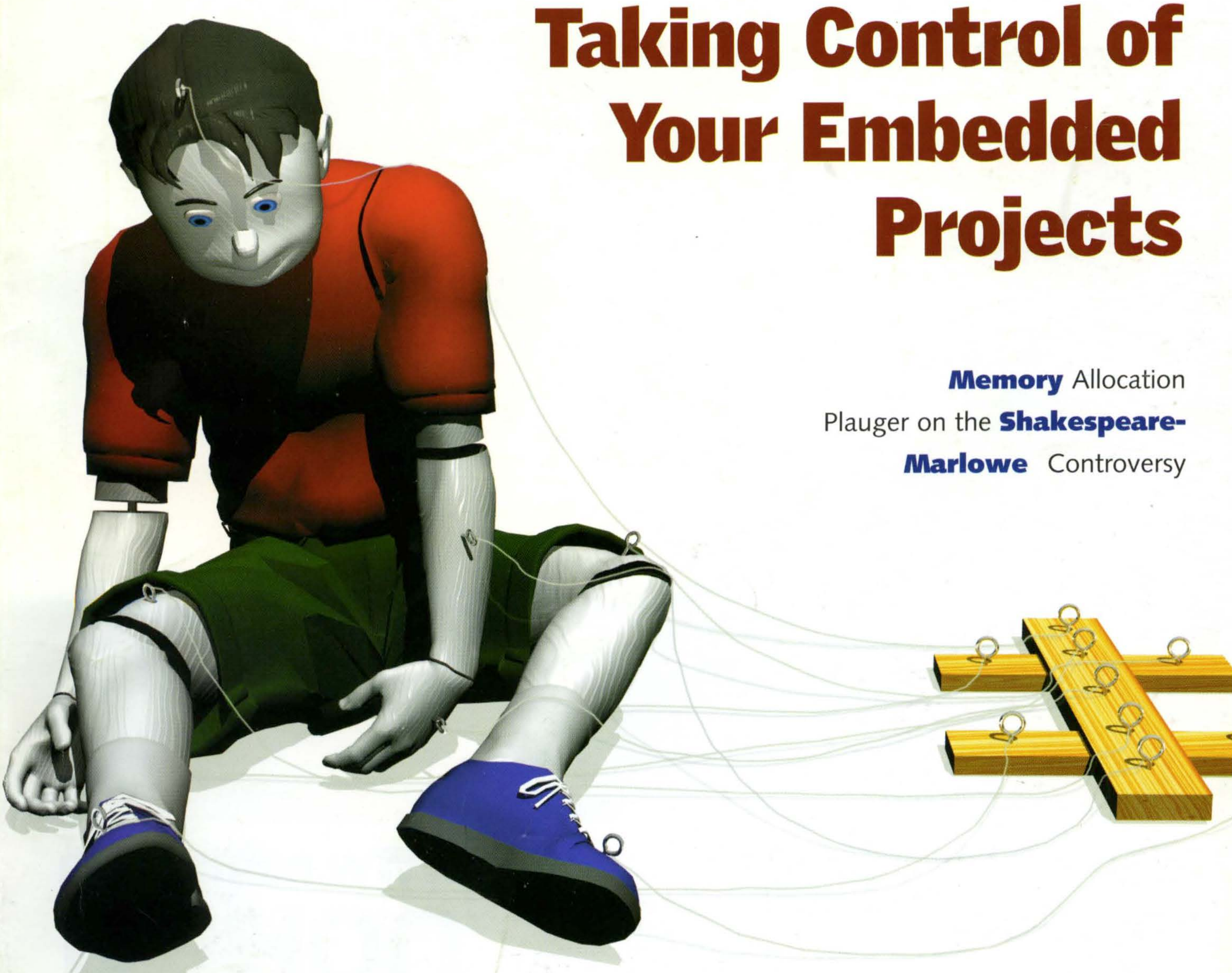
april 2000 www.embedded.com volume 13, number 4

EmbeddedSystems® PROGRAMMING

Taking Control of Your Embedded Projects



Memory Allocation

Plauger on the **Shakespeare-Marlowe** Controversy

Internet Appliance Design:
TCP/IP Configuration
Transmeta's Crusoe Processor

Cut Development Time

Using In-Circuit Emulators and BDM's

[startup|Main# 3] Cfg_PortAddress "0X200"

```
Sub Main
  ' Some fundemental values
  Cfg_PortAddress "0X200"
  Cfg_Processor "POD-196KC"
  ' Turn off logging
  'File_LogToScreen "No"
  'File_LogToFile "No"
  ' trace config stuff
  Trace_TracePort "208"
  Trace_TraceBoard "Yes"
```

Paused.

Macros

- Macro debug capability
- Project support
- Command line
- Shadow RAM support
- Single step in your macro
- GUI capable macros

Key Features

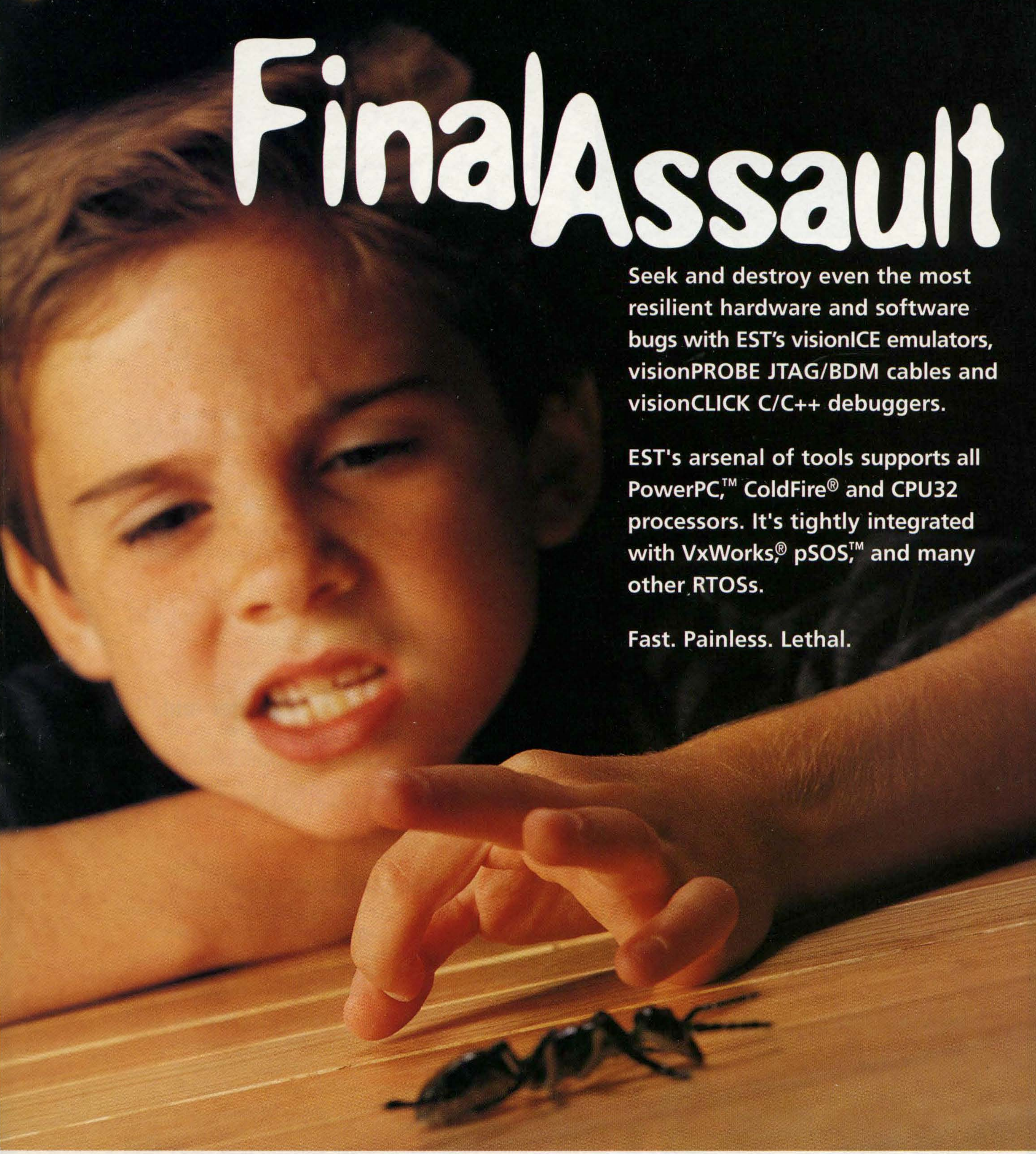
- Real-time emulation at maximum chip speeds
- High level support for popular C compilers
- Advanced tracing capabilities
- Configurable user interface with remote hookup capability
- High speed connection to PC through parallel port (LPTx) or plug-in ISA card

