

The following is an excerpt from an article published in Embedded Systems Programming, April 1999.

Nonvolatile RAM

by Jack G. Ganssle

...

Testing

Good hardware and firmware design lead to reliable systems. You won't know for sure, though, if your device really meets design goals without an extensive test program. Modern embedded systems are just too complex, with too much hard-to-model hardware / firmware interaction, to expect reliability without realistic testing.

This means you've got to pound on the product, and look for every possible failure mode. If you've written code to preserve variables around brownouts and loss of Vcc, and don't conduct a meaningful test of that code, you'll probably ship a subtly broken product.

In the past I've hired teenagers to mindlessly and endlessly flip the power switch on and off, logging the number of cycles and the number of times the system properly comes to life. Though I do believe in bringing youngsters into the engineering labs to expose them to the cool parts of our profession, sentencing them to mindless work is a sure way to convince them to become lawyers rather than techies.

Better, automate the tests. The *Poc-it*, from MicroTools (www.poc-it.net), is an indispensable \$250 device for testing power-fail circuits and code. It's also a pretty fine way to find uninitialized variables, as well as isolating hard-to-initialize hardware devices like some FPGAs.

The *Poc-it* brainlessly turns your system on and off, counting the number of cycles. Another counter logs the number of times a logic signal asserts after power comes on. So, add a bit of test code to your firmware to drive a bit up when (and if) the system properly comes to life. Set the *Poc-it* up to run for a day or a month; come back and see if the number of power cycles is exactly equal to the number of successful assertions of the logic bit. Anything other than equality means something is dreadfully wrong.

....