

In the days of old, when engineers were bold



In the 1970s, around the time that MOS Technology introduced the 8-bit 6502 processor, I took a job as chief engineer at a company that developed, manufactured, and supplied military and industrial security-monitoring systems. Job 1 was an intermittent problem in the current model that caused the watchdog timer to reset the system. The default condition for a system reset was to secure all doors and sensors throughout the monitored areas to condition “red.”

Then, when a normally “green” door opened, alarms sounded. This problem was a particularly disturbing one because this equipment was monitoring sensitive areas, such as nuclear-storage facilities in which the false alarms could cause response teams to react.

The 6502-based equipment had the usual ROM, RAM, I/O, and a built-in small Seiko line printer (Reference 1). In those days, we tried to make one little processor do everything; why not? Popular theory at the time was “it is a hardware problem,” and there was good reason to believe it. First, the design was poor and had mixed-logic families on a poorly laid out bus. It also

had some ground bounce, because these were the days before multilayer boards were common. Worst of all, we were just learning about the “dirty dozen,” 12 illegal instructions for the 6502 that, if executed, would cause the processor to halt. We used to call it “halt and catch fire”; some of these instructions required a power-down and backup to clear. After three months of many ECOs (engineering-change orders) to correct all the hardware deficiencies, the problem had not changed at all in nature. Now, with solid hardware, we were really scratching our heads.

There was one thing about the design that rubbed me the wrong way

right from the beginning: the fact that the little Seiko line printer was tied to the NMI (nonmaskable interrupt). The 6502 controlled the printer, and the NMI monitored the synchronous pulse from the printer. Now that hardware was looking good, there was some finger-pointing over whether the problem now was hardware or software. After many hours in the lab and reports from customers, we narrowed the behavior for the intermittent failure as occurring only after the printer was winding down from printing.

After yet another long day in the lab, the NMI was still bothering me, especially now that I knew that the printer was involved. I began thinking about the 6820 PIA (peripheral-interface adapter) and the initialization sequence; then, it hit me. It was so simple: The 6820 PIA takes two instructions executed in sequence to initialize it. You cannot interrupt the instructions. I drove back to the office, and the software guys were still there. I checked, and, sure enough, when the printer was done printing, they went out and reinitialized the PIA just for grins!

For me, it was so obvious and “case closed.” The short story is that I had them take out two lines of code, and we never saw the “hardware” problem again. The long story is that management and software people spent many more days testing because they just could not believe it. After much more testing, the senior programmer and I flew to one of our main problem locations and installed two new ROMs; no further problems occurred. **EDN**

REFERENCE

1 http://en.wikipedia.org/wiki/MOS_Technology_6502.

Lynn Smith was a consultant and hardware engineer in Silicon Valley for 30 years before recently pulling up stakes in San Jose, CA, to move to Alabama, where he is building an airpark. Like Lynn, you can share your Tales from the Cube and receive \$200. Contact Maury Wright at mwright@edn.com.