The Embedded Muse 171

Copyright 2008 TGG December 15, 2008

EDITOR: Jack Ganssle (jack@ganssle.com)

CONTENTS:

****
This issue of The Embedded Muse is sponsored by Netrino.

You know it is possible to write reliable and maintainable embedded software. So why are there so many bugs in your company's firmware?

It turns out there a few dozen coding best practices that dramatically reduce firmware bug counts, in programs with or without an RTOS. Improve the bug reducing skills of your team at the hands-on Embedded Software Boot Camp. The next public session runs January 26-30, 2009. Full details are online at http://www.netrino.com/Embedded-Systems/Training-Courses/Boot-Camp-Muse

****

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 .

Spectrum has a fascinating article about the birth of the fourth circuit element, the memristor: http://www.spectrum.ieee.org/dec08/7024

Quotes and Thoughts

I collect quotes, tidbits that illustrate some point, and interesting short ideas, especially if there's some real or metaphorical connection to the computer business. Starting with this issue I'll share them in the Muse to perhaps make you think, wince or grin:

We know about as much about software quality problems as they knew about the Black Plague in the 1600s. We've seen the victims' agonies and helped burn the corpses. We don't know what causes it; we don't really know if there is only one disease. We just suffer - and keep pouring our sewage into our water supply.
- Tom Van Vleck

Criminal Coding

There was a time when a "database" was a drawer of index cards. Word processing meant creating text on a typewriter, using carbon paper instead of a "print 3 copies" command. The only intelligence in a car resided in the brain of the driver. Factory controllers used banks of relays singing their clicking songs.

Mainframe computers were generally inaccessible to ordinary people. Most folks experienced computing disasters only in the form of an insane credit card bill or the IRS' repeated demands for immediate payment of a zero dollar tax due.

Today cars have computerized car brakes sans backup hydraulic system. All modern aircraft can and do fly themselves and many land without human intervention. A 100,000 ton tanker heavily laden with a potential environmental catastrophe relies entirely on autopilot, GPS and radar for navigation and collision avoidance. Factories producing the most noxious and toxic of chemicals would grind to a standstill – or perhaps fail spectacularly – without an array of microprocessors that sequence every activity.

The firmware component of these systems has all grown spectacularly since the advent of the microprocessor, from programs consisting of but a few thousand lines of assembly to today's millions of lines of C or C++. We've learned many things over that time about building better code; one is that "perfect" software is a practical impossibility. The best code ever written is probably that in the Space Shuttle, but even at a cost of $1000 per line defects still appear, though at the amazingly low rate of about 1 bug per 400,000 lines.

All large systems have lurking hidden problems. Most of us work hard to ensure we've addressed real safety issues. None of us knows how to prove we've built something that is correct and that will never fail.

I was struck by a letter in the RISKS digest (http://catless.ncl.ac.uk/Risks/21.84.html#subj10.1) about the most common security problem found in Unix and Windows systems. Henry Baker wrote suggesting that programmers producing software that does not check for buffer overflows are criminally negligent, and should perhaps be liable for resulting damages. He makes the very interesting point that it's truly trivial to check incoming data streams for length, that we've known to do this for a generation or more, and yet an endless succession of bug reports out of Redmond and CERT testify to programmers totally neglecting this obvious problem.

One or two buffer overflow problems in an application that causes crashes or security vulnerabilities are just bugs, easily correctable, and perhaps do not reflect on the developers' abilities or ethics. Repeated buffer overflow bugs, though, are a different animal indeed. That providers of PC programs do not take such action suggests they either have no clue what they're doing, or simply don't care. They're negligent. If such a simple and easily-avoidable bug causes a corporation to lose data, does the customer have a case against the developers?

Extending this line of reasoning a bit, the software community has learned that a disciplined development process yields better code: fewer bugs. Yet very few of us employ a rigid process. For instance, we know from countless studies that keeping functions to a single page reduces defects. How many of us enforce a short-functions requirement? Code testing is a notoriously ineffective way to uncover bugs. How many of us couple effective tests with code coverage checks or inspections?

If bugs result from undisciplined engineering, particularly when better approaches are known, is this malpractice?

The lead paint on toys debacle this year reinforced our notion that a EULA is no defense against defective products. Ship something dangerous and expect to be sued. Ship something defective, and lemon laws and class actions await.

But in software we routinely have bug lists. Software is the only industry left on the planet that gets away with selling known defective products. I wonder how long that will last?

Dot Com Redux?