Nohau Supports These Microcontroller Families
8051 80C196 683xx 68HC11 P51XA ST10
MCS296 MCS251 68HC16 C166 68HC12 M16C

www.nohau.com

NOHAU

51 East Campbell Avenue, Campbell, CA 95008
Phone: 1-888-88NOHAU (1-888-886-6428)
Fax: 408-378-7869 E-mail: sales@nohau.com



Seek and destroy even the most resilient hardware and software bugs with EST's visionICE emulators, visionPROBE JTAG/BDM cables and visionCLICK C/C++ debuggers.

EST's arsenal of tools supports all PowerPC,™ ColdFire® and CPU32 processors. It's tightly integrated with VxWorks,® pSOS,™ and many other RTOSs.

Fast. Painless. Lethal.

visionICE

visionCLICK

visionPROBE

PowerPC™ 68K ColdFire

EST

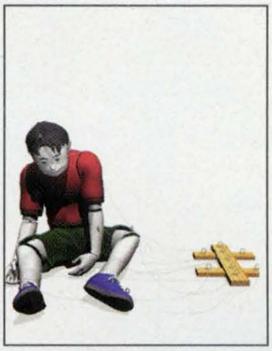
Embedded Support Tools Corporation

Call 800-957-5588 or Download a Free Demo: www.estc.com

Embedded Systems
PROGRAMMING

contents

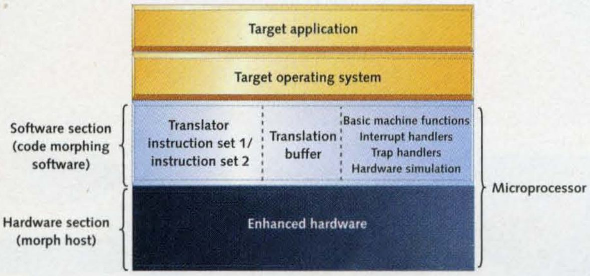
APRIL 2000



COVER

Managing people, processes, and technology may require pulling a few strings.

Cover illustration by Rupert Adley.



128

Microchip's PIC16F87X family of flash chips.

Featured Section

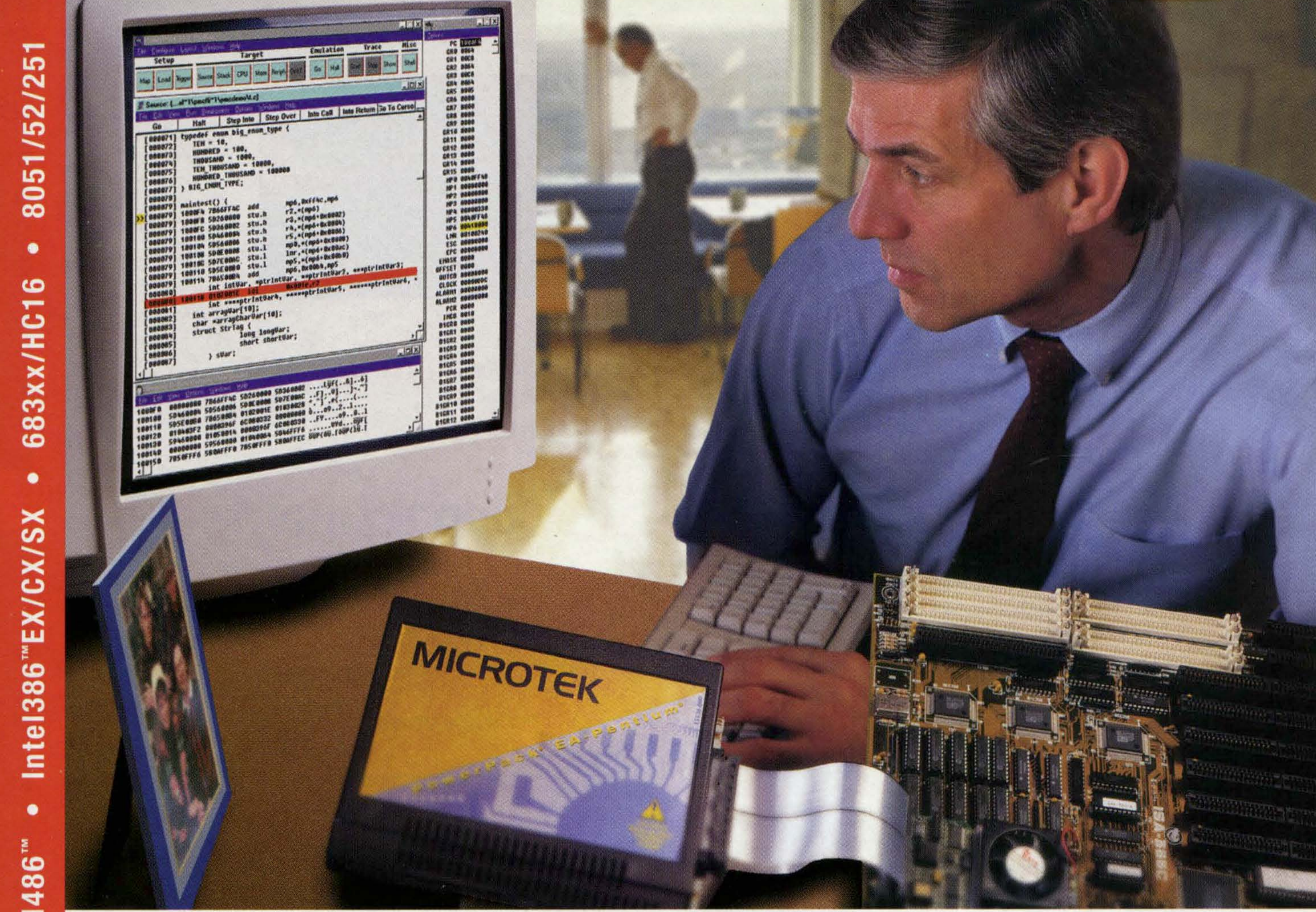
internet appliance design

New Technology

departments

columns

Pentium® III • Celeron® • Pentium®II • Pentium® • Intel486™ • Intel386™EX/CX/SX • 683xx/HC16 • 8051/52/251



"If only I had spent less time on debug..."

