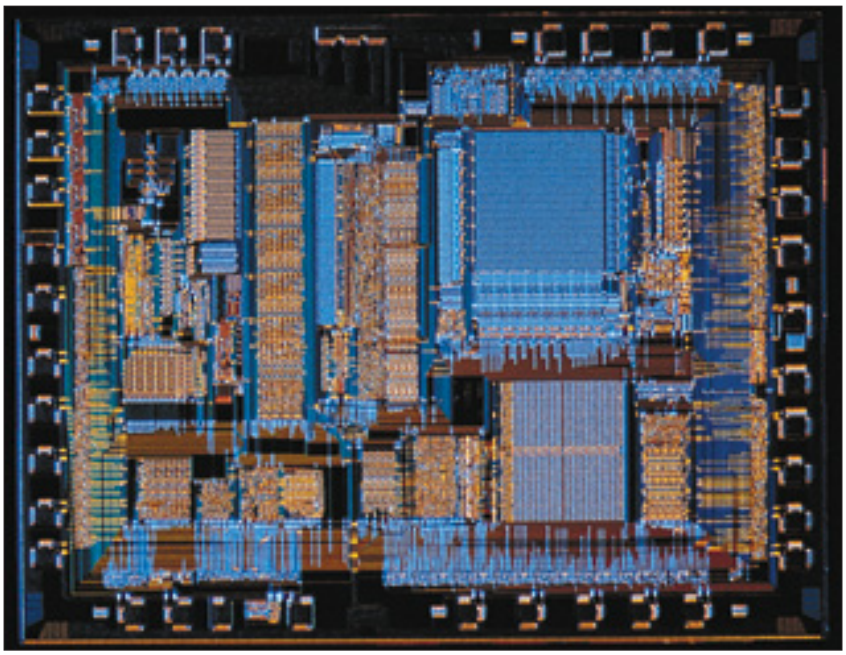


PHILIPS SEMICONDUCTORS

**[1] Microcontrollers are self-contained ICs designed to perform a specific function by themselves. This 8-bit CMOS microcontroller chip has 256B of RAM, 8KB of ROM, and another 8KB of electrically programmable ROM. It runs at 16 MHz.**

# Workhorses of

## Today's microcontrollers are performing better than ever through their use of high-level languages and multitasking techniques

A CLOSER LOOK AT THE MACHINES AND APPLIANCES that serve us well in our daily lives almost always reveals a microcontroller within. Controllers are embedded in cordless and portable telephones, point-of-sale retail electronic cash registers, scanners of all kinds, security systems, automobiles and gas pumps, automated tellers, computers, and compact disks and disk drives, not to mention phone-answering, fax, vending, and washing machines. In fact, electric lights are almost the only electrically powered devices that do not use microcontrollers, and even here things are changing with the welcome advent of power-saving and quick-starting intelligent ballast fluorescent lamps.

ATA R. KHAN
*Philips Semiconductors*

Still, although microcontrollers and microprocessors share many architectural features, they differ in important respects [Table 1]. A microcontroller is generally a one-chip integrated system meant to be embedded in a single application; so it is likely to include all the peripheral features—program and data memory, ports, and related subsystems—needed for the computer aspect of the application [Fig. 1]. By contrast, a microprocessor drives a general-purpose computer whose ultimate application is not known to the system designers.

Microcontrollers may not enjoy the publicity of their more glamorous cousins. Still, they are a key part of the computing landscape, as their sales and ubiquity prove [Fig. 2].

While certain sectors of the microcontroller market—for example, the large 4-bit segment—are essentially stagnant, the high end of the market is growing rapidly. Designs incorporating 16-bit microcontrollers continue to proliferate, and even 32-bit embedded control applications are beginning to appear in the industry [Fig. 3].


0018-9235/96/$5.00©1996 IEEE


VWGoA EX1019
U.S. Patent No. 9,955,551

The size of today's microcontroller market is fostering intense competition that helps spur technical innovation, benefiting users and designers of microcontroller systems by forcing down prices and creating a wide range of choices. In 1994, the top 10 microcontroller vendors (in descending order, based on revenues) were Motorola, NEC, Mitsubishi, Hitachi, Intel, Matsushita, Philips, SGS-Thomson, Toshiba, and Fujitsu. These familiar names have been joined by many third-party suppliers that provide a large assortment of development tools, real-time operating systems, and other products used to develop and implement the many applications microcontrollers have spawned.

