The Embedded Muse 135

Editor: Jack Ganssle (jack@ganssle.com) October 24, 2006

EDITOR: Jack Ganssle, jack@ganssle.com

CONTENTS:

Editor’s Notes

The place: Santa Clara, California. The date: December 1. The event: I’ll present my “Better Firmware Faster” seminar. Check out http://www.ganssle.com/classes.htm for details. The site is just a few miles from San Jose airport, which is inexpensively served by Southwest Airlines and others. Come for the seminar on Friday and explore San Francisco over the weekend! Register by November 1 and save.

Do you have a dozen or more developers who could benefit from this firmware training? I can present my Better Firmware Faster class at your facility. See http://www.ganssle.com/classes.htm.

For European readers, I’ll be giving a couple of talks at Oredev in Malmo, Sweden in November. This new conference looks interesting, with some intriguing speakers like Betrand Meyer and Martin Fowler. More info here: http://www.oredev.org/ .

Paul Swingle wrote about another source of free books. I haven’t tried this yet but it sounds interesting: Your citation of a source of free online books reminded me of a tremendous site: www.NetLibrary.com. They have <over 30,000> books online for free. Amazingly you can search the whole library by the content of the books. For example you could look for all books containing "embedded and design and firmware" and it would respond with the 83 books that contain all three of those words. The books contain <everything> in the physical books including the cover, table of contents, figures, photographs, and index.

The catch is that you must sign up initially in a library that offers the service. In my experience, most public libraries and almost all university libraries offer it (although they often don't know they do). The bigger libraries seem to have bigger collections (I presume that the more they pay, the more titles they can offer). You do not have to belong to the library; you can just walk in and sign on using one of their computers. After you have signed up, you can use the service anytime, anywhere.

In response to my links to some approximations, Bernd Toerner sent this: You have mentioned approximations routines for trigonometry functions. Perhaps it is interesting for the readers that there is an other way to calculate the functions: CORDIC. Basically it is a way to do this job in hardware with FPGA, but on uC it is possible too (need no variables of float or double type, only integers; the calculation is done with simple addition and shift operations)

You can find information at http://www.dspguru.com/info/faqs/cordic.htm and http://www.andraka.com/cordic.htm

New Kinds of Debugging Tools

Bugs and debugging have always fascinated me. Long ago, working on Nova minicomputers my colleagues and I debugged programs using crude tools and a panel full of switches, feeling like the pilot of a spaceship as we furiously toggled contacts. Later I started an in-circuit emulator company, where we designed and sold tools used to debug hardware and software in embedded systems.

We know a lot about getting rid of bugs. One approach is to use tools (like Ada or SPARK) that keep us from making mistakes in the first place. Another is to employ inspections to remove bugs before we start debugging. Test finds those that slip through.

Then there’s static analysis, which finds defects before code inspection. Lint, for instance, has been around for ages and has become widely adopted. But in recent years new classes of static analyzers, those that find runtime errors without executing the code, have been clamoring for attention. Companies like Polyspace, Klocwork, Coverity and others offer these sorts of compelling products.

I find this sort of technology fascinating. Pour your source code through a tool which identifies the bugs? Cool! You'd think the companies would be beating customers away. Price would be immaterial.

But do the tools actually work?

If you're using, or have used, these advanced static analyzers, drop me an email. Were you happy with the tools? Were they worth the price? Why or why not? What sorts of errors were trapped… and which slipped through?

I'll share the results in a future Muse.

More on Reading Code

Doug Abbott responded to my comments about reading code: Thank you, Jack, for forcefully articulating a problem that has bugged me for a long time. I too am a great admirer of Jean LaBrosse's MicroC/OS and have used it for several years in teaching real-time programming concepts.

For a number of years now, I've been involved in the Linux and Open Source movements primarily teaching Linux to embedded developers.
But, frankly, I'm absolutely appalled at the quality of Open Source source code. Sadly, the vast majority of it is write-only. Someone obviously wrote it, but no one else can read it.

I have seen many open source code files where the ^only^ comment is the GPL boilerplate at the top. And it's not like the code itself offers any useful hints. All too many programmers seem to have an aversion to vowels in symbol names.

For a long time, I have advocated the view that programming really has very little, if anything, to do with computers. The computer doesn't understand C, it doesn't even understand assembly language.
Heck, even the 1's and 0's are abstractions.

When you write a program, what you're really doing is communicating to another human what it is you want the computer to do. Programming is about communication, just like any other form of writing. So yes, computer science students should be exposed to good writing. But since there's so precious little of it in computer science, maybe they just need some basic creative writing classes.

Several readers wondered if there was a good place to find truly beautiful C++ source to read. I'm stumped. Any ideas?

Start With the Manual

Xtian Xultz wrote from Brazil:
I worked sometimes ago for a firm developing a car alarm system. I was developing the first alarm for the firm. It was a really simple circuit (and cheap) with a tiny microcontroller. The designer who contracted me said something like "I wish for a product that does this set of features." I agreed, and we signed a contract.

