
Embedded Muse 178 Copyright 2009 TGG April 19, 2009

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 .

Rich Testardi sent this cool introduction to building embedded systems: http://www.cpustick.com/index.htm .

Quotes and Thoughts

Geoff Patch sent in this gem: The trouble with the world is that the stupid are cocksure and the intelligent are full of doubt. - Bertrand Russell.

Tools

John Johnson contributed this: "From time to time I like to prototype code in an environment that does not have much baggage like Windows IO. LccWin32 is a DOS C/C++ compiler that works on WinXP is available from a number of sources.

"The following text appears on the http://www.cs.virginia.edu/~lcc-win32/ website.

"License: This software is not freeware, it is copyrighted by Jacob Navia. It's free for non-commercial use, if you use it professionally you have to have to buy a license.

“There are others similar compilers, e.g. Watcom, Borland, etc. that have been made available by the owners. LccWin32 does a good job for my purposes.

“The C compiler in Cygwin does a good job as well but you have work in a Cygwin window outside of the DOS/Windows environment making printing difficult. And Cygwin come with its own baggage, i.e. it is a full Unix implementation.”

Computer Science Education

Paul Carpenter commented on Pete Klammer’s statement: I am more concerned about "Software Engineering" than Computer Science education, because it is the engineering phase. Paul wrote: "I don't advocate prohibition of programming by unlicensed developers. But I do want something like truth in labeling: a manufacturer or public agency should be able to tell (and consumers should be able to find out) whether or not its software or firmware was developed by, or under the responsible supervision of, a licensed Software Professional Engineer, and with that information we can shop and negotiate and pay accordingly. Perhaps some critical (life-critical? Mission-critical? Safety-critical) products may even require such development, but I'm not going that far myself: just give me the choice at least, and I for one will pay a little more for the airplane, car, X-ray machine, etc., that came that way.

“Yes and no, unfortunately bureaucrats get involved, so they will stipulate Comp Sci qualifications, and to make their life easier to get this agreed to by bodies like lawyers, insurance companies (even your professional indemnity), it would become mandatory.

“The body that licenses would ensure it becomes mandatory to justify their position and get more revenue! This has happened around the world in various spheres.

“To the extent that in the UK many electronic or electrical engineers in the UK are not allowed in a lot of places to wire up the mains plug or even work on fixed wiring without expensive certificates and specialised insurance, annually renewed!”

David Fern wrote: “I would like to comment on Peter Klammer contribution to Computer Science Education. I agree with its position about the fact that Software Engineering is more important than Computer Science for practical purposes, but in the case of critical systems, what must be certified is the product, not necessarily the engineers that implemented it. Of course, I agree that a product developed following certain quality standards and by qualified engineers can be better. The point I want to make is that the product must be tested and certified to be ready for use on a critical system by an independent third party, even if it has been developed by the most capable engineers.”

Marty Hauff had this thought: :In response to Pete Klammer's ideas relating to Computer Science Education I have this to say:

“While I applaud any endeavour to improve software quality, I don't think it is relevant whether the software was designed by certified 'Engineers' or not. I don't believe you can assume anything about the quality of a product purely from the qualifications of the designers. It is for this reason that we have specialized standards for specific industries such as military, aerospace, and automotive. So while it might be equally qualified engineers working across the different industries, they are each producing different products that must conform to different requirements. Surely the ultimate test is whether the product conforms to the pertinent standard rather than the qualifications of the people who designed it.”

Responses to Naming Conventions

This topic is an important one to a lot of people. Rich Ries wrote: “ I would forgo any type of spell checking in an editor* -- after all, I spell well -- for something like Word's Comments, with a "Hide Comments" feature. There are times when I want to stick ideas into the code, without adding to the code's commentary. (I guess it would usually be stuff like, "What WAS he thinking of??")

“*Try putting some code into Word and spell-check it!

“As to the despised CamelCase, the ancient software we use translates ALL names to upper case for the link map. So we get such beauts as:

ADDBITTOMSG
CHKCRCEEPROM
CHKOFFNORMAL
DOQWIKARM
MDMGETRANNUM
MDMWAITONECPTASK
“... and many more.

“Then there's the lovely:
DSCNTL
DSPCNT
DSPCT
“... all used in the same file. There is a program, Liberty BASIC, with an editor that will flag similar variable names, so it would catch the above three, as well as case errors: DSPCNT and DspCnt. So that type of error-catching is possible.

“Hey, while I'm on a roll (buttered, of course!), I seem to remember that the F83 editor had a means of showing comments in a separate, aligned window. That should not be too hard to do, I would think. This way, if you want to concentrate on the code, or on the comments, you could expand the appropriate window, then restore it when you're done.

“On Long Variable Names: There has to be a "comfort level" that does not violate the "comprehensibility level." Typing "lfc_Modem_Do_A_Quick_Callback" repeatedly loses its appeal rapidly. I saw ASM code once that had function names that went across half the 17" printout width. The problem was that the first 50 or so characters were all the same. UGH! Thankfully, I did not have to work on that project!”