I remember when I budgeted for my project. I refused to spend money on debug tools. I didn't think I would need them. The money I saved would make me a hero.

Now, my project is one month, two months ... six months overdue. Nobody thinks I'm a hero. What would I give to get some of that time back?

Microtek debug tools can make time. They speed up development and testing, saving you precious time at the end of the project, where you need it most!

Microtek emulators offer the debug features you need to track down and correct software, hardware, and system integration issues. They can find the errors software debuggers cannot see.

First, emulators can debug before the operating system is functioning. If an unexpected issue is affecting boot up, an emulator can find it. They are also operational after a hard crash. This is significant, because software debuggers lose debug information. Microtek emulators keep track of the last 128 KB of bus cycles, allowing you to sift through and find the problem.

Finally, if there is a subtle issue during the integration of hardware, Microtek EA emulators are capable of providing a clock-cycle-by-clock-cycle trace that allows you to view each signal and determine whether it was in the correct state. Microtek's tools are clearly superior to software debug solutions.

URGENT!

If you are developing a Pentium class target using a reference design or off-the-shelf board, please take a moment to speak with our staff. They will insure your design is on track, and tool friendly if debug is needed in the future.

There has never been a project that couldn't use more time when it's critical ... like in final testing when everything has come together ... or in final debug, when everything is going down in flames.

Wouldn't it be great to deliver your next project on time? Next time, be sure to put a Microtek emulator in your project plan from the start.

It will help take the knot out of your stomach. And it will improve your company's bottom line!

MICROTEK
IN-CIRCUIT EMULATORS

1 (800) 886-7333
Phone (503) 533-4463
Fax (503) 533-0956
Email – info@microtekintl.com

Additional interfaces: CAD/UL® and Windriver Tornado II®

www.microtekintl.com

Financing Available

EmbeddedSystems
PROGRAMMING

EDITORIAL DIRECTOR
Lindsey Vereen, lvereen@cmp.com

MANAGING EDITOR
Felisa Yang, fyang@cmp.com

TECHNICAL EDITOR
Michael Barr, mbarr@cmp.com

SENIOR SPECIAL PROJECTS EDITOR
Tarita Whittingham, twhittingham@cmp.com

CONSULTING TECHNICAL EDITORS
Jack G. Ganssle
Jerome L. Krasner, PhD

CONTRIBUTING EDITORS
Jack W. Crenshaw
Larry Mittag
Don Morgan
P.J. Plauger
Dan Saks

VICE PRESIDENT/ELECTRONICS
Donna J. Esposito

EMBEDDED/DSP GROUP DIRECTOR
Mike Flynn, (415) 278-5251

PUBLISHER
Eric Berg, (415) 278-5220

CALIFORNIA SALES MANAGER
Andres Diaz, (415) 278-5274

CALIFORNIA SALES ASSISTANT
Molly Bruns, (415) 278-5298

WESTERN ACCOUNT EXECUTIVE
Sam Louis, (415) 278-5223

EASTERN REGIONAL SALES MANAGER
Damon Graff, (781) 235-8258

EASTERN SALES
Jared Grimm, (781) 235-8258

PRODUCTION COORDINATOR
James Whitehead

CIRCULATION MANAGER
Jennifer Schuler

CIRCULATION DIRECTOR
John W. Rockwell

SUBSCRIPTION CUSTOMER SERVICE
Toll free: (877) 676-9745
(847) 291-5215
esp@omeda.com
Back issues may be purchased on a prepaid basis through: Miller Freeman, 1601 West 23rd St., Suite 200, Lawrence, KS 66046; (800) 444-4881; (785) 841-1631

REPRINTS
Sherry Bloom, (415) 808-3980

ONLINE PRODUCTION COORDINATOR
Billy Biondi, wbiondi@cmp.com

PRESIDENT/CEO, CMP MEDIA INC.
Gary Marshall

EXECUTIVE VICE PRESIDENTS
Regina Starr Ridley John Russell
Steve Weitzner Tony Uphoff

SENIOR VICE PRESIDENT/ GLOBAL SALES & MARKETING
Bill Howard

SENIOR VICE PRESIDENT/ BUSINESS DEVELOPMENT
Pam Watkins

PRESIDENT/ELECTRONICS
Steve Weitzner

Consumer Audit BPA

CMP

Visit our Web site at
www.embedded.com

#include

Lindsey Vereen

Vehicular Rant

Maybe I'm just getting old, but I don't get the concept of the sports utility vehicle. How can a cross between a pickup truck and a station wagon be the coolest thing on wheels for the smart set? Among their myriad features, SUVs are hard to park, expensive to insure, and gas guzzlers.

In San Francisco, a city of narrow streets and few parking places, SUVs have become as common as pigeons, and nearly as much of a nuisance. Because of their size, they take up a disproportionate amount of precious parking space. Each morning you can find dozens of them perched on sidewalks because there were no spaces large enough the night before. Now you'd think that the rationale for people to buy SUVs would be to go skiing in the Sierras. But no, many owners don't want to besmirch their vehicles with mud and snow.

Now I'm not trying to annoy all of you SUV drivers out there. I certainly recognize the value of four-wheel drive for people living in snowy climes. And I like the way you can sit high enough to see over traffic, such as other SUVs. My beefs are twofold. First is the disparity between the large SUVs and small compacts. I'd like to feel that if I were driving a Geo Metro, I'd have a shot at surviving a collision with another vehicle.