## Trends in system development

Low-cost consumer items served by 4-bit controllers include microwave ovens, shavers, toasters, white goods, and tape players, to name just a few. The 8-bit market includes such embedded-control applications as television sets, disk drives, car radios, and auto body electronics, as well as personal computer peripheral systems (including printers, joysticks, modems, and mice). As compared with 8-bit controllers, 4-bit chips are ill-suited to running programs written in high-level languages, almost all of which are byte-oriented. So a 4-bit chip would have to execute multiple operations to do the simplest things. In addition, 4-bit controllers have very little compiler support. Programming must therefore be done in assembly languages, lengthening development cycles and creating a maze of problems for maintaining and upgrading code.

Meanwhile, 16-bit controllers are generally deployed in disk drives, automotive engine controls, and industrial control applications. And 32-bit devices are becoming more common in three principal areas: communications boards (such as network bridges and routers), laser and ink-jet printers, and x-terminals. Video games represent yet another opportunity for 32-bit embedded controllers, as well as for 64-bit ones.

With such widespread application, microcontrollers are subject to the same pressures that shape all segments of the electronics marketplace: how to achieve higher levels of performance and integration while not costing themselves out of a highly charged business environment where every penny counts.

System complexity and performance requirements have kept on growing. Meanwhile, product design cycles have shriveled to 6 to 12 months, compelling system suppliers to improve time-to-market and exploit ever-dwindling windows of opportunity. As a result abstraction and code reusability have become critical. But because these features are not adequately addressed by assembly language development methods, high-level languages, real-time kernels, and real-time operating systems are proliferating.

## High-level languages

Once dismissed as inefficient for 8-bit microcontroller designs, such high-level languages as C are now ubiquitous—and are even being used in systems with as little as 2KB of ROM on chip. In fact, for 16- and 32-bit applications, systems are now being developed exclusively with high-level languages. When optimized for microcontrollers, they yield efficient compilation programs that speed development and time-to-market. For maximum code density, the best course is to write assembly code by hand—assuming the designer has the advanced and increasingly rare expertise required to do so. But any loss of code density that occurs when code is written in a high-level language is more than offset by improvements in the product development time.

## Real-time kernels…

As programs grow larger in the real-time environments of many embedded control applications, the microcontroller has to coordinate demands from various parts of the system and schedule and service them. These duties call for multitasking capabilities and hence for real-time kernels (RTKs).

The kernels lay the groundwork for the required efficiency with their scheduling and orderly task execution facilities—tasks must be prioritized and deterministic, with known latency and so forth. In more detail, RTKs include mechanisms for starting and completing tasks, for switching between them in response to various priorities, and for facilitating communication among tasks for such purposes as synchronization, resource sharing, and sequencing.

## …and operating systems

Microcontroller operating systems that are extensions of RTKs, providing real-time control constructs, are called real-time operating systems. They combine the task-control, real-

# the electronic era

**1. A microprocessor and microcontroller compared**

| Feature | Intel 486 | IDT R36100 |
|---|---|---|
| Central processing unit | 32 bits | |
| Cache memory | 8KB unified | 4KB inst., 1KB data |
| Prefetch queue | ✔ | — |
| Floating-point unit | ✔ | — |
| Memory management unit | ✔ | — |
| Communication controller | — | ✔ |
| Direct memory access controller | — | ✔ |
| Timers | — | ✔ |
| Memory controller | — | ✔ |
| P1284 parallel port | — | ✔ |
| I/O ports | — | ✔ |

time–driven facilities of an RTK with the standard features and services of an ordinary operating system, including the integration of languages, debugging environments, utilities, and file management.

While quite common in 16- and 32-bit microcontrollers, which are dominated by high-level languages, real-time operating systems and kernels have also been created for 8-bit devices. These real-time operating systems supply a range of program control, task-switching, and resource-allocation functions for complex applications where it is not feasible to write code by hand. Designers have found it much simpler and more efficient to use the optimized real-time kernels and operating systems available from today's wide range of third-party vendors.

