The Embedded Muse 71

Editor: Jack Ganssle (jack@ganssle.com) February 22, 2002

ESD and Firmware

Long time friend and well-known firmware consultant Scott Rosenthal (www.sltf.com) was working with TI's MSP430 microcontroller recently and ran into some interesting problems.

Electrostatic Discharges (ESD) are those high voltage low amperage "zaps" that happen when we shuffle our feet along a carpet and then touch something metallic. Since practically everything is now electronic, regulatory bodies require that most products be resistant to ESD discharges. Our customers would be pretty upset if a casual static zap destroyed their high tech device! Silicon geometries are now so small that tens of volts can destroy the transistors… yet an ESD pulse is often tens of thousands of volts.

Careful grounding and a well-designed case will minimize most of these problems. But it's tough to eliminate it altogether; often some energy will couple into your circuits, no matter how well shielded.

Scott found that the MSP430's I/O pins, when subjected to just a bit of ESD, randomly change state. This is a pretty cool MPU, since every peripheral bit can be an input or output port, or even an interrupt input. He watched the ESD testing lab's experiments change bits from output ports to interrupt inputs, leading to erroneous and erratic interrupts. Though the parts survived the ESD events, the mode changes could horribly crash the firmware, a very bad thing for critical applications.

An A/D converter also suffered from temporary brain damage when zapped.

The solution was software that monitors the settings of all I/O pins and the A/D's setup and calibration. Additional code rejected spurious interrupts.

How reliable do our systems have to be? If we have to survive an ESD discharge without crashing does the entire unit have to live in a Faraday shield, one that blocks out all RF and ESD energy? If a system has external inputs they may be hard to impossible to completely protect. Optical isolators help, but are an expensive addition to a low cost product.

By adding software that monitors the hardware's health Scott cured the problem. But he raises an interesting philosophical issue: can we no longer assume the hardware is reliable? I wrote about unreliable hardware in Embedded Systems Programming this month (http://embedded.com/story/OEG20020125S0104), but ESD is another source of erratic problems.

Watchdog timers have always been our last-ditch defense against crashing code. If you're worried about transient hardware issues, as it's beginning to seem we must, a well-implemented watchdog is essential. If we cannot trust the hardware, that suggests the watchdog must hit the CPU's reset input (not an interrupt), since only reset is guaranteed to cure the processor of all weird modes.

Some safety critical code doesn't even assume that RAM and ROM work; the systems continuously run RAM and ROM tests in the background, looking for bit flips or part failures.

High reliability systems will benefit from a healthy dose of paranoia. Assume everything breaks and nothing is reliable. I saw a system recently where 80% of the code was for exception handling and hardware monitoring. Think of it – for every line of code that implemented the application the developers wrote four lines of paranoia-code. The costs are astronomical, though probably necessary for some systems. I suspect that as time goes on, as our software and hardware complexity increases, this sort of defensive programming will become the norm.

Thought for the Week

Thanks to Laurence Marks who found this.

System Crash (to the tune of "The Monster Mash")

I was working in the lab, late one night
When my eyes beheld an eerie sight,
Some smoke from our VAX began to rise
And suddenly, to my surprise...

[chorus]
(There was a crash) There was a system crash
(A mighty crash) I heard the disk heads smash
(A system crash) It came down in a flash
(There was a crash) A fatal system crash

The lab manager then appeared from his room,

Said "I don't want to be a prophet of doom,
But we had one like this just the other day
Which blew up 4 megs and the SBA"

[chorus]

The system had been booted,
diagnostics all run through,
When a power flux made it run amuck,
then SCOTTY and IRVING blew too

So we'd lost all our VAXES in less than one night
When a VP came in and said "hey, that's all right,
I'll loan you a Venus - here's what to do
When you call up Support, tell them Gordon sent you..."

[chorus]

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.