Letters to the Editor

Allow me to add one more voice to the maelstrom of debate over how to popularize Forth, and to make three observations based on my experiences as a freelance hardware/software designer.

Observation #1: No one will try Forth because of productivity. I've become an enthusiastic proponent of Forth because no other language offers me its programming productivity and expressive power. But I was originally attracted to Forth in the 70's for a different reason: what it enabled me to do with my then-limited resources.

Face it: no one believes the claims of expressiveness and productivity until they've used Forth for a few years. This won't sell Forth. "Me too" won't sell Forth either. Our biggest selling points are economy, and suitability for programming on the "bare metal."

Perhaps this is why I've never been able to sell a computer programmer on using Forth, but I've engendered lots of interest among electronic engineers.

Observation #2: The days of limited resources are not over. Sure, the dropping cost of CPUs and memory has led to a market where millions of systems have a 32-bit engine and 1 meg of RAM. But it has also created a market of tens of millions of systems with 8-bit CPUs and a few K of memory, and no one is addressing the needs of their programmers! Most of these truly "micro" systems are programmed in assembly language, by default.

For every General Motors, there are hundreds of million-dollar companies putting microprocessors into their product line. These are companies who can barely afford an IBM PC/AT, and are lucky if they have one programmer on staff. They have to design cheap and simple.

Observation #3: Forth's tools for embedded programming are woefully inadequate. A case in point: I'm currently developing a control system using a multiprocessor Zilog Super8 architecture. Undoubtedly, like the previous four Super8 products I've done for this client, it will be programmed in assembler. This is not at their request; they're open to Forth. I, the Forth advocate, have made the decision that our assembler tools are better for this job.

The individual Super8s won't support resident development. To use Forth, we would have to revert to the old "edit, crosscompile, download, test" cycle... making Forth little, if any, better than C. At least C and the Zilog assembler are designed with this in mind; the Forths I've used treat this as a poor cousin, an adjunct to resident development.

Even more important: I cannot leave my client to maintain their software with a set of tools that only I, or another Forth guru, can understand!

I understand chipFORTH is addressing these problems. But even if a Super8 chipFORTH were available, it would be too expensive, and I suspect too obscure, for my client to use. I have a Super8 Forth, and my own metacompiler that I'm trying to upgrade for embedded programming and document for a neophyte; perhaps Forth may yet invade this product.

I think the efforts to push Forth into bigger and bigger systems are misguided. Our future lies not in persuading Ashton-Tate to write dBase VII in Forth, but in persuading hundreds of small companies to put Forth into their cost-sensitive widgets. To do this, we need more cheap Forths for today's embedded chips: 8051, 6801, Z8, TMS320, to name a few. And we need tools which are designed for the embedded environment, which a beginner can use. It's not enough that Forth's virtues become evident after years of use; we must offer immediate and obvious value to Forth's prospective users.

Sincerely,

Brad Rodriguez
T-Recursive Technology
Toronto, Ontario