Real-time kernels, executives, and operating systems must provide quite a few services. One is multitasking, for more than one task must be executed on one and the same system without interfering with one another. They may proceed to completion or be interrupted while executing, in which case their state must be saved so it can be resumed precisely. A task that fails to execute properly—whether because of bad programming, unexpected inputs, system glitches, or other factors—should not cause other tasks or the system itself to crash.

Second, the executive must be able to preempt tasks that are running so that more urgent ones may execute. If required, it must be possible to execute tasks in a specified, sequential manner.

Third, it may be necessary to support real-time clock-based operation, because some events are absolutely or relatively time-dependent: they must occur within, before, or after some specified time or interval.

Then, too, communications and resources must be synchronized. Tasks must communicate with one another in an orderly and predictable fashion within the context of a protected environment. In addition, they must pass resources back and forth, without having those resources overwritten by intervening tasks. Resource sharing must occur in such a way that tasks receive the proper attention without monopolizing system resources. Such classic large-machine operating system techniques as semaphores and queues fulfill these functions.

Finally, a kernel or executive has to provide for efficient switching among tasks. Efficiency can be defined as appropriate use of time or resources, or both, achieved through tradeoffs permitting users to select the degree of context saved. Time is exchanged for the amount of machine and task state that must be saved and later restored.

A real-time kernel should use system and chip resources economically. Embedded control is highly cost-sensitive, and memory size ought to increase minimally if at all. A real-time executive that makes large demands on memory may in fact reduce the performance of the system and add to its cost.

### Trends in development tools…

The development tools and environments that come along with the silicon have become integral to today's microcontroller architectures. As clock speeds increase and real-time development becomes a must, it has gotten more and more difficult to emulate and debug the target system. This simply cannot be relegated to an afterthought.

Indeed, the emulation and debugging of circuits must be built into the microcontroller design. Some aspects of these features (such as breakpoint/trace instructions and special cache instructions) may be visible at the architectural or instruction-set level, while other aspects are accessed through special emulation/debug modes. With various degrees of effectiveness, every advanced microcontroller today provides on-chip support for emulation and debugging.

As the use of assembly language has tapered off, tool suites have pushed to the fore to accommodate design requirements. By any measure, good tools save valuable time. The more popular the architecture, the broader the array of tools written for use with it.

Usually, these tools include an assembler, a simulator, a debugger, a compiler with a good library of routines, a linker and librarian, and a library of common embedded-control routines. Although some of the tools come from the microcontroller vendors themselves, ample third-party support for the leading architectures gives customers an array of choices for implementing system designs. An entire industry of third-party tool vendors is flourishing around the larger and more widely used microcontroller architectures.

### …and on-chip memory

More often than not, program memory is included on today's embedded controller chips. These single-chip solu-

## The onward march of the microcontroller

The first true microcontrollers grew out of the Intel 4004, a 4-bit microprocessor architecture developed in the early 1970s. They ruled the industry until the late '80s, when 8-bit implementations dethroned them. Today, the microcontroller realm is still dominated by the 8-bit segment, which represents three-fifths of the total worldwide market.

The most popular 8-bit architecture is the Motorola 68XX. Next comes the 80C51, originally developed by Intel but now offered by several vendors. In 1994, the top five suppliers of 8-bit microcontrollers were Motorola, NEC, Mitsubishi, Philips, and Hitachi.

Designs employing 16-bit microcontrollers grew more common in the mid-to-late '80s, after the introduction of the architecture that still leads the 16-bit segment: the Intel 80C196. Motorola (the 68H16), Philips (the XA), NEC (the K series) and other companies recently introduced new 16-bit architectures that are competing for market share.

Intel's i960 leads the way in the 32-bit market, which is as yet quite small but has been buoyed by a number of entrants. These include Hitachi's SH, Motorola's Coldfire, and a slew of upcoming embedded reduced–instruction-set computer (RISC) offerings from such vendors as IDT, Toshiba, LSI Logic, Philips, and NEC, based on architectures from MIPS Technologies.