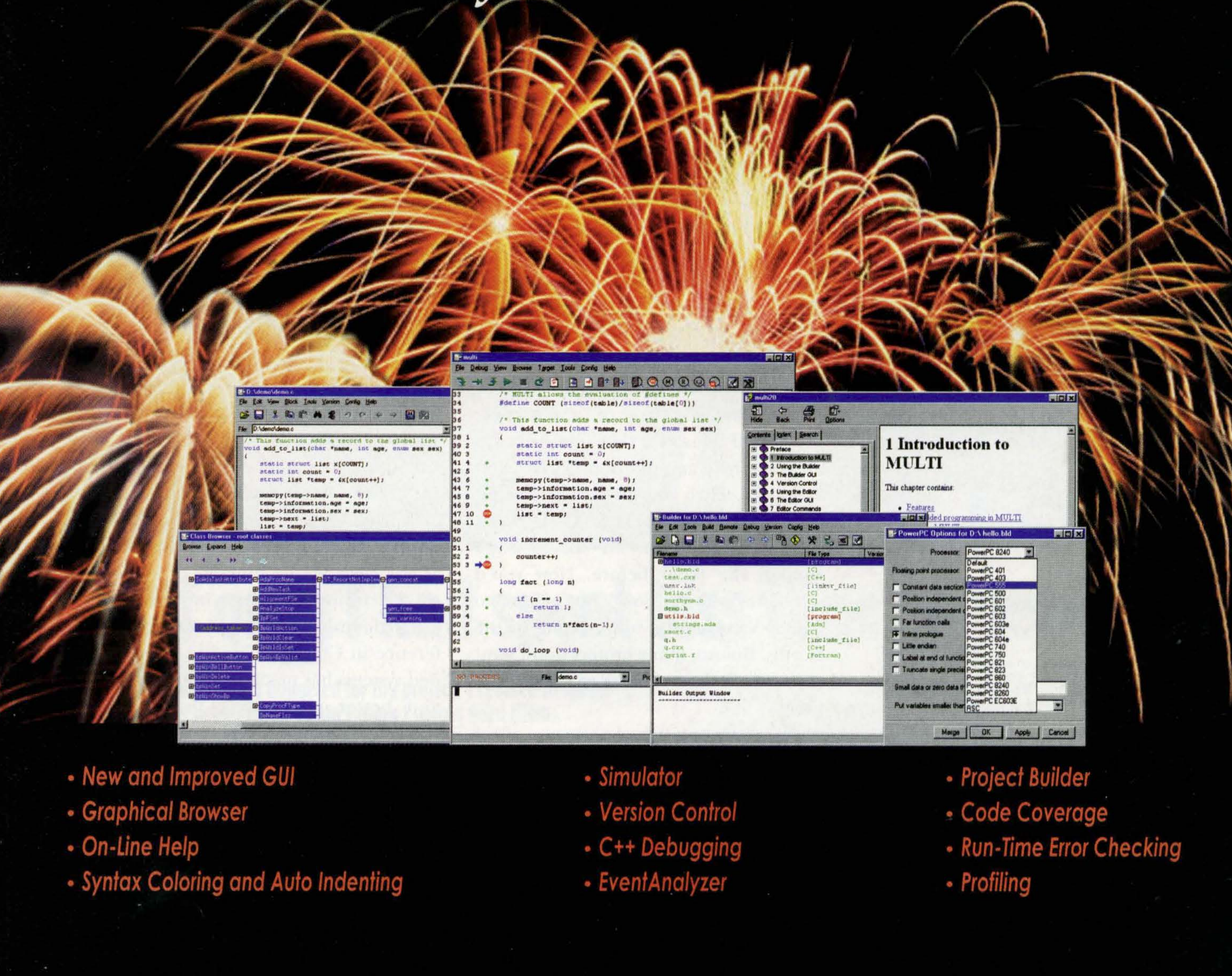
My other beef is about the fuel economy and emissions that characterize these vehicles. That problem appears to be going away—eventually—thanks to recently passed legislation. From then on it's up to the auto industry to squeeze out all the gas mileage and extra weight.

If we were to eliminate one or the other, small fuel-efficient vehicles don't seem the ones to put on the chopping block. I guess mine is a minority viewpoint. SUVs aren't disappearing from the streets and highways of America. And to make matters worse, I'm even seeing more Hummers on the streets of San Francisco. Figure that one out. Fortunately we have the technology to help make other (smaller) vehicles safer to the occupants when SUVs try to occupy the same space at the same time.

We're all aware of the litany of virtues that embedded intelligence can add to products: features, reliability, reduced power consumption, and added safety. Those virtues are being exploited more and more by the auto industry. High-end automobiles now boast more than 60 processors. The auto industry is beginning to depend on embedded systems, as was clear by the number of automotive electronics engineers in attendance at last month's embedded systems conference in Chicago. The use of embedded systems has made automobiles safer and better performing than before CAFE (corporate average fuel economy) standards were put into place in the 1970s.

Since SUVs are not going away, and their drawbacks (other than size) can only be mitigated through technology, we can take comfort in one additional virtue that embedded systems offer. In a small way they can help save us from ourselves.

lvereen@cmp.com



```
/* This function adds a record to the global list */
void add_to_list(char *name, int age, enum sex sex)
{
    static struct list x[COUNT];
    static int count = 0;
    struct list *temp = &x[count++];

    memcpy(temp->name, name, 8);
    temp->information.age = age;
    temp->information.sex = sex;
    temp->next = list;
    list = temp;
```

```
        /* MULTI allows the evaluation of #defines */
        #define COUNT (sizeof(table)/sizeof(table[0]))

        /* This function adds a record to the global list */
        void add_to_list(char *name, int age, enum sex sex)
        {
            static struct list x[COUNT];
            static int count = 0;
            struct list *temp = &x[count++];

            memcpy(temp->name, name, 8);
            temp->information.age = age;
            temp->information.sex = sex;
            temp->next = list;
            list = temp;
        }

        void increment_counter (void)
        {
            counter++;
        }

        long fact (long n)
        {
            if (n == 1)
                return 1;
            else
                return n*fact(n-1);
        }

        void do_loop (void)
```

```
hello.bld        [program]
  ..\demo.c      [C]
  test.cxx       [C++]
  user.lnk       [linker_file]
  hello.c        [C]
  sortbynm.c     [C]
  demo.h         [include_file]
utils.bld        [program]
  strings.ada    [Ada]
  xsort.c        [C]
  q.h            [include_file]
  q.cxx          [C++]
  qprint.f       [fortran]
```

1 Introduction to MULTI

This chapter contains:

- Features
- ...ded programming in MULTI

Processor: PowerPC 8240

Floating point processor:

□ Constant data section
□ Position independent
□ Position independent
□ Far function calls
☑ Inline prologue
□ Little endian
□ Label at end of functio
□ Truncate single precis

Small data or zero data th

Put variables smaller than

Default, PowerPC 401, PowerPC 403, PowerPC 555, PowerPC 500, PowerPC 601, PowerPC 602, PowerPC 603, PowerPC 603e, PowerPC 604, PowerPC 604e, PowerPC 740, PowerPC 750, PowerPC 821, PowerPC 823, PowerPC 860, PowerPC 8240, PowerPC 8260, PowerPC EC603E, RSC

QED launches IPO

Quantum Effect Devices Inc. announced the initial public offering on February 1 of its common stock trading under the symbol "QEDI" on NASDAQ. In its first day of trading, the stock opened at 39 1/4 and closed at 56 1/2. At press time, the stock was at 95 3/16.

The initial public offering price for the 3,720,000 shares of common stock was $16 per share. The company offered 3 million shares and a shareholder offered the remaining 720,000. The offering was managed by Morgan Stanley Dean Witter, Lehman Brothers, and Robertson Stephens.

Quantum Effect Devices, headquartered in Cambridge, MA, was founded in 1991 and develops embedded microprocessors for use in information processing in networking/communications infrastructure equipment.

Briefly Noteds...

