The Embedded Muse 166

Editor: Jack Ganssle (jack@ganssle.com) October 6, 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 .

```
switch(city){
```

```
case San Jose:
case Phoenix:
case Austin:

Are you in the San Jose, Phoenix or Austin areas? I'll present a public
version of the Better Firmware Faster class in San Jose on December 8,
Phoenix on December 10 and Austin on December 12. Registration and
other info here: http://www.ganssle.com/classes.htm . You'll earn 0.7
Continuing Education Units, learn a lot, and have more than a little
fun. Sign up before November 8 and receive a $50.00 discount.
}
```

Emmett Kyle sent a link to this page: http://klabs.org/history/build_agc/ . If you've ever wanted to build your own Block I Apollo Guidance Computer, you've GOT to check this out!

Comm Monitors

Dale Botkin contributed yet another approach to monitoring comm links: "There's a much simpler and far less expensive solution. I have yet to find a PC that actually requires EIA-232 voltage levels to operate; in fact, if you review the data sheets for line receivers back as far as the venerable 1489, you will find that none of the ones in general use require a negative voltage. For reasonable cable lengths and environments, a simple inverter will do the job quite nicely.

http://www.botkin.org/dale/rs232_interface.htm

This is a circuit I've used in production for several years now with no issues talking to a wide range of PCs, laptops and other serial devices."

Book Review

This week I had the pleasure of reading John Davies' new book "MSP430 Microcontroller Basics." It's a 668 page course in embedded systems, using the MSP430 as an example platform. It's also a great reference to the MSP430, whether you're a novice or a gray-beard who needs to know more about that part.

The MSP430 from TI is a 16 bit RISC-like microcontroller that sports extremely low-power modes and a nice mix of on-board memory and peripherals. TI offers some cool low-cost development platforms. Once a stealth part, it has garnered great market share.

Unlike some books that jump all around, or others that dive right into deep technical discussions, this book starts with a high level view of the MSP430 itself. From there it moves into an introduction to embedded systems, one that's aimed at college CS or EE majors just getting exposed to the whacky world of embedded. That does not mean it's a textbook; far from it as the material is very accessible and the writing lively and engaging. Just the sort of thing I wish textbooks were, and I could see this being used in a hands-on college course.

An intro to embedded? Yawn, right? No, stay with me.

John shows aspiring developers how to write a simple C program to turn an LED on, first in C and then in assembly. These are complete tiny apps that will compile/assemble. He describes the intent of each line, and couples that description to the hardware design. He then reframes the assembly into relocatable code and shows what that is all about.

An intro to embedded C and assembly? Double yawn. Nope - hang in there.

Using the same careful explanations he grows the discussion to use buttons, timers instead of delay loops, and much more. The lessons are complete, detailed, and clear.

Then the good stuff starts for working MSP430 engineers. He describes in clear detail with code and hardware designs the use of the MSP430 and its peripherals. Need to run the SPI in a particular mode? The code is there. Want to get interrupts up? Just copy the listings (there's no CD but he does provide a web link to the code).

He keeps drilling deeper, giving more complex examples that are all directly useable by any MSP430 developer. Oscilloscope traces, where applicable, ground the discussion into the physical effects we engineers need.

The MSP430 does have analog I/O, and John doesn't neglect that. He talks about aliasing, signal/noise ratio issues, and handling the A/Ds. Yep – the math is there, too, not complicated at all, but essential to working with analog. A CS person or novice EE would learn a lot about electronic circuits from this material.

Unusually, this book is both a great introduction to embedded systems for the novice, as well as an incredibly-useful reference for experienced developers. Highly recommended. The folks at Newnes Press have offered Muse readers a 20% discount good till the end of the year – go to www.newnespress.com and use discount code 93682.

Coders Vs Programmers

I get ticked off whenever a manager uses the word "coder."

Unless, of course, he or she is referring to a peon who does nothing but crank code.

