.

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.

Atomic Object of Grand Rapids, Michigan is looking for both experienced embedded developers and solid generalist developers. We're an Agile methods shop and are very serious about Test-Driven Development.

Atomic Object people are more than technical wizards; they're also proficient in writing, managing changing requirements, working directly with clients, and satisfying customers. They care about good code, create tools to automate tasks, are skilled at problem solving, think testing is essential and not optional, are teachable, and know how to learn the things they don't know. If you are interested in applying Test-Driven Development to firmware development, and aren't afraid of development work on both ends of the wire (i.e. embedded *and* high-level software), then you might just be the type of embedded developer we'd like to meet.

See our site (http://www.atomicobject.com/pages/Working+at+Atomic+Object) for more information.

Mindtribe has a bunch of open positions at a product development company (with a great team of engineers and work environment in general!). Were here in downtown Palo Alto, CA. See http://www.mindtribe.com/jobs.html

Nordic Semiconductor has openings for Embedded Software Developers and Senior Software Test Engineer in Trondheim, Norway.

The Senior Software Test Engineer will take a leading role in the testing of ULP Bluetooth protocols and other software products.

As Embedded Software Developers we are looking for both newly qualified and experienced software developers. You have experience with embedded microcontroller

programming in C and an interest in radio protocols and wireless applications. Tasks include development of protocol stacks for ULP Bluetooth and proprietary protocol, as well as development of support tools.

See http://www.nordicsemi.com/files/Embedded_SW_dev_job_ad.pdf
and http://www.nordicsemi.com/files/Senior_SW_test_eng_job_ad.pdf
for the full texts.

Based in Minneapolis, Transition Networks distributes hardware-based connectivity solutions exclusively through a network of resellers in 50 countries. We're looking for a Sr. Software Engineer /Embedded Linux.

As a senior software developer, you will be responsible for planning and developing new products in an environment focused on the requirements, analysis, design, coding, debugging of embedded software.

Specific Skills/Abilities
- Demonstrated expertise in embedded systems development.
- Familiarity with C and structured programming techniques.
- Experience in writing applications for Linux.
- Experience in writing embedded software for SNMP network management.
- Experience in taking products from concept to market.
- Good communication skills, both oral and written.

Education/Experience
- BS in Computer Science or Technical Field is required
- 6+ years experience in embedded systems firmware development

Contact Joet@transition.com

Netrino provides engineering services related to the design and development of embedded systems. Netrino provides expert assistance in product development and in depth training on a variety of embedded programming topics. Netrino has a number of openings:

We are looking for a Software Engineer to develop high quality embedded applications for our clients. The right person for this job will be very comfortable designing and coding a variety of GUIs and application-level programs for embedded systems. Communication and problem solving skills are essential for success in this position. This position will be located in Columbia, MD.

Responsibilities:
-Collaborate with clients on specifications and functional requirements
-Design, develop, and document high quality embedded applications
-System architecture and coding
-Code verification and validation
-Assess and improve application performance
-Analyze legacy applications

Requirements:
-BSCS, or BSCE
-At least 5 years experience
-Experience with Windows CE or Windows Mobile
-Experience with Microsoft .Net/C# programming
-Experience designing graphical user interfaces
-Strong experience with object-oriented programming
-Strong problem solving ability
-Strong communication skills

Netrino is also looking for a distinguished Electrical Engineer to provide direction, guidance, and instruction to clients and other engineers within the company. This person needs to be not only experienced, but readily recognized as an expert in electrical, electronic, and embedded hardware applications. Communication skills are essential for this position. The right engineer for this job will be able to readily explain complex issues to customers and prospective clients and will enjoy teaching. This position will be located in Columbia, Maryland.

Responsibilities:
-Evaluate client RFPs and write proposals
-Provide expert advice and explanation to Netrino's clients and prospects
-Provide direction, advice, and training to Netrino engineers
-Oversee technical quality in Netrino's hardware designs
-Client consulting
-Hardware architecture and schematic design

Requirements:
-BSEE
-Not less than 15 years experience
-Demonstrated mastery with microprocessor board design
-Extensive experience with board design tools
-Experience with programming tools and environments
-Experience with varied processors, languages, and operating systems
-Extensive digital and analog circuit design experience

Netrino is also looking for Firmware Engineers to write high quality code for our clients. The right people for this job will be very comfortable designing and coding a variety of embedded drivers and applications. Communication and problem solving skills are essential for success in this position. For the right person, relocation may not be necessary.

Responsibilities:
-Design, develop, and document high quality embedded code
-Collaborate with clients on functional and performance requirements
-Code verification and validation
-Assess and improve code performance
-Analyze code from other engineers

Requirements:
-BSEE, BSCS, or BSCE
-At least 5 years experience
-Experience with a range of processor families
-RTOS experience
-Experience with different communications protocols
-Experience with varied programming environments
-Strong C and some assembly coding expertise
-Strong problem solving ability
-Strong communication skills

If you have what any of these jobs require, we want you! Please respond with a resume and cover letter explaining why you should be one of Netrino's Software Engineers. Reply to careers@netrino.com .

Joke for the Week

Steve Bresson has some hints about recognizing engineers.

You might be an engineer if...

Q: What is the definition of a engineer?
A: Someone who solves a problem you didn't know you had in a way you don't understand

Q: When does a person decide to become a engineer?
A: When he realizes he doesn't have the charisma to

be an undertaker.

Q: What do engineers use for birth control?
A: Their personalities

Q: How can you tell an extroverted engineer?
A: When he talks to you, he looks at your shoes instead of his own

Q: Why did the engineer cross the road?
A: Because they looked in the file and that's what they did last year

Q: How do you drive a engineer completely insane?
A: Tie him to a chair, stand in front of him, and fold up a road map the wrong way

You might be an engineer if:
Choosing to buy flowers for your girlfriend or upgrading your RAM is a moral dilemma.

You take a cruise so you can go on a personal tour of the engine room.

In college you thought Spring Break was metal fatigue failure.

The sales people at the local computer store can't answer any of your questions.

You can quote scenes from any Monty Python movie.

You can type 70 words per minute but can't read your own handwriting.

You look forward to Christmas so you can put the kids' toys together.

You see a good design and still have to change it.

You spent more on your calculator than you did on your wedding ring.

You still own a slide rule and know how to use it.

You think that people yawning around you are sleep deprived.

Your wife hasn't the foggiest idea of what you do at work.

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.

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.