As in all other segments of the microcontroller marketplace, these new offerings give customers more choices, driving down prices and opening up new opportunities. Because the algorithms of digital signal processors (DSPs) are used in many embedded-control applications, DSPs are often regarded as a class of microcontroller, further expanding the range of today's applications. —*A.K.*

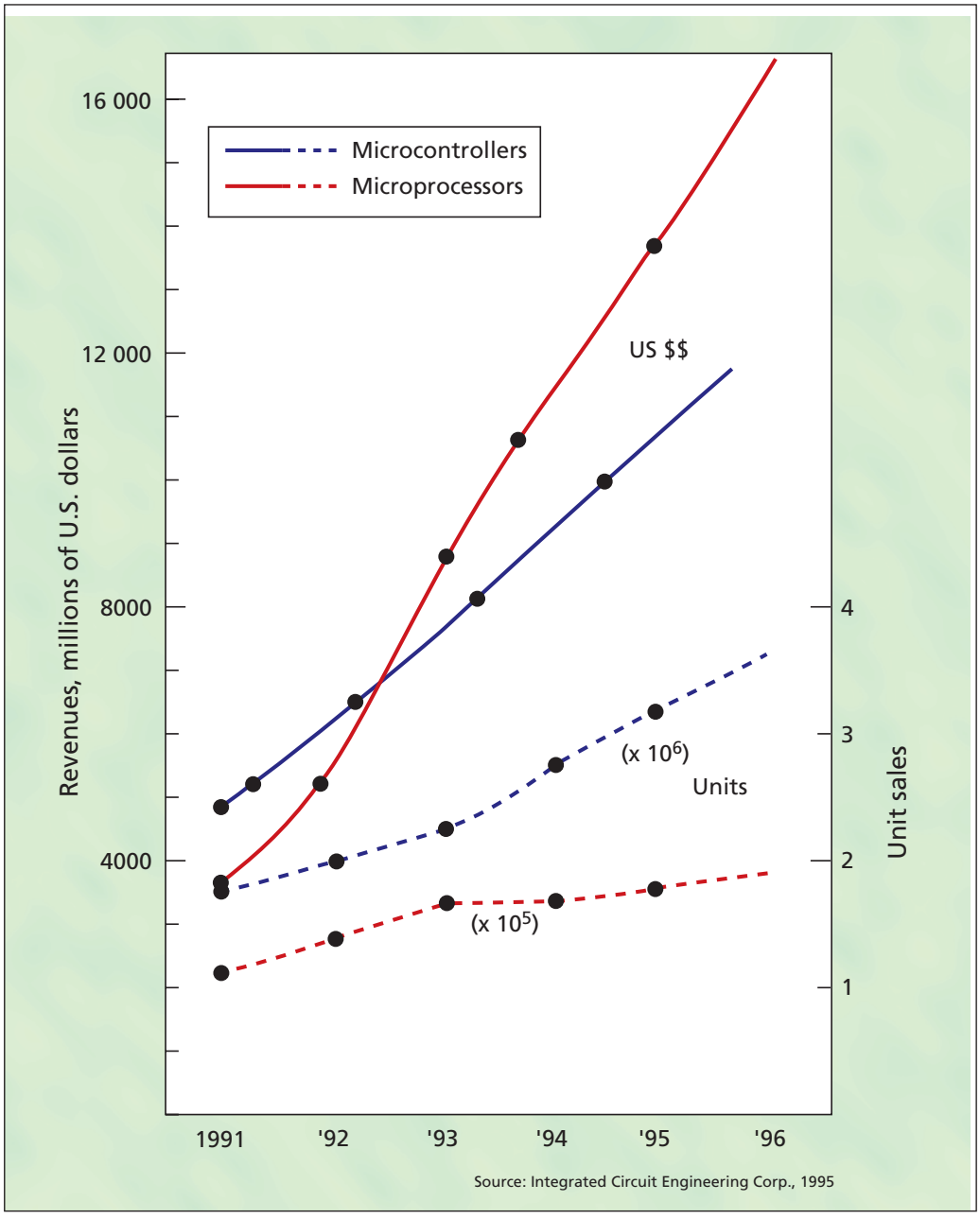
tions supply the need for small form factors and low cost, particularly in the 8-bit arena. If the task at hand warrants it, 32- and 16-bit controllers, which can address many megabytes of memory, may also use external memory.

