
Embedded Muse 181 Copyright 2009 TGG September 23, 2009
Permanent link to this issue: www.ganssle.com/tem/tem181.pdf

EDITOR: Jack Ganssle, jack@ganssle.com

CONTENTS:

Editor's Notes

Are you happy with your bug rates? If not, what are you doing about it? Are you asked to do more with less? Deliver faster, with more features? What action are you taking to achieve those goals?

In his book "Quality Is Personal" Harry Roberts shows that "trying harder" won't change anything fundamental. More substantive changes must be made, and in fact it IS possible to accurately schedule a project, meet the deadline, and drastically reduce bugs. Learn how at my Better Firmware Faster class, presented at your facility. See http://www.ganssle.com/classes.htm .

I'll be speaking at the Embedded Systems Conferences in England (October 7-8). See http://esc-uk.techinsightsevents.com/index.php .

Many of you have contributed many great comments about tools you like. Thanks and keep 'em coming! I have posted them, pretty much at random, on my website. But we recently reorganized that part of the site to make it easier to find information about tools – see http://www.ganssle.com/tools.htm .

For many years I've avoided the social networking sites. I had filters in place to ditch email from LinkedIn and all of the other similar web networks. But a lively debate here recently made me rethink that logic. So I've removed the filters, set up a Facebook page (whatever that is), and more. Feel free to start socializing!

Quotes and Thoughts

Paraphrasing George Santayana's "Those who cannot remember the past are condemned to repeat it": Failing to acknowledge and record past success condemns an organization to not to repeat them. (Barry Boehm)

Book Review

Your system has to read a keyboard and update the display. That's pretty easy to handle in a simple loop.

Oh, wait, then there's the A/D converter which needs service once a millisecond. The data is noisy so ten samples must be averaged and the result fed into a computation which is sent to the display. But you can't do the math till the results of the encoder become available, and that can only be read on 20 msec intervals.

But don't forget to monitor the radiation source; if it goes out of limits a safety protocol has to be invoked to avoid harming users. That has to be monitored every 250 milliseconds.

How would one write this code? Sure, it's possible to write an interrupt handler that takes clock ticks and then, via a number of tortured loops, sequences off the proper activities. It'll be tough to debug and harder to maintain. You can be sure the boss will come in, red-faced, wondering why the heck the system only looks at safety parameters every quarter second when any idiot knows the rate should be 0.230 sec, no matter how he wrote the spec. The loops grow more complex and the program ever more convoluted.

This is a very old problem, one solved by the use of a Real-Time Operating System (RTOS). Write each activity as a separate task. The code is simple, crystal clear, and easy to change.

An old problem, yes. But there's surprisingly little written about the use of an RTOS. Jean Labrosse wrote one of the first and best books on the subject: uC/OS. I'm told it, and the subsequent second edition, are the best selling books ever published about embedded systems, and I'm not surprised. Extremely-well written, they covered the subject in depth and with finesse. He wrote using the uC/OS and uC/OS-II RTOSes as examples.

Now Jean and the crew at Micrium have a new and hugely improved version of that RTOS: uC/OS-III. Where uC/OS-II is a commercial quality product, one that even meets the highest safety-critical requirements, uC/OS-III takes that quality and reliability levels to even the most demanding applications.

Jean has supplemented the new RTOS with a new book, uC/OS-III, The Real Time Kernel. It's much weightier than his previous RTOS books as this volume goes deeply in depth into the nuances of using an operating system in real applications. uC/OS-III lays out the rational behind an RTOS, and then in a very logical fashion presents each of the resources provided by an RTOS and how one goes about using those features in a product. Though uC/OS-III is used as an example, it is not presented as the canonical RTOS, and users of any real-time operating system will find this material immensely useable.

I have long counted Jean a friend, and have great respect for his perfectionism. That is clear when reading the uC/OS source code, which is probably the most beautiful code I have read, and, since it has been used in products certified to DO-178B level A, also works!

That perfectionism also manifests itself in this book, in which it's clear he has taken pains to get every fact right, every drawing clear, all while maintaining a very consistent style.

