The Embedded Muse 55

Editor: Jack Ganssle (jack@ganssle.com) November 8, 2000

Tires That are Too Smart

From: http://dailynews.yahoo.com/h/nm/20001023/tc/tires_phones_dc_1.html

Reuters reports that in Finland work is afoot to embed chips inside of your car's tires. When the pressure falls too low they'll call you on your cell phone.

Technology run amok? Can I block calls from my tires? I'll have to get call waiting in case the tires want to break in on another conversation!

More on Measuring Latency

Gary Bergstrom wrote:
The other very interesting thing one could do with your counter is to log the time when the ISR finished. e.g. Read the hardware timer as the last thing you do. This catches overflows, and sometimes the higher priority interrupt that interrupts the current interrupt.

And Alf Katz commented:
Of course, you're right, but of much more interest to most embedded programmers is the longest system latency. This can easily be found by keeping the highest overrun reached in another variable, and updating only when a higher one is found. Of course you could go overboard & collect means & standard deviations of the latency time, but you wouldn't want to do it in an interrupt - and the longest latency is *usually* the most important metric.

Finally, from Dean TerHaar:
I use a similar approach to time the duration of an ISR. I define a struct of two variables: TimerIn and TimerOut, and then declare a global array of this struct. On entry to the ISR I read the system timer and store it in Array[i].TimerIn and on exit from the ISR I read the system timer again and store it in Array[i].TimerOut. I also have a control variable that I can set dynamically to enable/disable this ISR duration monitor. After I execute a sample run of the program (in real time), I'm use these timer values to calculate the min,

max, and average durations of the ISR and display a report. (without the aid of a debugger or oscilloscope!)

The routine that calculates the ISR time also calculates the overhead of the duration test (i.e. the time it takes to read the system timer and store its value). It subtracts this overhead (or bias) from the calculated ISR time to compensate for the additional time it takes to perform the test. This routine also normalizes the result in microseconds and handles cases where the system timer rolled over during the execution of the ISR under test.

This test, as it is written in C, however, does not account for the overhead of the ISR under test (working register pushes and pops), but this fixed time could be calculated manually and included into the final result returned by the routine that calculates the ISR execution time.

Thought for the Week

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 shop keeper, "I'll have a C monkey please".

The shopkeeper nodded, went over to a cage at the side of the shop and took out a monkey. He fit 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 a few hundred dollars. Why did it cost so much?"

The shopkeeper answered, "Ah, that monkey can program in C - very fast, tight code, no bugs, well worth the money."

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

"Oh, that one's a C++ monkey; it can manage

object-oriented programming, Visual C++, even some Java. 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 consultant."

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.