Wind River Systems will resell Rapid Logic's MIBway for Wind Web Server product through its Wind River Direct program. MIBway provides support for Wind River's Tornado platform and VxWorks. ✳ **FSMLabs** and **Synergy Microsystems** have released the beta version of RTLinux for the PowerPC. ✳ The MULTI 2000 Software Development Environment from **Green Hills Software** is now available for the ARM processor family. ✳ **U S Software**'s TronTask! real-time operating system now supports Intel's StrongARM processor. ✳ **emWare** has partnered with **Invensys Network Systems**, **Motorola Semiconductor Products Sector**, **GE Appliances**, and **Sunbeam** to produce end-to-end solutions for networked devices. ✳ **PointBase** will bundle its 100% Pure Java database with **Symantec**'s VisualCafe Development Environment. ✳ The Embed The Internet (ETI) Alliance has changed its name to the **Extend the Internet (ETI) Alliance**. **STMicroelectronics** recently joined the group. ✳ **Enea Data AB** has acquired **TekSci**, a software consulting firm. ✳ **NEC** has entered into a licensing agreement with **Tensilica** for its Xtensa processor technology. ✳ The SC Series controller chip from **Scenix Semiconductor** is used in the Sennheiser Digital 1000 Series Wireless Microphone System's microphone and receiver unit. ✳ **VMIC**'s generic device driver supports the QNX operating system. ✳ **ESS Technology** will use Green Hills Software's optimizing C compiler in developing embedded software for its Internet television set-top box platforms. ✳ **Radstone Technology** and **Lynx Real-Time Systems** have partnered to provide Lynx's BlueCat Linux OS on Radstone's PPCx family of PowerPC-based SBCs. ✳ **Netstream**'s OpenSSL will be ported to **Enea OSE Systems**' OSE RTOS. ✳ **LinuxDevices.com** has launched the Embedded Linux Polls center. The online polls track user and developer trends and preferences.

VME Continues to Dominate Embedded Merchant Computer Board Shipments

The world market for merchant computer boards totaled $1.98 billion in 1998 and is expected to total $2.19 billion for 1999, according to a report by the Electronics Market Forecasters Group (EMF).

In 1998, VME shipments accounted for 53.2% of worldwide shipments, PCI accounted for 18.5% of shipments, and CompactPCI accounted for 2.2% of total worldwide board shipments. Combined shipments of ISA, EISA, Multibus, STD, PC/104, and S-bus constituted 16.0% of worldwide shipments.

In 1999, VME shipments declined to 61.9% of worldwide shipments, notwithstanding an 8.5% increase in total revenues. PCI shipments accounted for 20.3% of worldwide shipments, whereas CompactPCI accounted for 3.6% of worldwide shipments.

Combined ISA, EISA, Multibus, PC/104, and STD shipments accounted for 14.2% of worldwise shipments in 1999 as ISA and EISA legacy sales continued their decline.

NAMES IN THE NEWS

American Microsystems Inc. (AMI) has appointed **H. GENE PATTERSON** president and CEO. ● **S. CAINE O'BRIEN** has been promoted to CEO of ARTiSAN Software tools. He was previously vice president of marketing and executive vice president of its North American operations. ● **LARY EVANS** has been named to Applied Microsystems Corp.'s (AMC) board of directors. Evans retired from Dell Computer Corp. in 1998, where he served as vice president and general manager of Dell's server business. ● Red Hat named **MICHAEL TIEMANN** chief technology officer. ● Aisys Inc. has appointed **ORI INTRATOR** vice president of sales and marketing for Europe and **ELAN ROTH** vice president of business development. ● **ROBERT MOORE** has been named vice president of marketing for CoWare. ● NEC Electronics Inc. has announced several management appointments: **GEORGE WANG** has assumed responsibilities to include general manager of the VR Series Microprocessor Strategic Business Unit. Wang will also continue his role as general manager for the consumer and automotive divisions; **KARL AUKER** was promoted to assistant general manager for the VR Series Microprocessor; **CHUNG HO** was appointed assistant general manager of the Communications Division; and **STEPHEN TANG-KONG** was promoted to associate general manager of Mixed Signal Technology Group.

BASEBAND PROCESSORS
GaAs•RF/IF•SOFTWARE

Jack W. Crenshaw

A Simulator's Paradise

After I finished and submitted last month's column ("And Another Thing...!" March 2000, p. 15), in which I released some of my pent-up frustrations about software that doesn't work, I realized that I hadn't kept my part of our bargain. When I originally announced my intention to tell you about software products to watch out for, I promised that the knife would cut both ways; I'd also tell you about the "good guys," companies that make good stuff, and are a pleasure to deal with. So before we get into the meat of this month's column, I want to try to even up the balance sheet a bit.

I've already told you about Matlab (see *www.mathworks.com*), which I've turned to as my baseline problem solver. No one who has attended college in a technical discipline, in the last decade, needs to be told about Matlab. It's become as ubiquitous on the college campus as Unix once was.

The biggest surprise about Matlab for me was that wonderful accessory called Simulink. In case you missed the description, Simulink is a "partner" program that runs on top of Matlab, and gives a GUI interface for building dynamic simulations.

You can take this to the bank: in my opinion, Simulink is the hottest thing to hit scientific computing since Fortran.

Unless you're old enough to be able to remember the "good old days" of Project Apollo and similar programs, you will have difficulty appreciating just how important simulation programs were, and have since become. I'm talking about simulations of dynamic systems, which can include anything that moves, such as cars or spacecraft, but also anything that changes with time, like chemical reactions or populations of rabbits and foxes.

If you want to know how a car will work, or how rabbit populations will change, you can always build the car or breed the rabbits to find out. But if you want to know how to get to the Moon and back, you'd danged well better be pretty sure you can do it, *before* you light that skyrocket.

This month, Jack gets back to some Good Guys, his childhood (and adult) dreams of building legions of robots, and minima.

My first simulation was one for a solid-fuel space booster. It was written in assembly language for an IBM mainframe, because Fortran hadn't yet been invented. My second, a simulation of Apollo, was written in what would now be called Fortran I, except we didn't call it that, because we never thought there would be a Fortran II.

Needless to say, those early simulations were crude; they took a long time to write. We were learning as we went, and the algorithms for things like numerical integration weren't well known. In years to follow, we got better at writing them.

Even so, for many years the words "computer simulation" were enough to make even the toughest managers blanch and feel faint. From experience, they knew what was coming next: a multi-person, multi-year project fraught with massive cost and schedule overruns and a horrendous debugging/validation process. For years after I'd learned how to write good simulations in a short time, my biggest hurdle was convincing managers that I could really do what I said I could do, without the kinds of costs and overruns to which they had become accustomed.

To paraphrase a book on Simulink I recently read, even as recently as 10 years ago, if you wanted to simulate a dynamic system, you really had no choice but to write a simulation from scratch. To be sure, we eventually got better at writing them, and we got software components that helped, but virtually every new simulation was a custom job, from start to finish, and it took time. Lots of time. And even after the program was completed, there was always the problem of demonstrating that the methods you used were really accurate enough to give meaningful results. In other words, you had to validate—a decidedly nontrivial exercise.

Nowadays, with a tool like Simulink, the problem has become almost trivial. In the olden days, we used to begin a new simulation by drawing out the design as a control system flow chart. We'd have boxes for

For years, I've dreamed of building and playing with robots of one sort or another. As early as age 12, I literally dreamed that I had built a robot of the C3PO class that would obey my every command.

the various components and lines connecting them to show how data flowed between boxes. That control diagram became the guiding light, the master plan for how the simulation program would behave.

