The Embedded Muse 70

Editor: Jack Ganssle (jack@ganssle.com) February 1, 2002

Asynchronous Firmware

I was in the Midwest last week talking to a small medical instrumentation company. The developers were a lot of fun, lively and very proud of their work. One questioned my long-held belief that there are no such things as glitches; he felt that "stuff happens" - sometimes mysterious events occur that can create havoc.

To illustrate his point he talked about an 8 bit parallel position encoder they use. Sometimes the normally monotonically-increasing data jump to bizarre values. Reading the encoder again almost always results in correct data.

"Sure, the encoder could be flaky," I mused, but the engineer assured me that a logic analyzer proved it's output was always reasonable. We looked at the hardware, and found that, oddly, the input byte is distributed across two ports, four bits on each. No real problem there, of course; just read port A and then port B, and combine the two values.

But this is a very dangerous practice. The encoder is always rotating, and it takes a few microseconds to read the pair of ports, so each read collects data when the encoder is in a slightly different position. Consider:

- Read the low port. The data is 1111
- The encoder's output flips from 00001111 to 00010000
- Read the high port. We get 0001.

Combining the two discrete samples yields a value of 00011111, almost twice the actual value. Read it again and odds are the data will be OK. We've just seen what could be called a glitch, but is in fact an artifact of a software bug.

In a pathological case you might get a value that is so out-of-bounds that the system crashes – say, if the data is used (unchecked) to index into an array.

I see this sort of problem a lot. Usually it's related to timers. A lot of systems extend a (say) 16 bit hardware timer to 32 bits by using an ISR that counts overflows. When a routine needs the elapsed time, it calls a read_timer routine that concatenates the integer in memory (the "high time" value) with the current contents of the 16 bit counter. This is a seductively simple approach that just does not work reliably. It's sure to occasionally fail, for the same reason as the encoder example just cited.

My Midwestern friends were delighted yet horrified to understand the problem. A hardware engineer suggested adding a latch, an 8 bit register between the encoder and the two input ports. When the code wants encoder data it asserts a "latch the data now" signal which clocks position data into the register. That, of course, cannot work either because of the problems of hardware metastability. The clock and the encoder are necessarily asynchronous, which insures rare but occasional latching of totally random values. (For more on metastability see TI's report number SDYA006: "Metastable Response in 5-V Logic Circuits").

I've written much more about these issues and their solutions in Embedded Systems Programming magazine. See http://embedded.com/story/OEG20010615S0111 and http://embedded.com/story/OEG20010718S0067 for more details.

A few quick code changes (much like detailed in the articles just referenced) solved the company's encoder problems.

Thought for the Week

Peter Cahan sent in this gem:

Start with a cage containing five monkeys. Inside the cage, hang a banana on a string and place a set of stairs under it. Before long, a monkey will go to the stairs and start to climb towards the banana. As soon as he touches the stairs, spray all of the other monkeys with cold water. After a while, another monkey makes an attempt with the same result – all the other monkeys are sprayed with cold water. Pretty soon, when another monkey tries to climb the stairs, the other monkeys will try to prevent it.

Now, put away the cold water. Remove one monkey from the cage and replace it with a new one. The new monkey sees the banana and wants to climb the stairs. To his surprise and horror, all of the other monkeys attack him. After another attempt and attack, he knows that if he tries to climb the stairs, he will be assaulted.

Next, remove another of the original five monkeys and replace it with a new one. The newcomer goes to the stairs and is attacked. The previous newcomer takes part in the punishment with enthusiasm! Likewise, replace a third original monkey with a new one, then a fourth, then the fifth.

Every time the newest monkey takes to the stairs, he is attacked. Most of the monkeys that are beating him have no idea why they were not permitted to climb the stairs or why they are participating in the beating of the newest monkey.

After replacing all the original monkeys, none of the remaining monkeys have ever been sprayed with cold water. Nevertheless, no monkey ever again approaches the stairs to try for the banana.

Why not?

Because as far as they know that's the way it's always been done around here.

And that, my friends, is how company policy begins.

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.