Historically "coder" means someone who translates a very detailed design to C, C++ or whatever language is used. In the old IBM "chief programmer" days coders were an important part of the development team, but were at the bottom of the pyramid. It was assumed these folks were trainees or had little expertise, so they slaved at more or less rote translation activities. Coders were analogous to draftsmen, those skilled technicians who render engineering concepts into the important but dull drawings needed to actually build something.

Let's face it: it's not hard to write code. Even teenaged hackers manage to build simple systems. UML advocates often demonstrate that the machine itself can write the code from a sufficiently-detailed design.

The challenge we engineers face is designing a solution to a particular problem. I believe the real fun is in inventing the solutions, not in the dreary coding process. For me the most exciting and creative part of a project is doing the rough system design, figuring out what should be done in hardware and what in the firmware, and then blocking out the overall approach.

But the only complete design spec tends to be the code. There are a hundred little decisions we make as we type in C statements, from oddball exception handling to what happens if some really weird combination of inputs happens. What happens if the user presses all three buttons, belches and kicks the machine? In my experience virtually no design document captures every essence of what the program will really do.

In the embedded world there are few true coders. We're highly trained engineers who code and design together, or who create the structure and then implement it in firmware. Perhaps we'd be more efficient if we did separate designers and coders, but this seems unlikely to happen. It's certainly valuable to keep ones hands in the dirty business of actually making the system and getting it to work. The crucible of making things work always teaches us a lot about the limits of what we can actually build.

The "coders" word is usually used by managers who view software as a necessary evil, rather than a critical part of the value delivered to the customer. That attitude dooms projects and frustrates the staff.

Desktop developers embrace "programmer" as a job description. Perhaps that adequately captures the limited range of skills needed to crank out a Windows app. Me, I prefer "engineer", a moniker that better describes the breadth of experience needed for embedded systems, where hardware and software seamlessly blend together to yield a product.

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.

Netrino has multiple open positions for embedded software developers and FPGA and board-level electronics designers, in Maryland and California. Join the Embedded Systems Experts(tm) in a fun work environment that offers challenge and variety. Learn more at http://www.netrino.com/Embedded-Systems/Embedded-Software-Jobs

St. Jude Medical: Atrial Fibrillation Division Epicor Product. We're looking for a Lead Software Engineer to develop embedded firmware for new and existing commercial medical devices. We offer a small, flexible engineering environment with an exciting opportunity to architect and design new code from the ground up on a next generation platform. The ideal candidate would have expertise in developing embedded C/C++ code (requirements development, design, coding, hardware integration and test) in a regulated environment.

Desired Skills/Experience:
* B.S. degree or equivalent experience
* 5+ years work related experience
* Experience with embedded C/C++ software development
* Real Time Embedded operating systems concepts and low level programming
* Experience developing in regulated environments (medical, aerospace, etc.)
* Knowledge of embedded digital hardware architectures

Please contact tciciarelli@sjm.com.

Terumo Cardiovascular Systems is hiring a Senior Embedded Hardware Engineer. This position is responsible for the development of a blood parameter monitoring system used in cardiovascular surgery. This is a lead designer position for a custom analog/digital design on a team of experienced engineers from multiple disciplines included Mechanical, Software and Systems Engineering. It typically requires a four-year degree, PhD strongly preferred (not required), coupled with at least 8 years experience in an engineering environment developing, designing and specifying products. Terumo Cardiovascular Systems is a medical device manufacturer specializing in the design, production, marketing and service of life-sustaining cardiovascular equipment and disposables. www.terumo-cvs.com

Joke for the Week

Brian Souter sent this:
I don't know if you follow the scientific journals and news or not. But lately there has been talk about how they are generating light pulses whose length is measured in 'attoseconds'.

So how long is an attosecond? It is 1/1000 of a femtosecond. How long is a femtosecond? A femtosecond is one millionth of one billionth of a second. Or in scientific notation, a femtosecond is 1*10^-15 (1E-15), and thus, an attosecond is 1*10^-18 (1E-18). Or for short (pardon the pun), an attosecond is an insignificant amount of time.

Does this mean when your boss gives you an attoboy, he is giving you an insignificant amount of praise?

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.