With Simulink, the diagram *is* the program. Using a GUI interface, you drag boxes into a worksheet and connect them as needed. When you have the diagram drawn, you click "start" and your results are ready. It's as simple as that. The GUI interface of Simulink makes Visual BASIC look like assembly language. There is *no* programming to do at all. When the diagram is done, so is the simulation.

I can't begin to tell you how the availability of such a tool can change the way things get done. Just the other day, some colleagues and I were concerned with whether or not adding a new component to a dynamic system—a component that would tend to couple previously independent control loops—would destabilize the system. In the old days, to answer such a question would have involved either a complex simulation program, or a long and complicated stability analysis.

Using Simulink, I sat down and did a simplified version of the existing system. Then I added the new component. In a matter of about 20 minutes, we discovered two things:

- The new component did indeed destabilize the system
- We found a simple change that stabilized it again

The time saved in having a tool like this is, to me, incalculable. It means we are now free to concentrate on other things, such as having bright ideas, instead of slogging around inside variable-step Runge-Kutta integrators. It's great.

Did I mention that Simulink is revolutionizing the business of developing systems? I thought so. I know of at least two companies—companies who provide embedded systems to federal agencies—who have virtually stopped using either Fortran or C. They develop their systems and simulate them in Simulink. When they're done with their simulation, they use another Matlab tool, Real-Time Workshop, and generate the embedded code directly from the simulation. The engineers design and maintain the entire safety-critical and highly controlled software for the system, using Simulink alone.

If there's one problem with Matlab and its friends, it's the cost. Matlab is great, but it is *not* cheap. It should come as no surprise, then, that other companies have begun offering Matlab companions or Matlab equivalents, at lower cost. One that's come to my attention recently is Mathtools (*www.mathtools.com*), makers of Mideva and quite a number of other products. I won't say much more about them now because I'm planning a review for the near future, but on the face of it the products look great, and I've heard very good things about them.

Fun and games

"All work and no play makes Jack a dull boy," and I certainly have no desire to be dull. For years, I've dreamed of building and playing with robots of one sort or another. As early as age 12, I literally dreamed that I had built a robot of the C3PO class that would obey my every command. Using my trusty Gilbert Erector Set, I proceeded to try to build what I'd dreamed of. Sadly, I learned that my engineering skills and Gilbert's technology weren't quite up to the challenge, and the reality fell considerably short of the dream.

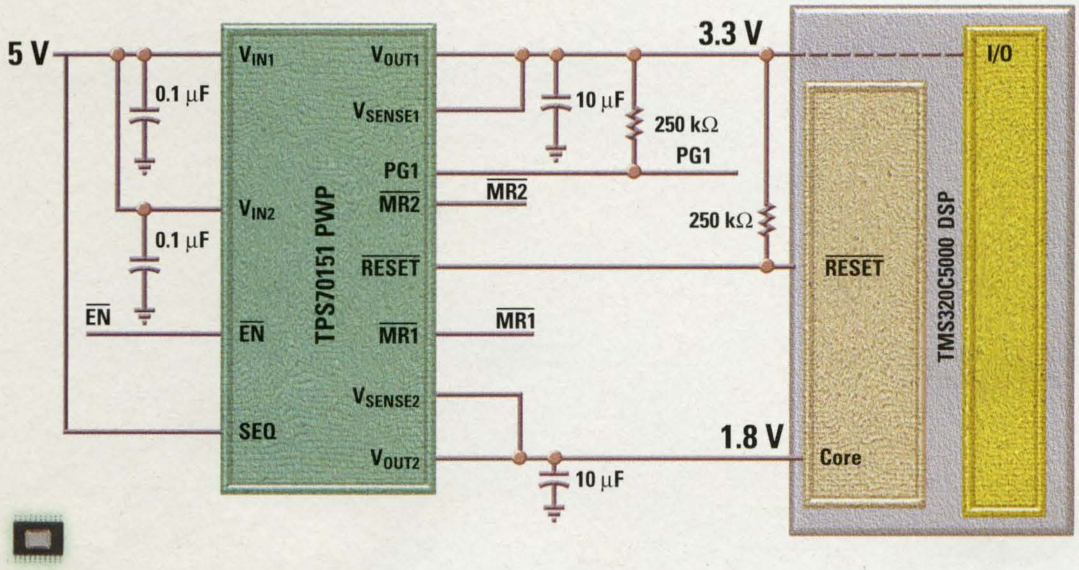
Nevertheless, that dream keeps popping up in one guise or another over the years. It's one of the reasons I continue to tinker around with electric motors, pneumatic systems, and controllers. When Heathkit came out with their ill-fated HERO 1 robot, I thought I'd found the answer to my dreams, but that one just didn't quite make the grade.

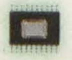
Fortunately for me, the dream is slipping ever closer to reality. A few years ago, I discovered a wonderful company that was, apparently, a secret to no one but me. FischerTechnik is the modern, German equivalent of Gilbert, and they manufacture the European equivalent of the old Gilbert Erector Set. You don't hear much about FischerTechnik in the States, for the simple reason that they don't see much reason to do aggressive marketing here. They're already producing at top speed to meet the demands of the European market alone.

FischerTechnik makes a whole range of kits, mostly for kids, but they also sell some robotics kits that are intended mostly for the educational trade (many universities use them to train budding industrial automation engineers). Like Matlab, they are not cheap—my first three kits cost $700, $500, and $500, and that was in 1990 dollars. But you will not *believe* the quality, which follows the traditions of German engineering. Beautiful, extruded aluminum beams replace the stamped-steel parts of the erector set. Self-stabilizing plastic connectors replace the old screws and square nuts. The electric motors are marvels of modern engineering—small, powerful, smooth, and quiet. Exquisite.

Most importantly to a budding roboticist, the FischerTechnik kits include controllers for both stepper and proportional motors, and a com-

DUAL-OUTPUT LDOs SIMPLIFY POWER-UP/POWER-DOWN SEQUENCING IN SPLIT-VOLTAGE DSP SYSTEMS.



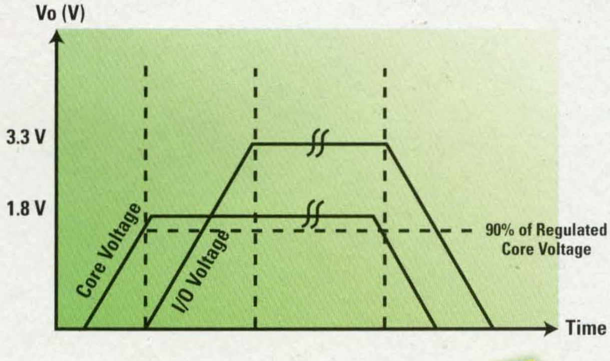


20-pin TSSOP PowerPAD™ package

POWERING A TMS320C5000™ DSP USING THE TPS70151

- *Complete power solution from Texas Instruments for DSP, microprocessor, ASIC and FPGA applications*
- *Logic-selectable power-up sequencing prevents bus-contention possibilities*
- *Dual low-voltage outputs with fixed options for DSP core and I/O voltages*
- *250-mA/500-mA output current*
- *Integrated processor reset (SVS) with 120-msec delay*
- *190-µA typical quiescent current ensures longer battery life*
- *TI's TPS701xx from only $2.47 per device in quantities of 1,000*

Sequential power-up and power-down showing core supply enabling and disabling I/O supply.



