Embedded Control as a Path to Forth Acceptance

PHILIP KOOPMAN JR., Ph.D.
Harris Semiconductor, 2525A Wexford Run Rd., Wexford, PA 15090
(Internet) koopman@greyhound.ece.cmu.edu

ABSTRACT:
This paper presents a strategy for promoting Forth acceptance based on a narrow focus of concentration on a target application area. The proposed goal is to make Forth the language of choice for embedded real time control systems, and thus establish a foothold in the general computing community.

INTRODUCTION

The opinions in this article are those of the author, and do not necessarily reflect the views of Harris Semiconductor.

Discussions on the USENET Forth bulletin board and other arenas often concentrate on the theme of improving the acceptance of Forth among those not currently using it. I propose a strategy to accomplish this, which involves striving to make Forth the *language of choice* for embedded real time control. This essay is meant to stir up discussion in the community with where we are really going, and focus attention on why Forth should become more popular, and how this may be done. It is intended as a starting point in a continuing discussion, not as a final word on the subject.

WHY WORRY ABOUT FORTH POPULARITY?

Why are any of us worried about the popularity (or unpopularity) of programming in Forth? I would categorize the reasons for desiring that Forth become popular as follows:

- **A belief that Forth is a fundamentally better tool to solve problems.** Many of us believe that Forth is a fundamentally better way to solve problems. There are advantages to the interactive, incremental compilation environment used by most Forths. Many of us would simply like to see Forth in more widespread usage because we believe it is better.
- **Using Forth at the workplace.** Many of us write programs in Forth as a hobby or for minor projects, yet are unable to use Forth in our regular jobs. Others can use Forth for production programming, but only under unusual circumstances, usually involving tight deadlines or naive supervisors. One serious obstacle to using Forth is the lack of trained Forth programmers to maintain code. If Forth were a more socially acceptable language, such obstacles would be reduced and we could all use our favorite programming language for work as well as play.
- **Language support and development.** If Forth were to come into wider use, greater support would be available for it. The Forth market is small, so commercial Forth vendors can only afford to put relatively little effort into development and maintenance (compared to the resources used to support something like an Ada compiler). Even though Forth environments traditionally need much less support and maintenance than environments for other languages, the involvement of an order of magnitude more active Forth programmers would surely make available better tools, utilities, and environments.

- **Ego.** Many ardent Forth advocates want to eventually receive praise for their wisdom and foresight in choosing Forth as their favorite programming language. Forth programmers tend to be the best programmers (just ask them!).

But, how compelling are these reasons? Are they sufficient to justify spending time and effort winning wider acceptance for the language? Before we begin the attack we should satisfy ourselves that Forth is worth fighting for. Gut feel and emotion are not enough for this; we must understand why we want Forth to succeed before we commit ourselves to the goal of having it succeed.

THE IMPORTANCE OF BEING A LANGUAGE OF CHOICE

Forth is not a major force in the programming language scene. It is not taught in most schools. It is poorly represented in or absent from the computer section of bookstores. It is seldom presented at computer science and computer engineering conferences. It is not openly used in most companies. It is not used by a significant number of programmers in most disciplines. Yet, this lack of use is certainly not because of a lack of inexpensive software, nor any other simple reason.

There is a long litany of reasons why Forth isn't used by more people. I won't indulge in an enumeration of why Forth is not popular now, since the list is well known. Most of the problems boil down to a lack of professional-quality software development tools, lack of appreciation for the strengths and weaknesses of the language for different application areas, and the fact that it isn't already popular. (Most managers have a well-founded fear of using novel technology to solve problems that can be solved using familiar tools.)

One of the strengths of Forth is that it can be used in many applications. Since it is flexible and extensible, the language itself can be modified to incorporate many requirements. One of the schools of thought to gain greater acceptance for Forth is to extend the language so that it comes ready-made to solve any problem. Then, potential users will see the power of the language revealed, and will use it for all their programming needs.

This approach of developing "fat" Forth systems will not gain acceptance to Forth. Don't get me wrong — this is not to say that such Forth systems are bad or useless, but rather to say that they are not the means to the end of Forth acceptance. The primary problem is not one of whether or not particular Forth standards or systems are "fat" or "thin". It is not what standard a Forth conforms to. It is not any technical reason at all.

The problem is one of marketing.

Ask almost any programmer what language is the best to use for numerical applications. The answer is FORTRAN. The best programming language for UNIX-based applications is C. The best programming language for exploring language implementation techniques and artificial intelligence is LISP. The best programming language for business programs on IBM mainframes is COBOL. The best programming language for a neophyte with a personal computer is BASIC (or, perhaps, LOGO for youngsters). Outside the Forth community, these statements will generally receive little argument.