This is a book by an engineer, for engineers (including engineering students). Devoid of fluff, it's packed with information about using an RTOS in a real system… today. What do I need to do to get started? What are all those files? Where is the information I need located?

Are you using an RTOS? If so, read this book. If you're not using one, read this book; not every embedded system needs an operating system, but there are too many that have been cobbled together through the painful use of ad hoc loops that an RTOS would vastly improve.

Tools and Tips

Tim Peterson wrote: "I enjoy the embedded muse greatly. I just wanted to pass along information about a potentially useful podcast. Though it's not embedded or even engineering specific, I felt that the information would be valuable to many. The podcasts Manager Tools and Career Tools can be found at: http://www.manager-tools.com/ ."

Clyde Shappee contributed: "Another very capable schematic tool and PCB package, all in one is Kicad:
http://kicad.sourceforge.net/wiki/index.php/About_KiCad

"It is a nicely integrated tool that keeps getting better!

"I have a colleague who uses it now exclusively (start-up == no money) and has done 5 designs with it with very few complaints or "Gotchya"s."

Naming Conventions

In "Some Studies of Word Abbreviation Behavior" (M.H. Hodge and F.M. Pennington, Journal of Experimental Psychology) researchers had subjects abbreviate words. Other subjects tried to reconstruct the original words. The average success rate was an appalling 67%.

What does "Disp" mean? Is it the noun meaning the display hardware, or is it the verb "to display?" How about "Calc?" That could be percent calcium, calculate or calculus.

With two exceptions never abbreviate a name. Likewise, with the same caveats, never use an acronym. Your jargon may be unknown to some other maintainer, or may have some other meaning. Clarity is our goal!

One exception is the use of industry-standard acronyms and abbreviations like LED, LCD, CRT, UART, etc that pose no confusion.

Another is that it’s fine to use any abbreviation or acronym documented in a dictionary stored in a common header file. For example:

```
/*  Abbreviation Table
* Dsply    == Display (the verb)
* Disp     == Display (our LCD display)
* Tot      == Total
* Calc     == Calculation
* Val      == Value
* MPS      == Meters per second
* Pos      == Position
*/
```

I remember with some wonder when my college physics professor taught us to cheat on exams. If you know the answer’s units it’s often possible to solve a problem correctly just by properly arranging and canceling those units. Given a result that must be in miles per hour, if the only inputs are 10 miles and 2 hours, without even knowing the question it’s a good bet the answer is 5 MPH.

Conversely, ignoring units is a sure road to disaster. Is Descent_Rate meters per second? CM/sec? Furlongs per fortnight? Sure, the programmer who initially computed the result probably knows, but it’s foolish to assume everyone is on the same page. Postfix all physical parameters with the units. Descent_Rate_MPS (note in the dictionary above I defined MPS). Timer_Ticks. ADC_Read_Volts().

Joke for the Week

PAUL REVERE VIRUS:
This revolutionary virus does not horse around. It warns you of impending hard disk attack: Once, if by LAN; twice if by C:

POLITICALLY CORRECT VIRUS:
Never identifies itself as a "virus," but instead refers to itself as an "electronic micro-organism".

ARNOLD SCHWARZENEGGAR VIRUS:
Terminates and stays resident. It'll be back.

FEDERAL BUREAUCRAT VIRUS:
Divides your hard disk into hundreds of little units, each of which does practically nothing, but all of which claim to be the most important part of your computer.

TEXAS VIRUS:
Makes sure that it's bigger than any other file.

ADAM AND EVE VIRUS:
Takes a couple bytes out of your Apple.

AIRLINE LUGGAGE VIRUS:
You're in Dallas, but your data is in Singapore.

FREUDIAN VIRUS:
Your computer becomes obsessed with marrying its own motherboard.

STAR TREK VIRUS:
Invades your system in places where no virus has gone before.

HEALTH CARE VIRUS:
Tests your system for a day, finds nothing wrong, and sends you a bill for $4,500.

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.