Steve Melnikoff wrote: On the use of underscores versus CamelCase: our convention is for variables with file scope, and non-static functions, to be named with the module name and identifier name in CamelCase, joined by an underscore, e.g. Uart_Rx(), Log_au8Buffer[], etc.

“Neither method of breaking up words on its own would suffice here, hence we use both.

“For macros, just the module name is in capitals, e.g. UART_u8MaxNumOfBytes, LOG_Join().

“I agree that acronyms present a problem for CamelCase, so we go for what is hopefully the lesser of two evils, and treat them as words, e.g. Uart, Spi, I2c.

“I've been prefixing variables with their type for a while, so it was interesting to read this article by Joel Spolsky, which suggests that what may be the norm in embedded software may not be what Simonyi had in mind: http://www.joelonsoftware.com/articles/Wrong.html

“Incidentally, I recommend the "Joel on Software" blog. Though he is more concerned with PC and Web software than embedded, a lot of what he writes is common to both worlds.

“Re MISRA types: This is purely subjective, but I've never liked "int8_t" etc as type names. "int" harks of the C types we're trying to avoid, and I feel the names are a bit...bulky, given how often they would be used.

“Instead, I prefer u8, u16, u32, s8, s16 and s32. They're short, they get away from C's type names - and they're identical to the prefixes we use in variable names.

“Stack Overflow (http://stackoverflow.com/) is a programming Q&A website, where the questions and answers are ranked by the users. Anyone can ask or answer a question, but the ability to vote, comment on other people's answers, and edit content are only bestowed on users who have gained sufficient reputation by having their questions and answers voted on by others. In doing so, it aims to keep the quality of the content high.

“It was created by Jeff Atwood (http://www.codinghorror.com/ - another interesting blog) and the aforementioned Joel Spolsky, and so does lean towards PC and web software. However, there are a small number of embedded systems-related questions on there, as well more general, but relevant, material on C, business, good software practices, and the like.

“It would be nice to increase its embedded-related content, so I'd definitely recommend this site to your readers.

“(For what it's worth, my current reputation is 196; the most anyone has is 47,100; so I have a way to go yet! My profile is here: http://stackoverflow.com/users/45552/).”

Dave Hansen submitted this: Naming Conventions are kind of a hot button with me. I agree with most of what you said. Some caveats and additional comments:

“Spelling: I recently looked at some code that consistently misspelled a particular variable name by dropping one of the vowels in the middle. Something like right_anlog_difference. Yet a corresponding variable name (something like left_analog_difference) and several other references to the same word (e.g., process_analog_value) were spelled correctly. How could this be? The previous engineer had used one of those very "helpful" IDE's that automatically inserted variable and field names for him by presenting him with a list of choices as he typed. Who needs spellcheck? Sheesh. I'm allergic to IDEs anyway, but that's a topic for another day...

“Case: I like to use case to denote a symbol's place in a name hierarchy. Obviously, ALL_CAPS is used for #define and enum constants, but there are other useful rules:

“Type names and file-scope variables get title caps. In addition, type names have a _T suffix. E.g. Vehicle_Direction_T, Current_Vehicle_Direction. Global-scope variables don't exist.

“Function-like macros similarly get title caps, but with a leading lower-case "m". Since macros perform simple text substitution, they can be dangereous and should be treated with more care, and the "m" is just a gentle reminder. It also dissuades overeager maintainers from "optimizing" out the "function call" by inserting code in-line...

“File-scope (static) functions, function-scope variables, and struct or union fields are all lower-case. Global-scope functions get an all-caps prefix indicating the subsystem (and often the source file) they are part of, e.g., end_calibration and UI_display_calibration_screen. Local automatic variables get names as short as I can clearly make them (psrc for "pointer to source data", idx for "loop index"), while local statics are more descriptive (e.g, previous_state). No formal standard there, just what feels right.

“And function names should always have exactly one verb.

“I generally mangle acronyms to fit the symbol type, but that rule is not so hard and fast, and may vary depending on my employer. E.g., PWM_FREQUENCY, Pwm_Duty_Cycle, PWM_set_duty_cycle, init_pwm_hardware.

“I have in the past used struct tags and typedef names that differ only in case, and I would argue that's still valid, e.g., typedef struct pwm_data Pwm_Data. Since these symbols live in different namespaces, the C standard would allow them to be identical. However, I've since started using something like typedef struct pwm_data Pwm_Data_T;

“Hungarian Notation: As used by many, it's an abomination and should be outlawed. I have seen it cause more bugs than it has prevented (e.g., where the type of the variable was misdeclared, but the name reflected the intent, resulting in an invalid usage of the sizeof operator to test the limits of an array). As originally envisioned by Simonyi,

they're not a completely bad idea, but with less general applicability than is envisioned by most practitioners. So I use prefixes in a similar way, though they tend to be slightly longer: pdest for "pointer to destination", is_calibration_active for a function returning or a variable containing a boolean value. Semantic information rather than type. A symbol should describe the variable or function rather than its type.

“If a value can have different units, those go in a suffix: pressure_adc for the raw ADC reading, pressure_kPa or pressure_psi for kPa or PSI, valve_current_mA for millamps. With suffixes, you have to be careful not to run into the name length limit though. I never seem to have too much trouble with that, however...”