The least expensive type of memory available for this purpose is ROM. But, since ROM programming takes place during IC manufacture using a dedicated mask, any error means that a new mask must be generated. When that happens, design cycles drag on and costs start to balloon—not exactly to be desired in very cost- and time-sensitive business environments.

To circumvent these difficulties, semiconductor suppliers often integrate more expensive one-time–programmable (OTP) ROM variants on chip: electrically programmable ROM and electrically erasable and programmable ROM. The first, EPROM, can be programmed electrically and erased by ultraviolet (UV) light. Putting these devices in plastic packages opaque to UV light produces OTP controllers that cannot be reprogrammed. EPROMs are cheap because they use very dense memory cells based on a single transistor. EEPROMs can be programmed and erased electrically but cost more because each cell requires the functional equivalent of an extra transistor.

Often used in prototype development, reprogrammable memories make it easier for designers to modify or upgrade code, providing a higher degree of flexibility. Their biggest advantage is a shorter time to market, obtained by eliminating the ROM masking cycle. Coupled with today's sharply reduced product life cycles, this has popularized the use of OTP microcontrollers in embedded applications.

Ideally, though, memory should be both reprogrammable and inexpensive. While nothing today fully satisfies these dual requirements, most industry experts believe that flash memory may soon fit the bill. Flash mass storage is highly flexible and reprogrammable, and steady declines in




**[2] On average, the price of a microprocessor has more than tripled over the past five years, whereas a microcontroller sells for only 16 percent more. As a result, revenues from processor ICs have grown twice as fast as those from controller ICs even though unit sales of the controllers outpaced those of processors by the same factor. The data shown here includes 4-bit microcontrollers and all 8-, 16-, 32-, and 64-bit processors and controllers. 1996 figures are projected.**

associated costs indicate that the technology will probably be cheaper to implement than EPROMs within the next three to five years. Today, many microcontroller vendors—including Philips, Motorola, Hitachi, Atmel, and Mitsubishi—are hard at work developing flash-based products.