Why can we say something like FORTRAN is the "best" language for numerical applications and find almost universal agreement? Because FORTRAN is the *language of choice* for those numerical applications. It is not necessarily the provably best language. It is the one that most people use. This means that there are libraries, development environments, programmers, and massive amounts of installed code which all presuppose the use of FORTRAN for certain classes of scientific and engineering applications.

Why not make Forth the *language of choice* for embedded real time control applications? This would create an installed base of applications

and programmers. It would also give an air of acceptability to Forth not only for embedded systems, but for other application areas as well.

The key is to market Forth as the best solution available to a restricted set of applications, and not as just a good solution to everyone's problems. No one will believe that one language can do it all. But, the idea of a flexible language which is well adapted to a single class of applications has considerable appeal. This is not to say that Forth shouldn't continue to evolve to support non-embedded applications, but rather that such evolution should not compromise Forth's abilities in its major area of strength.

EMBEDDED CONTROL AS A TARGET APPLICATION AREA

There is a vacuum in the embedded control marketplace. Until recently, almost all programming in the small embedded control systems (*e.g.* 8051 and other 8-bit microcontroller systems) was done in assembly language. Now, with the advent of more powerful 16- and 32-bit controller chips, that is changing. More and more programmers are using or evaluating high level languages to ease the burden of software development.

Embedded control systems are the ideal application area for which to make Forth the language of choice. Their tightly constrained environments with demanding response requirements are ideally suited to the capabilities of classical Forth systems.

Forth has always had a strong foothold in embedded systems. But, it has never been used by a majority. Because there is a vacuum and no consensus, a wide variety of programming languages are being pressed into service. The dominant language is C. This is not because C is inherently better for this application (or even, for that matter, very good at all). It is because that is the language most programmers are familiar with, having been trained in its use by universities and workstation-based programming projects.

If events are left to progress by themselves, C will become the language of choice for embedded control. Most of the user community will be happy with this, since C will be usable for many applications. Managers will be happy, since C has a lower risk and lower perceived cost than Forth. It always seems easier to pay the deferred, intangible costs for suboptimal programming environments than the immediate and concrete costs of programmer training to switch to a new language.

Large vendors of embedded control products and systems are not likely to change all this. They see the trend to using C, and will fall in line with the results of the market surveys and polls. Even makers of so-called "Forth chips", which includes but is certainly not limited to Harris, will probably support Forth just as an alternative language. Successful companies will have to spend most of their time and money supporting C to please their large number of C-based customers. C may well become the most used language by popular demand, even if it makes suboptimal use of the hardware.

A CHALLENGE

The reality of the situation is that the spark for making Forth the de facto standard language for embedded control will have to come from within the Forth community itself. Vendors and customers, if convinced of a trend towards Forth, will probably jump onto the bandwagon. But, someone has to build the bandwagon, fuel it, and pilot it. The members of the Forth interest groups are probably the only ones who can do this.

What is needed is a concerted marketing effort to promote Forth as the language of choice for embedded control, and in particular real time control. This effort must incorporate articles in major periodicals (especially application-based articles), educational campaigns, and participation in mainstream events. For the purposes of Forth acceptance, one paper in a general computer application conference is worth two dozen at SIGForth, Rochester or

Asilomar. One article in Byte, IEEE Computer or EDN is worth an issue or two of Forth Dimensions. One application written in Forth in a company previously using C is worth a staff full of FIG members at a Silicon Valley company (or a staff full of SIGForth members at a Texas company). The point is to increase visibility, and make Forth look respectable to the outside world.

If we spend time and resources trying to convince programmers in the UNIX world that they should be using Forth to write their window programs, or programmers on supercomputers that they should adapt LINPACK to run in a Forth environment, or developing do-it-all Forth systems in hopes that outsiders will be impressed based on technical merit alone, we are wasting our time. And, we have little time to waste. The window of opportunity for embedded systems is closing fast, driven by the optimizing C compilers and fancy development environments of the 32-bit RISC processors. As this window closes, not only will we lose our chance to make Forth the language of choice, but we may well lose the ability to sell Forth to most customers for these tasks at all. So, we need to act now, or resign ourselves to using a language that will forever be associated with APL and SNOBOL: neat ideas that never really caught on, but seem to never quite die either.

None of this is meant to say that steps in this direction have not been taken in the past, nor to say that many of the facets of a successful Forth promotion are not already in place. However, the Forth community is internally fractured (or perhaps it was never unified). What is needed is a unity of purpose: a common vision. What we need is a champion to make it happen. We have not one but two special interest groups now: FIG and SIGForth. We also have a more academically oriented organization in the Institute for Forth Applications and Research. Among these three organizations, we should be able to come up with a structure and method to seriously market Forth for the next year or two. If we don't, we had better brush up on our C and FORTRAN to feed our families in the coming years.