Tom Harris wrote: “In further response to Naming Conventions, I have found that coding standards (whether for style or for usage) have to be short to be usable. Developers have to be able to understand, memorize, and internalize them. MISRA is good but too long to deploy unmodified. Here was my alternative entry:

“Very Short Coding Standard (VSCS)

“VSCS Rule #1: Use Meaningful Entity Names
“VSCS Rule #2: Maintain Zero Compiler Warnings

“Read more at http://talkaboutquality.wordpress.com/2006/07/09/very-short-coding-standard/. Or, to keep the VSCS short, don't read more!”

Paul Carpenter sent this: “Long names are a great way to convey meaning, but C99 requires that only the first 31 and 63 identifiers to be significant for external and internal names, respectively. Restrict all names to 31 characters or less. (And define them as "static" if their scope is local.)

“If the name is too long and contains non-obvious capitalisation, it becomes a pain using it (see a lot of Windows API programming), you end up making more typos when writing code. let alone the continual what was that variable/function called EXACTLY.

“If you have long names, make them VERY different, to avoid calling the wrong function or using the wrong variable

```
BathroomSinkTapHot0
BathroomSinkTapHot1
```

“Instead of

```
BathroomSinkTapHotClose
BathroomSinkTapHotOpen
```

“Avoid names that can be spelt in different dialects/languages differently for the same thing. Examples being US and non-US English for

```
color          colour
```

capitalize capitalise
authorize authorise

“I was just doing some updates to some 10 year old code where I have variables like

data_path
temp_path
data_root
server_root

“My main bug bear with types is signedness and lack of using it correctly. For instance:

“1/ Standard libraries and language usage have signed when they should have unsigned:

File Handles
__LINE__ macro
Array indecii (the actual memory index for associative arrays)
Database record numbers
Social Security numbers
Badge numbers
ID numbers

“Have you ever seen a negative one?

“I would love to see the negative line number for source code! What is a negative file handle?

“2/ Variables relating to real world physics that are impossible when something is Absolute it is UNSIGNED, and Relative it is SIGNED, e.g. Light is absolute, Reflection of light is relative to source and reflective, surface, taken from ABSOLUTE measurements/values.

“Measurements like Voltage and Current, these are ALWAYS relative as the two wire method means we have an implied referenece. Some can be implied relative (water flow compared to no flow), but depending on situation may flow either way (tides).

“See http://www.opengroup.org/onlinepubs/009695399/basedefs/stdint.h.html for some interesting extensions to these typedefs for use where performance issues mean we want the compiler to make the smartest decisions possible.

“Unfortunately the one thing I don't like about that page is that integers are talked about generically "intN" but pointers are ALWAYS 16 bit! We moved on from there DECADES ago (68000/8086) and i regularly work on embedded micro controllers with 24 or 32 bit pointers as NORMAL.”

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.

Rain Bird Corporation Golf Division in Tucson, AZ is looking for an exceptional Senior Product Engineer to define and create software-based irrigation control systems. Responsibilities include project planning and high-level software application design. Lead engineering effort to accomplish new product designs. This person must be a strong project leader and is expected to provide technical leadership. He or she may also be required to lead the development effort of more than one significant project.

Please visit http://jobs-rainbird.icims.com/jobs/1879/job for more information.

We have several positions open with relo available to Melbourne, FL for highly skilled EEs and firmware programmers. BIOS programmers to the top, please!

See:
http://www.recruitingsite.com/csbsites/drs/JobDescription.asp?SiteID=10371&JobNumber=594627

...send resumes to rwehrli@drs-ts.com for immediate review.

Joke for the Week

Tatu kemppainen sent in this classic:

A Tourist walked into a pet shop and was looking at the animals on display. While he was there, another customer walked in and said to the shopkeeper, "I'll have an AutoCAD monkey please."

The shopkeeper nodded, went over to a cage at the side of the shop and took out a monkey. He fitted a collar and leash, handed it to the customer, saying, "That'll be $5000." The customer paid and walked out with his monkey.

Startled, the tourist went over to the shopkeeper and said. "That was a very expensive monkey. Most of them are only few hundred dollars. Why did that one cost so much?"

The Shopkeeper answered, "Ah, that monkey can draw in AutoCAD - very fast, clear layouts, no mistakes, well worth the money."

The tourist looked at a monkey in another cage. "That one's even more expensive! $10, 000! What does it do?"

"Oh, that one's a Design monkey; it can design systems, layout projects, mark-up drawings, write specifications, some even calculate. All the really useful stuff," said the shopkeeper.

The tourist looked around for a little longer and saw a third monkey in a cage of its own. The price tag around its neck read $50, 000. He gasped to the shopkeeper, "That one costs more than all the others put together! What on earth does it do?"

The shopkeeper replied, "Well, I haven't actually seen it do anything, but it says it's a project manager."

About The Embedded Muse

The Embedded Muse is a newsletter sent via email by Jack Ganssle. Send complaints, comments, and contributions to me 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.