To order a free data book or LDO product catalog, call 1-800-477-8924, ext. aap3772u

For data sheets, application notes, samples and EVM availability, visit www.ti.com/sc/tps701xx

Free Power Management Products Data Book

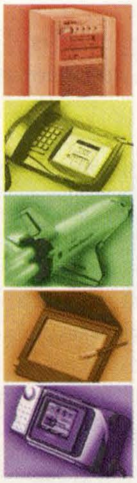
TMS320C5000 and PowerPAD are trademarks of Texas Instruments.

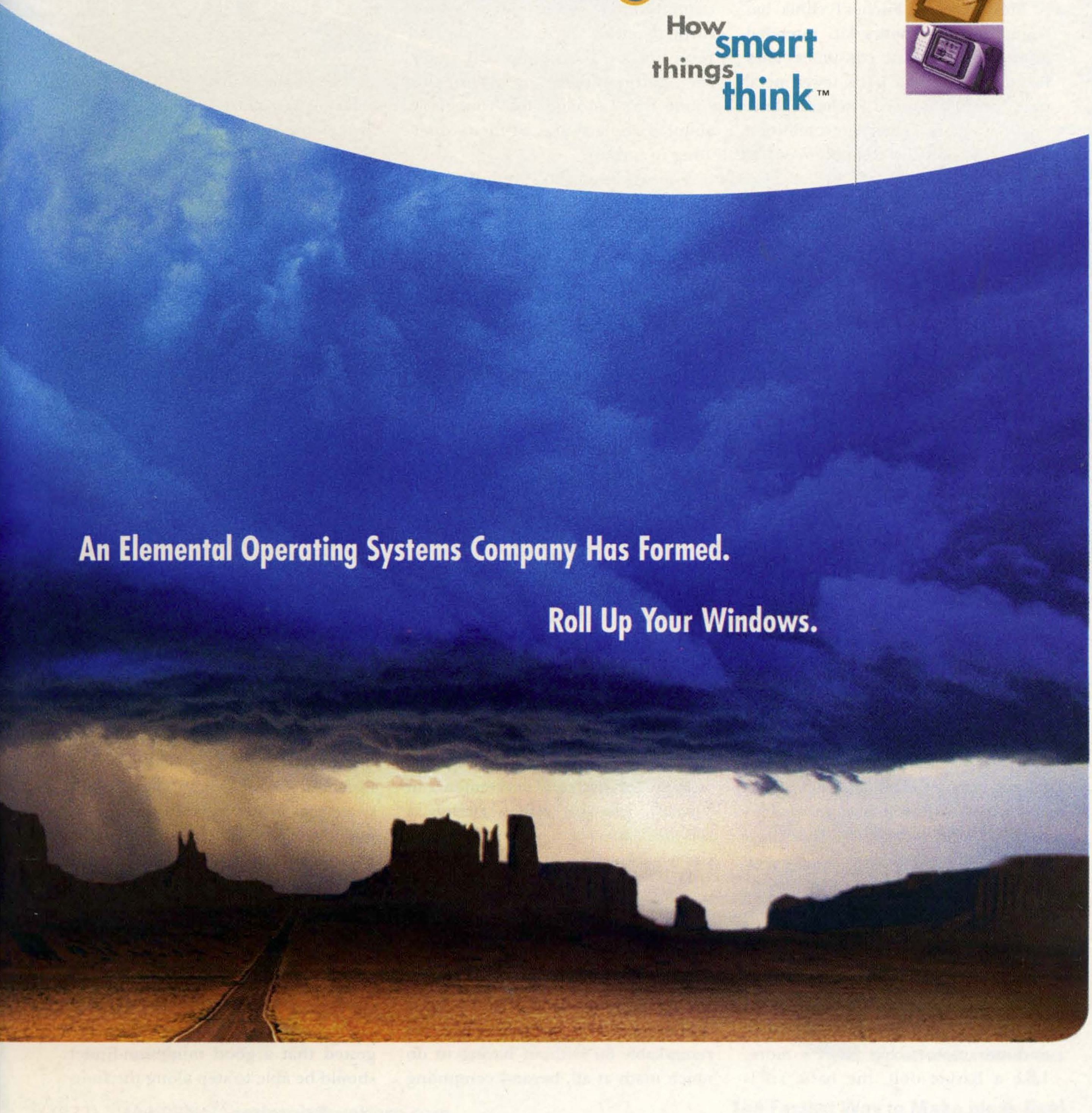
8983-01 © 2000 TI

THE WORLD LEADER IN DSP AND ANALOG

TEXAS INSTRUMENTS

WindRiver®




An Elemental Operating Systems Company Has Formed.
Roll Up Your Windows.

An autonomous, programmable robot isn't enough for you? How about a whole swarm of them, operating in unison?

puter program to control them. The kits using proportional motors have optical encoders to give position feedback. Other kits use other methods, such as potentiometers.

Most recently, FischerTechnik has begun selling robotics kits intended more for the private consumer. They feature more plastic parts, fewer metal ones, but the famous FischerTechnik quality remains. Their latest addition is a kit that provides computer control for pneumatic systems; as far as I know, a first in that area.

Last Christmas, I treated myself to a whole gaggle of FischerTechnik kits, with more on the way. I can hardly wait!

Leggo my Lego!

Speaking of robotics: remember Lego? You know, those crazy plastic blocks that would never stay together? Well, they still don't stay together very well, or at least the poorly built things don't (you have to use a little engineering, after all), but Lego may very well have the hottest thing going in terms of robotics. It's the Lego Mindstorms Robotic Invention System (RIS). The key component of this system is the "smart brick," an oversized Lego block called the RCX controller. I understand that the RCX was developed for Lego by a team of MIT wizards, and it shows. Inside is a microcomputer. Unlike the FischerTechnik systems, which require cables back to your computer, the Lego RCX is battery powered and IR-programmable, which is another way of saying that the Lego robots have cut the umbilical cord. They can operate autonomously.

The standard RIS includes two touch sensors (also known as microswitches), two optical sensors, and two motors, but there's more. Like a Barbie doll, the basic kit is only the beginning. Lego will be happy to sell you more motors, more sensors of all kinds, and more controllers that can work with the RSX. Lego robots can even—get this—communicate with each other, through either IR transmitters and receivers, or short-range CB radio. An autonomous, programmable robot isn't enough for you? How about a whole swarm of them, operating in unison?

For anyone who's interested in embedded systems, as I presume you readers are, and especially those who wish they knew more about how to control physical sensors and actuators, I can't think of a more enjoyable way to learn than to tinker with these robots. Space prevents me from mentioning all the sources for such gems, but fortunately, I don't have to; I need only give you the right search keywords for your Internet browser. Try the terms "construction," "robot," and "kits" in various combinations. Also, look for products by FischerTechnik, Lego, and K'Nex (not robots, but still fun). As you do, you will discover one more facet of owning a Lego Mindstorms kit: Lego gives each buyer a free Web page to display pictures and plans for his/her inventions. You'll surely run across several such Web sites.