I started the development, and when I finished the first prototype, I installed it in the car, so he could test it. He tested the product for two days, and then came with new features to put in the firmware, always asking "is it possible to implement such a function", and I said "yes, of course" and that answer always made him happy.

Then, I put the new firmware in his prototype. He tested for a day or two, and then came back with new ideas. I told him some "yes, it's possible" and I wrote new functions for the product. The third time he asked for new functions, I asked him if the product would not become too complicated for the user, but he said no. Well, my job was to write the firmware, so I did it. He asked me some more functions, and I wrote all of them. When the product was quite stable, he started the first production.

Out of curiosity, I picked up the user's manual he wrote, and took a look. He wrote about only 1/3 of features of the product. I asked him why he did so, and the answer was funny: because it would be too complicated for the user. Well, what happened was not so funny: some users called me with questions about problems with the product, that was in fact features, but the manual didn't say about that. And as a result, a lot of functions became ignored in the manual, and consequently by the user, functions that I had to write, debug, document, etc.

The next product he asked me to make, I said to him to imagine all the product functions, and to first of all write the users manual. To develop that product was a heaven: when I made a product that did all of the manual functions, the product was finished. Today, I think is a really good approach for any product development.

Tools

Dave Kellogg wrote: This tool of an entirely different sort, but at least equally as useful as editors and GUIs: www.Manager-Tools.com offers a weekly podcast comprised of practical, hands-on, effective how-to advice covering a wide range of topics. Most of us embedded software folks are not managers. Even so, these topics still are very often exactly on target. We each need to take an active part in developing both our careers and soft people skills, and these podcasts certainly help. Past casts have covered: Resume writing, managing your boss, effective meetings, brain storming, building a network, how to resign, plus about 60 others. The audio form makes effective use my commuting time. Highly recommended.

Roland Bennett likes PSPad: I really appreciate the feedback from users regarding their favorite editors. Having used Visual SlickEdit for many years, I was looking for a free editor when I left the company where I had used it.

I used Notepad++ for a long time, and appreciated the fact that I could add the keywords for VxWorks to the code completion function by simply copy and pasting the text from the VxWorks PDF manuals into a text file!

However recently I discovered an even better editor: PSPad. http://www.pspad.com/en/ ,which has all the options I wish Notepad++ had! Difference tool, hex editor, file explorer, etc.

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.

KVH Industries, Inc. http://www.kvh.com/main/index.asp, has an immediate need for a Embedded Software Engineer in its Middletown, RI office.

KVH is an international leader in developing and manufacturing innovative, mobile, high-bandwidth satellite communications systems, tactical navigation products, and fiber optic products.

The position researches, designs, and develops systems-level software, in conjunction with hardware product development, and provides systems support. To learn more about the position please refer to the job posting on the KVH Career website http://www.kvh.com/CareerOpps/

Joke for the Week

.

Though this submission from Dan McCarty isn’t a joke, it does make you think:

Ever wonder what all those kilobytes, megabytes and gigabytes do? Here's something you might find interesting...

1 bit: a 0 or a 1
8 bits: a byte (can store 256 different values)
16 bits: the instruction size of a Super Nintendo
32 bits: the instruction size of the Pentium chip you're using right now

Now here's where it REALLY takes off...

10**2 bits: a hectobit (100 bits)
- For example, the key size for some encryption algorithms these days

10**3 bits: a kilobit (1,000 bits)
- For example, the size of the magnetic stripe on your credit card
- 1/75 of a second of one channel of audio

10**4 bits: (10,000 bits)
- one page of typewritten text

10**5 bits: the size of a short-to-medium newspaper article

10**6 bits: a megabit
- a one-page black and white fax
- one frame of a DVD (about 1/30 of a second)

10**7 bits:
- the size of a screenfull of data on a medium-resolution monitor
- a single 3.5" floppy disk (remember those?)

10**8 bits:
- amount of data in a large foldout map
- amount of data in a 5-minute, CD-quality recording
- amount of data your eyes can digest in a single glance

10**9 bits: a gigabit
- the capacity of a regular audio CD

1o**10 bits: the capacity of a single-sided DVD

10**11 bits: the capacity of a Blu-ray (high-def) DVD

10**12 bits: a terabit
- A medium sized hard drive these days
- the amount of space needed to store the largest number of digits of pi ever recorded (1.24 trillion)

10**13 bits: possibly the capacity of your functional memory

10**14 bits: the size of the Library of Congress in digital form

10**15 bits: a petabit
- The size of the Internet Archive (an organization that tries to maintain an archive of the Internet)

10**16 bits: the amount of hard drive space that Google uses

10**17 bits: the size of the memory of the robot, Data, in Star Trek: 800,000,000,000,000,000 bits

10**18 bits: an exabit
- The amount of all printed material in the world

Putting it all in perspective...

10**22 bits: the amount of data stored in 1 gram of your DNA.

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 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.