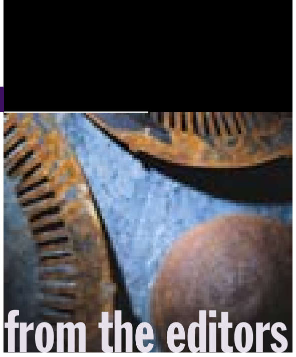

The Truth About Embedded Systems

George Neville-Neil, Queue Advisory Board Member

They are different FROM ANY OTHER SOFTWARE SYSTEM.

Embedded systems are different in several ways from other software environments. The hardware they run on is often resource-constrained in terms of both memory and processor cycles, but still these systems must respond in realtime. They control the brakes of cars, the flaps of airplanes, traffic signaling systems, medical equipment, and other life-critical devices. Programming as if someone's life depended on it is a new concept to many systems engineers.

In the past 10 years the embedded computing world has expanded from its traditional role in military, aerospace, and industrial automation applications to the networking equipment and consumer markets. This is the place where many software engineers now have their first contact with embedded systems. Whether it's building an application for a PDA, cellphone, router, or set-top box, working in this new environment requires different ways of thinking about design and implementation.

This issue of *ACM Queue* brings the challenges in working with embedded systems to a broader audience and attempts to lay out the problems for designers and implementers as they interact with this part of the computing field.

One hurdle in working with embedded systems is that the programming model is different from what most programmers are taught. Embedded systems often lack the safety of a process model and virtual memory. Programs can see into and modify the others' private data, global variables, and program instructions, which means that a misbehaving program can step all over correctly working ones and cause the entire system to crash. I explore these problems in "Programming Without a Net."

It is common to use embedded systems as a small part of a larger system. They may be in the line card of a router, a video card, or the control processor of a RAID. These types of systems divide the work among a set of cooperating processors or have a hybrid system where the central controlling system is a traditional operating system, and the specialized devices run an embedded operating system (EOS) or none at all. The challenges in designing and implementing these specialized distributed systems are addressed in "Division of Labor in Embedded Systems" by Ivan Godard.

Whereas in non-embedded environments the programmer may know little or nothing about the hardware on which their software runs, embedded software often exists solely to support a specialized piece of hardware. Functions that were initially implemented in software need to be quickly moved into hardware. Homayoun Shahri discusses these issues in "Blurring Lines Between Hardware and Software."

Delivering an embedded system involves more than just writing a single application. It is a process that requires integrating disparate bits of hardware and software—some of which you build and some of which you buy—and making them all work together as a coherent whole. In "Putting it all Together," Rolf Ernst shows us what it takes to finish that last 20 percent of a project that takes 80 percent of the time, focusing on getting the performance you expected when you designed the system.

It used to be that an embedded design went onto a large single-board computer or a set of them. With the advent of system-on-a-chip (SoC) technology the entire main board of a fairly powerful computer can exist on a single chip. This shrinkage has several side effects. Telle Whitney and I discuss these in "System on a Chip: Software, Hardware, Nightmare, or Bliss."

Jim Ready has been working in embedded systems for more than two decades. He built an entire company, Ready Systems, around hard realtime kernels and the tools to work with them. Now he is president and CEO of MontaVista, a company that is putting Linux into the embedded world. Randy Harr caught up with Ready to get his personal take on where the embedded world is going and, in particular, what part Linux will play in that world.

Embedded systems encompass almost every aspect of computer science and software engineering. We hope that this diverse set of articles gives you the kick-start you need to begin working with them. Q

GEORGE NEVILLE-NEIL, gnn@neville-neil.com, works on networking and embedded operating systems in various environments.