

## Re: Self restarting property of RTOS–How it works?

**Source:** <http://coding.derkeiler.com/Archive/General/comp.arch.embedded/2005-02/0806.html>

---

**From:** CBFalconer (*cbfalconer\_at\_yahoo.com*)

**Date:** 02/12/05

Date: Sat, 12 Feb 2005 18:47:39 GMT

Ed Beroset wrote:

> *Del Cecchi wrote:*

>> *Ed Beroset wrote:*

>>

>>> *I have also noticed that the programmers from a computer science*

>>> *background tend to be much better at working out a system*

>>> *architecture and planning first.*

> [...]

>>>

>> *Those comp-sci geniuses are the ones that gave us a software*

>> *paradigm that is susceptible to attacks as simple as buffer*

>> *overruns, and store data in randomly scattered chunks linked by*

>> *pointers. And put multiple unrelated locks in the same cache*

>> *line? That the ones you are talking about?*

>

> *It's interesting to learn that no engineers were ever involved in*  
> *building such flaws.*

>

> *My background happens to be more in the engineering than the*

> *computer science end of things, but I don't share your evident*

> *contempt for the field. Here's an example: An embedded*

> *communication system receives packet-based messages of varying*

> *lengths at an average rate of 100 packets per minute, but*

> *asynchronously. Because the system also checks its timing*

> *against the recovered clock from the messages, which it can*

> *easily keep synchronized within limits as long as it doesn't go*

> *too long without receiving a packet. What is the probability*

> *that no packets will arrive in an interval of five seconds?*

>

> *I can answer that question easily because I've studied a little*

> *computer science. Can you? If not, how can you properly*

> *engineer the system?*

If its internal clock can't stay synchronized over 5 seconds, or even much longer, I think there is something wrong with the hardware design. Of course you haven't defined synchronized. I

comp.arch.embedded: Re: Self restarting property of RTOS–How it works?

certainly couldn't answer it, but I would know enough to hunt up queueing theory, which is quite mature and predates computers. Whatever the synchronizing requires, I would attempt to put something in the transmitter system to ensure satisfaction. Statistics can always burn you.

But you are asking the wrong question. However, if you asked what is the probability that 10 packets will arrive in 5 seconds, you would have a good point. Again, the place to look is queueing theory. I do know that the design is going to require some sort of buffering, and if there is nothing else critical and resources are pre-established I will assign as much buffer space as possible (assuming no other similar requirements) and not bother with the details.

--

"If you want to post a followup via groups.google.com, don't use the broken "Reply" link at the bottom of the article. Click on "show options" at the top of the article, then click on the "Reply" at the bottom of the article headers." - Keith Thompson