

“Not I,” said the rat: the tale of the Little Red Hen-gineer



Once upon a time, my boss handed me what I thought was a simple assignment: Build a receiver/demodulator for a multichannel PPM (pulse-position-modulated) data stream from a remote sensor. The sensor was a custom-built unit; I felt fortunate because the design was well-documented and we had spare units that I could play with. I soon found, however, that the sensor’s designer had derived each channel’s information pulses from the system’s master clock, which was many times higher in frequency than the sensor’s data signal.

Because of this discrepancy, the information pulses were almost nonexistent, which caused problems with synchronization and decoding. It also made it next to impossible to see them on the oscilloscope, even with the brightness turned all the way up. So I asked my co-workers for help.

“You have too little energy in your signal,” said my supervisor.

“Your pulses are too narrow,” said my fellow lab-bench rat.

“Let me see the transmitter controller’s source code,” said the new hire. “That will tell me how to program the receiver.”

I spent the next several days con-

sidering their comments until I realized they were either pointing out problems I had already seen or redesigning parts of the system that had nothing to do with the problems. So I went ahead with my design. When I asked for their opinions again, they responded:

“That preamp won’t work. Use a comparator,” said my supervisor.

“That filter will clip your pulses too much,” said my fellow lab-bench rat.

“Where’s the microcontroller?” asked the new hire. “How did you program this thing?”

I went back through my notes and diagrams, wondering whether I had missed something. I made changes,

changed them back, tried different approaches, argued, explained, and then sat down and assembled a breadboard model. When they saw it, their comments were:

“It falls out of sync too easily,” said my supervisor.

“Your error rate’s too high,” said my fellow lab-bench rat.

“You don’t need all that stuff,” said the new hire. “Here, I’ll do a simulation. Then you can program a microcontroller to take care of your problems. This one’s perfect for this job.”

By now I was really exasperated, so I put the demodulator aside and worked on other projects. Then, late one night, the answer came to me: Change the modulation from pulse position to pulse width; it would add only a couple of flip-flops and some logic. In that way, I had an approximate square-wave output in the synchronous channel and longer stable voltage levels for the data channels. That combination meant there would be plenty of signal amplitude for synchronization and more setup time for the data registers. It also simplified my filter design.

When I showed my co-workers my now-working-correctly circuit, they had plenty of comments:

“So you took my advice,” said my supervisor. “Be sure you spell all of our names correctly on your write-up. Oh, and be sure you log the time you spent on it this week.”

“I told you it would work,” said my fellow lab-bench rat. “Say, can you look at my data synchronizer? My problem’s similar to yours.”

“A microcontroller would’ve done it better,” the new hire grumbled. “My stereo’s doing something strange; would you mind taking a look at it?”

Well, I did learn something from this situation: It sometimes pays to shut off all the extra noise—um, information—and handle things your own way. I’m still having trouble, though, setting a proper noise threshold. **EDN**

Steve Lubs has been an engineer in a variety of roles at the Defense Department for 30 years.