The Embedded Muse 86

Editor: Jack Ganssle (jack@ganssle.com) October 20, 2003

EDITOR: Jack Ganssle, jack@ganssle.com

CONTENTS:

Editor's Notes

I'll present my Better Firmware Faster seminar December 5 in San Jose, the last public class this year. See http://www.ganssle.com/classes.htm for more details. This is the only non-vendor class around that shows practical, hard-hitting ways to get your products out much faster with fewer bugs. The crummy economy is putting pressure on all developers– come and learn the tricks you need to be more efficient.

There's also cheap fly-in options listed there for folks coming from out-of-town.

I often do this seminar on-site, for companies with a dozen or more embedded folks who'd like to learn more efficient ways to build firmware. See http://www.ganssle.com/onsite.htm.

Rob Wehrli has a free port of uClinux (Linux for Microcontrollers) available for download. This port currently runs a 2.4.20 uClinux kernel version on the Hitachi H8S-2674R "EDOSK" board. The board is available for around $200 from Avnet, Reptron and Nu Horizons, but he has several to give away to anyone interested in actively developing GPL code for it. See http://www.azpower.com/H8-uClinux for more details.

Perri Matthews had an interesting twist on my watchdog comments. He wrote: "About 15 years ago or so, when I was a fairly young software engineer, I worked on an 8086-based embedded system where the electrical designer thought it would be a good idea to tie the watchdog strobe line to I/O-space reads & writes. He figured that since we would be doing inputs and outputs to I/O-mapped devices fairly often, this would keep the

watchdog happy and he could save an extra wire. At the time I didn't think much of it, until I had to figure out why the watchdog wasn't helping us at all when the program would crash. The reason was that we were doing I/O accesses in our interrupt service routines! So even though the foreground code was hosed, the interrupts were still working and keeping the watchdog happy.

"Now I always use a more elaborate two-level scheme for servicing a WDT. Since I don't like to sprinkle watchdog accesses all over my code, and I don't like having to make sure that I count cycles in my code so that I don't inadvertently add too many lines between WDT services, I have an interrupt service routine that services the WDT at a high enough rate, but only if a flag has been set by the foreground code. If the flag is not set, the ISR will continue to service the WDT for only about 10 seconds (or whatever time the designer chooses), at which time it will quit servicing the WDT and allow it to reset the CPU. The ISR clears the flag whenever it detects the flag is set. So all I need to do is set the flag periodically in my foreground code, usually at only one place in the main executive loop, or in other long loops such as polling for serial input, etc. Then if the foreground code gets stupid, it will stop setting the flag, and the background will eventually give up and let the WDT do its thing to bring the system back on track."

Taming C

In my most recent Embedded Pulse article (http://embedded.com/showArticle.jhtml?articleID=15306089) I bashed the C language. As I write this the replies – both positive and negative – are still flooding in.

Don't get me wrong – I think C is a fantastic language, one that can lead to wonderful products. You can't beat it for bit twiddling and getting close to the hardware.

But C does little to force us to build good code; it's very easily abused.

I read a lot of code – a LOT of code – and the sad fact is that little of what goes into modern products is well-written. You can almost see the heroics that went into beating the code into submission.

Use all of C's intricacies with abandon and your firmware is going to be a mess. As professionals we must tame it, using discipline, wisdom, the experience of others, and the appropriate tools.

Develop code without Lint, for instance, and you're flirting with suicide. Lint is a syntax checker on steroids. Though most of its warnings are nitpicks, it's truly astonishing how many real flaws Lint will uncover.

Write firmware that generates no (or few) Lint warnings. That means changing the way we produce code, using explicit casts, more parenthesis than is rigidly needed, and the like. But if a sophisticated syntax-checker is confused, your code is probably suspect.

There are plenty of Lints around. My two favorites are splint (http://www.splint.org/), which is free, and PC-Lint (http://www.gimpel.com/), which is not.

For those new to the idea, Ian Darwin wrote "the" book on the subject (Checking C Programs with Lint, http://www.oreilly.com/catalog/lint/). But most of us will do fine using just Lint's user manual.

Isn't it ironic that this book was written by Mr. Darwin – I'd argue that not using Lint is a sure way to find your career become extinct!

Also consider using tools to check the quality and complexity of your code. Oakwood Computing's (http://www.oakcomp.co.uk/) Safer C Toolset analyzes source files to look for code that's likely to be wrong.

Programming Research's (http://www.programmingresearch.com) QA-C tool applies various metrics to measure a C file's complexity; if the code is too complicated it's rarely reliable.

Both companies also produce tools that check C against the MISRA (http://www.misra.org.uk/) standards, a set of guidelines that promotes better C code. A very intriguing alternative way to check source against this standard comes from The Mathworks. This paper (http://www.mathworks.com/company/digest/july03/checking_code.shtml) shows how to configure MatLab as a MISRA compliance-checker.

(I reviewed MISRA in Muse 52 - http://www.ganssle.com/tem/tem52.pdf).

Also see http://www.plethora.net/~seebs/c/10com.html for the 10 Commandments for C Programmers.

Jobs

Dynamic Telecommunications in Germantown, MD is hiring a number of people, including:
- Quality/Manufacturing Engineer
- Senior Software Design Engineer
- Electrical Engineer
- DSP Engineer

- Program Manager
- Systems Engineer (Technical Product Manager)

Information on all openings can be found at: http://www.dynatele.com/careers.htm.

Let me know if you're hiring firmware or embedded designers. I'll continue to run ads for embedded developers as long as the job situation stays in the dumper.

Joke for the Week

Colin Walls wrote:

An optimist thinks the glass is half full.
A pessimist thinks a glass is half empty.
The software engineer thinks the glass is twice as big as it needs to be. [Or, rather, the buffer overflow protection is rather over done].

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.