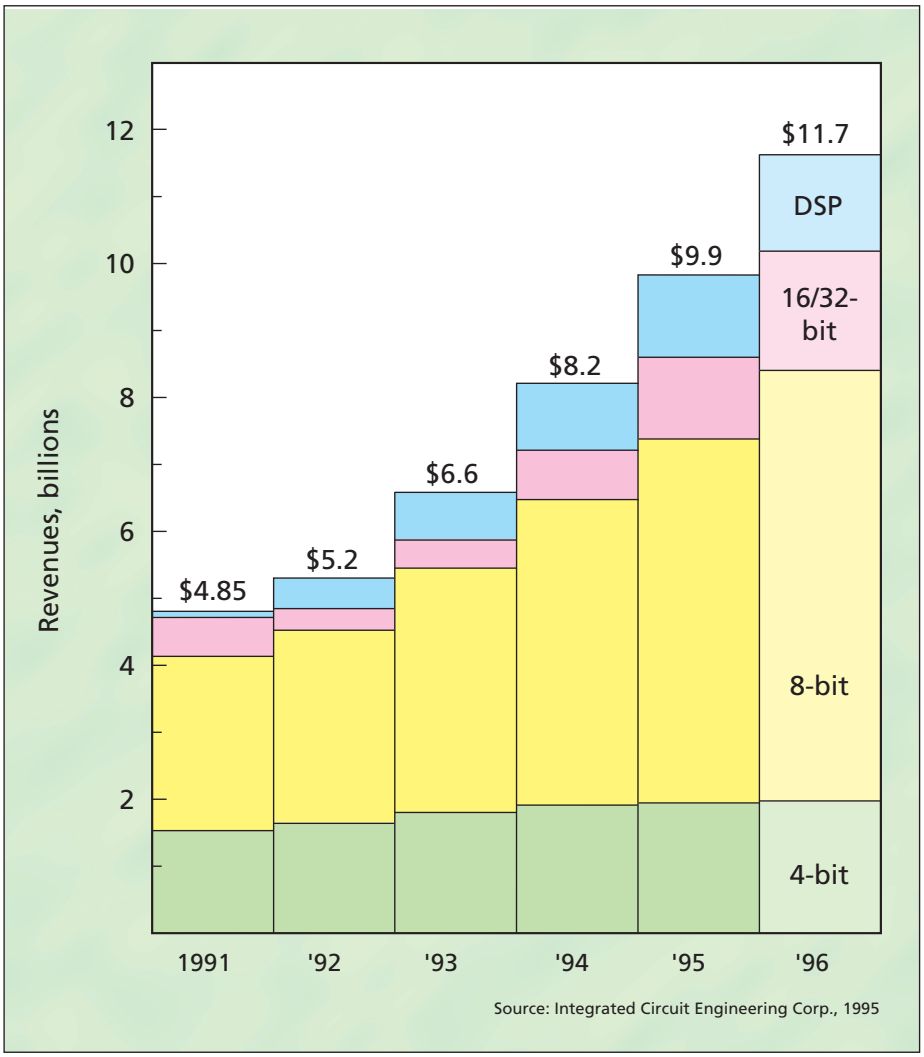
## Coming next

Looking ahead, a number of technologies are bound to influence the evolution of microcontrollers for embedded-control applications. As integration marches on, submicrometer process technologies will affect system form factors and designs. Although today's microprocessor processes are typically in the 0.35—0.5-μm range, the cost sensitivities associated with microcontrollers currently dictate processes on the order of 0.8 μm. Within a few years, continuing advances promise to drive microcontroller process technologies down to 0.5 μm and below. At process dimensions of this small size, providing and maintaining compatibility with nonvolatile technologies will present manufacturers with a stiff challenge.

Yet other pertinent technologies include the use of embedded dynamic RAM to provide for greater levels of system integration and more flexible options in system repartitioning; hardware description language (HDL) and synthesis techniques to save time and provide automated system simulations; and object-oriented program development to write code automatically in a way that is transparent to system product designers. Between them, all of these developments will make possible very large and complex single-chip control systems, complete with memory and peripheral subsystems, of a type that can be developed and programmed very quickly.

Other hopeful trends are multichip packaging, micromachining techniques that integrate sensors on chip, and the mapping of behavioral system descriptions into the microcontroller's program. Ferroelectric memories in particular bear watching. because they are nonvolatile and because, unlike dynamic RAMs, they do not need to be refreshed, while unlike electrically erasable and programmable ROMs, they do not need special program-and-erase cycles. Unfortunately, the process of reading the cell is volatile, so information must be written back into memory. In addition, dedicated manufacturing lines are required to implement the technology. When these cells are able to read and write for around $10^{12}$ cycles, ferroelectric memories start to become an interesting alternative for embedded applications. Current ferroelectric technology is at about $10^{8}$ cycles.

These trends point to systems—all on a single chip—that will be able to sense such physical inputs as temperature, acceleration, and light sensitivity; convert them into digital information; perform signal processing; output process-control signals; communicate status; handle remote diagnostics; reprogram themselves; and adaptively change their functionality. ◆




**[3] The 8-bit controller ICs bring in the lion's share of total income from microcontrollers. But sales of 16- and 32-bit microcontroller are growing rapidly. This year, digital signal processors are expected to bring in almost 14 percent of the total microcontroller income. The 1996 figures are projected.**

## To probe further

The internals of a small real-time preemptive multitasking kernel are described in *μC/OS, the Real-Time Kernel,* by Jean J. Labrosse (R&D Publications, 1992, distributed by Prentice Hall).

Journals and magazines that regularly cover microcontrollers include *Electronic Engineering Times* (CMP Publications, Manhasset, N.Y.), *Microprocessor Report* (MicroDesign Resources, Mountain View, Calif.), and *EDN* (Cahners Publishing Co., Highlands Ranch, Colo.).

## About the author

Ata R. Khan is manager of the Microcontroller Development Center at Philips Semiconductors, Sunnyvale, Calif. He is responsible for the strategic planning and architecture for microcontrollers.

*Spectrum* editor: Linda Geppert