

## Re: Disobeying jet engines – why?

---

*Source:* <http://sci.tech-archive.net/Archive/sci.electronics.design/2008-01/msg04224.html>

---

- *From:* Terry Given <[my\\_name@xxxxxxx](mailto:my_name@xxxxxxx)>
  - *Date:* Sun, 27 Jan 2008 22:44:59 +1300
- 

John Larkin wrote:

On Fri, 25 Jan 2008 15:38:15 +0000, Martin Brown  
<[newsam@nezumi.demon.co.uk](mailto:newsam@nezumi.demon.co.uk)> wrote:

I do embedded realtime apps that have no OS. I've written a few RTOS's, but haven't needed to use one in a long time.

If the code is...

```
START: do thing1
do thing2
do thing3
goto START
```

where is the OS?

The OS is the bit that should step in when one of thing1, thing2 or thing3 fails to complete. Most non-trivial embedded applications have some kind of RT kernel underpinning the allocation of resources.

I doubt it. You can do arbitrarily non-trivial apps with just state machines running in a loop. The only RTOS would then be a hardware watchdog timer.

The code blocks can themselves use whatever resources they want, assuming they are neither hostile nor stupid. The rare allocation-of-resources situation can be handled by code blocks locking out interrupts, or with simple flags.

If ever you assign a programmer to do a deep-embedded app, and he says "first, we'll have to write/buy an rtos", fire him instantly.

Re: Disobeying jet engines – why?

John

LOL! I had a programmer write code for a user interface module. 4x20 LCD, 3 LEDs, 5 buttons and a MODBUS interface (it was a master). Used a 8951 with 64kb flash and 4kb SRAM. He wrote an RTOS, which wasn't a bad start, although not really necessary. but ran out of RAM, and used most of the FLASH, which was supposed to store drive config data – I expected the code to take 4kb max.

turns out his C compiler was using 32-bit numbers for BOOLEANS! and it got a lot worse. It was a really fancy RTOS, multi-threaded & multi-tasking. And got binned. I learned a lot about writing clear specs from that debacle. The replacement programmer got all the code in 8kb including the GNU real-time debugger. and used a fraction of the RAM.

Cheers  
Terry

.