Hot Technology Definitions

Have you seen the hot new technology dictionary? It's full of great new definitions. Look up "sadist," for example, and you'll get, "A software engineer or embedded designer who insists on writing applications in Assembler or some other arcane low-level code." Boy, I cannot believe guys still do that, but I've spoken to some. "You've gotta stay low to get the performance," one told me. "I must become one with my code," waxed another, "or I won't reach application Nirvana."

OK. I get it. We all want to rise to a higher plane—in life and in work. We all need to be in the right frame of mind to get the best results, whatever we're doing. Clearly, assembly code just cannot get us there in an era of tight—and I mean TIGHT--schedules. Wondering what will, I looked up "framework." Here's what I got:

"A Reference Framework can put you in the right state of mind for application design by taking you away from low-level coding and raising you to higher application planes, where you can design-in your feature set and differentiate your product in jig time."

If you don't believe me, ask Steve Paulsen, the author of our cover story. In Paulsen's experience, the right Reference Framework not only can slash application design time, it can give you the flexibility you need to shape the application any way you want.

The idea of a ready-made, production-quality framework is that you can provide your customers with reference designs without duplicating framework development for each algorithm combination.

Naturally, all frameworks are not alike. But select one that's designed for easy customization, Paulsen says, and you can focus on integration, not framework development, thereby banking substantial design and test time. Also, he cautions, not all frameworks can accommodate your required customization. Nor will designs necessarily be reliable if you have to force a framework to provide functionality outside its original design criterion.

In the cover story, Paulsen shows, step by step, how Imagine Technology took advantage of the Texas Instruments multilevel Reference Framework (RF). Many benefits accrued, but the most dramatic one was a huge 75-percent lopping off of the development time of a production solution.

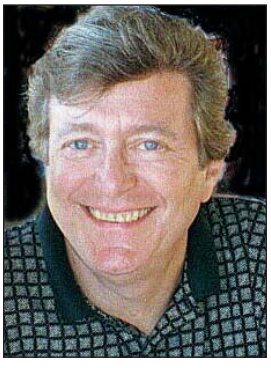
While flipping through that terrifically with-it dictionary (don't ask me where to get one; they're very hard to find), I came upon this marvelous definition of "trendy": "Using a dual-processor architecture to combine nifty applications, such as digital audio."

What a coincidence, I thought, because that's exactly the subject of one of our contributed articles.

Written by Messrs. Cohrs, Powell and Williams, senior software engineers at Indesign, LLC, the piece shows you how to tap the virtues of a DSP to add digital audio to your host application running on a high-end processor. It's fairly simple, the authors say, if you conduct a thorough analysis up front, take into account certain design constraints before you begin, and pay attention to details during the integration.

With the DSP serving as a digital-audio engine focusing on the key algorithms, there will be minimal impact to an existing design, and the logical separation will provide for modular development, improved maintenance, and easier enhancement capabilities. All the details are inside.

One definition I've always had trouble with is the one for "bootstrap." Anyone remember the bootstrap circuit of yore? I never got how it worked. Nor did I get the meaning of "boot loader" in the primal days of PCs. So you can imagine my distress at hearing that word again, this time with respect to embedded software design.



Thank goodness. I've finally learned the meaning of the word, thanks to Stanford Hudson, an embedded software technologist at Tekgenix Corp., who not only goes into the basics of boot loaders in our third contributed article, but also shows how to embed one in a DSP application.

Now I can concentrate on reading the dictionary.

—Stan Runyon
testman2@earthlink.net