Economists predict a gloomy 2009. But considering the great job they did predicting the financial crisis it's hard to give them any credibility. So last issue I asked for peoples' feelings on the current economic woes in terms of their jobs and/or the embedded industry in general for the next year. Most respondents wanted to remain anonymous so I've summarized the results:

10% - Business as usual
4% - Things will get better soon
31% - Somewhat pessimistic
52% - Really pessimistic
3% - The end of the world as we know it

Here are a few thoughts readers wanted to share. Paul Wright wrote: Our company is growing and expanding - possibly because our market in medical alarms is something that is needed regardless of how much money is about - last thing to cancel and some government funding in various countries. We have taken on 2 new R+D Engineers and more production workers. Business as usual.

I would hazard a guess that a lot of companies would be wise to continue R+D so when the next boom comes they are ready with up-to-date products for consumers.

(Comment from Jack: A number of readers repeated the thought in the last paragraph. Certainly we learned from the dot-com bust that dumping engineers is a short-term solution with unfortunate long-term results. But tight credit markets could make it simply impossible to meet payrolls without some very painful decisions.)

Anonymous wrote: Layoff meeting last night. No idea who or how many will be let go but engineering "will not be unaffected."

Another reader's thoughts mirrored many others: Sales are good but the company has become very cost conscious.

More on Reuse

Roland Bennett had a few comments on reuse: At my previous company, we had always done our device drivers for VxWorks. The need arose to port the driver to Windows, and for that I designed an OS wrapper layer (HAL) if you will. Basically I searched for all the

VxWorks specific functions in the existing code base, and ended up with only 20 or so VxWorks specific API calls. I then searched for similar functions in the Windows API and came up with a design or wrapper API that did a good job with abstracting the specific OS API call. It wasn't always a one-to-one mapping, it most cases a single API call in one OS would require multiple API calls in another.

Later on the need arose to port the driver to Linux as well, and with this OS wrapper layer in place, it meant that only the function implementations had to be rewritten. It took less than a week!

Functions like sleep are the easiest to code. You define a new API e.g. osSleep that takes as input milliseconds. In the implementation you implement your conversion to microseconds or whatever and call the native sleep method.

Your driver code will have your OS Wrapper header include files, and the only difference between ports would be which library you add to the project. The *.lib (Windows) or *.a (Linux) one.

With not a lot of effort it was fairly easy to abstract away the differences between the OSes. The abstraction included ISR, memory mapping API and threads. If on a certain platform virtual memory wasn't supported; the function simply returns the same pointer, without doing any MMU lookups.

Datatypes are easy to ensure consistency, by using the sizeof operation one can create types like osINT32 to ensure a 4 byte integer. A simple header file with conditional defines can ensure the correct int size for all platforms.

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. Please keep it to 100 words.

Horizon Hobby, Inc. is seeking a Senior Engineer - Electrical for work in the Proprietary Product Development Department at our world headquarters in Champaign, Illinois. Full details are available online at http://www.horizonhobby.com/Horizon/Careers.aspx This is the ideal job for an RC hobbyist with experience in development of electronics used in the consumer electronics/hobby arenas.

Despite the downturn, embedded systems consultancy Netrino (http://www.netrino.com) is still growing fast. The company currently has open positions for embedded software developers as well as digital and analog electronics designers in many parts of the U.S. The company's mix of development, consulting, and training work will entertain your brain and makes for a collegial learning environment. If you may be interested, please send your current resume to careers@netrino.com.

Joke for the Week

Appropriate and timely – from Paul Bennett:

New Discovery - The Heaviest Element Known to Science

Lawrence Livermore Laboratories has discovered the heaviest element yet known to science.

The new element, Governmentium (Gv), has one neutron, 25 assistant neutrons, 88 deputy neutrons, and 198 assistant deputy neutrons, giving it an atomic mass of 312.

These 312 particles are held together by forces called morons, which are surrounded by vast quantities of lepton-like particles called peons.

Since Governmentium has no electrons, it is inert; however, it can be detected, because it impedes every reaction with which it comes into contact. A tiny amount of Governmentium can cause a reaction that would normally take less than a second, to take from 4 days to 4 years to complete.

Governmentium has a normal half-life of 2- 6 years. It does not decay, but instead undergoes a reorganization in which a portion of the assistant neutrons and deputy neutrons exchange places. In fact, Governmentium's mass will actually increase over time, since each reorganization will cause more morons to become neutrons, forming isodopes.

This characteristic of morons promotion leads some scientists to believe that Governmentium is formed whenever morons reach a critical concentration. This hypothetical quantity is referred to as critical morass.

When catalysed with money, Governmentium becomes Administratium, an element that radiates just as much energy as Governmentium since it has half as many peons but twice as many morons

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.