Enjoy. Learn by doing. If what you find there doesn't give you even a little hankering to buy your kid a Lego toy (strictly for his own amusement, of course), you are either already a robotics expert with your own lab, or you would make a great accountant.

Back to minima

Now let's get back to the main problem at hand, searching for good ways to find the minimum of an arbitrary function. We've managed to go remarkably far without having to do much math at all, beyond computing new trial values of the function, and evaluating it at each step. That situation is about to change. The methods we'll be talking about next all require considerably more mathematical rigor than we've needed so far.

Since I haven't given our test function lately, and some of you readers have asked for it, here it is again:

$$f(x) = \cos\left(2\pi x^3\right) \tag{1}$$

The power of three in the function makes it just crazy enough, and also asymmetric enough, to keep it from being a trivial function. From elementary trigonometry, we know that $f(x)$ has a minimum value of –1, and that minimum occurs when the argument is equal to π. So the value of x at the minimum is:

$$x_f = \sqrt[3]{\frac{1}{2}} = 0.7937005 \tag{2}$$

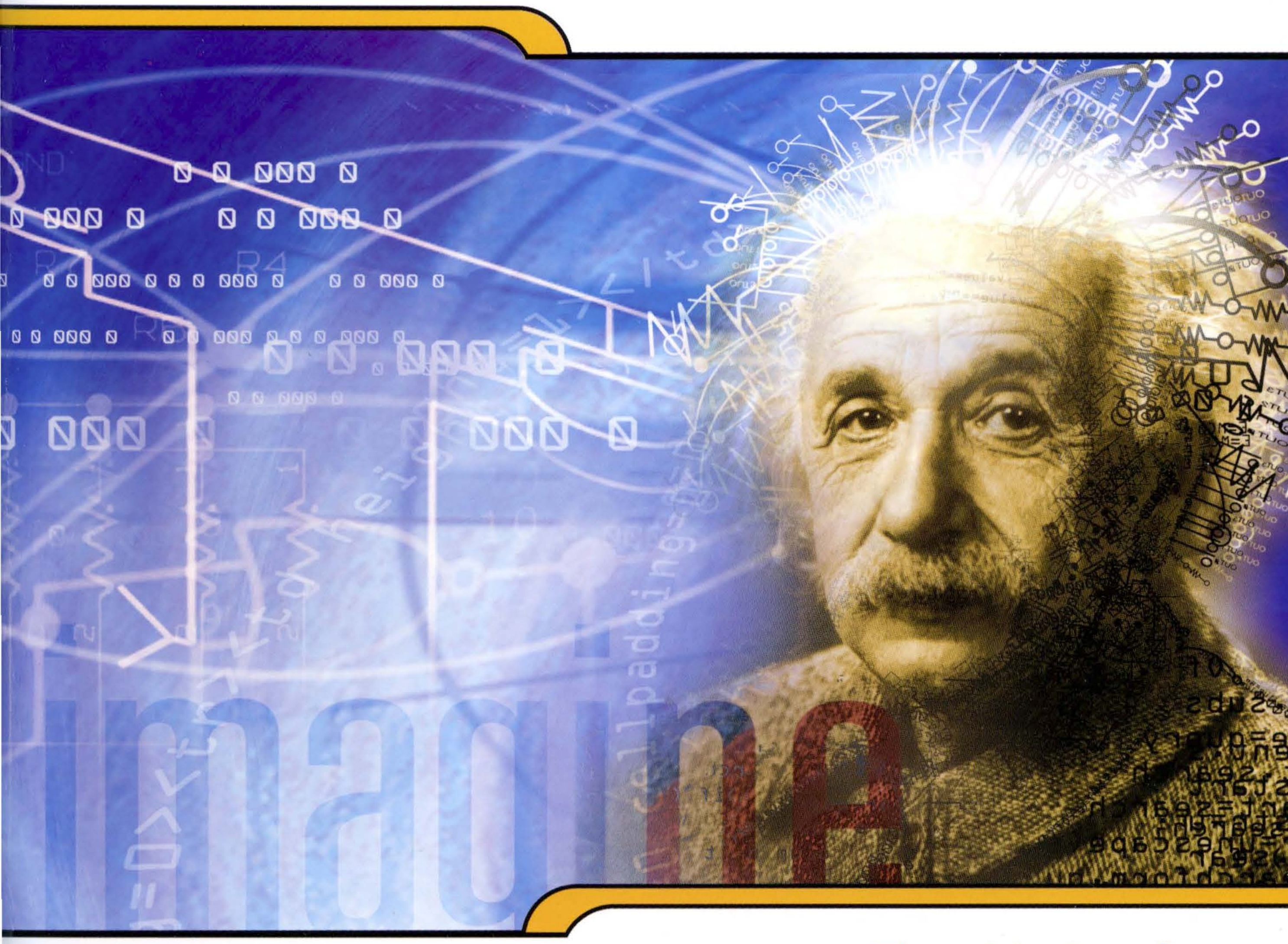
In the process of working up the code for last month's episode, I also settled on a second, very similar function:

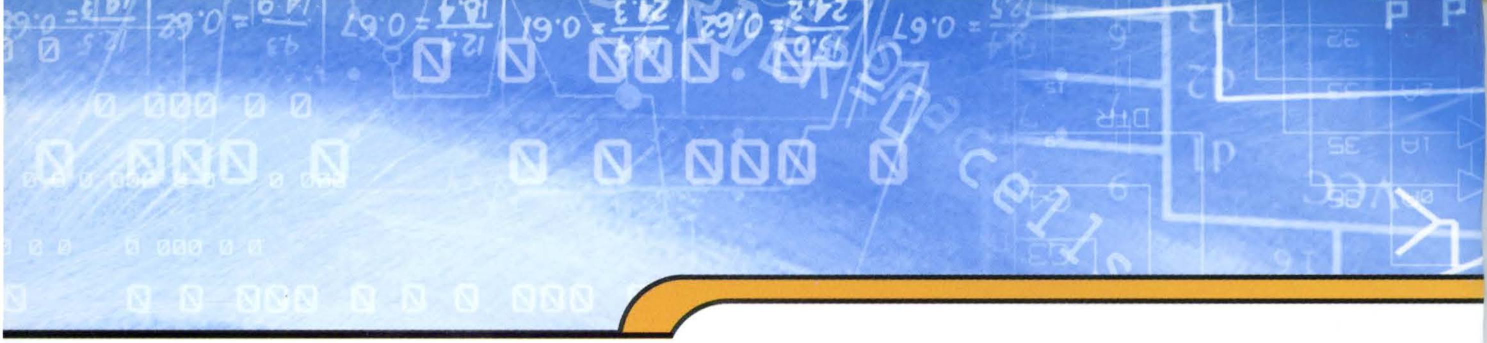
$$g(x) = \sin\left(2\pi x^3\right) \tag{3}$$

Its minimum occurs when the argument is $3\pi/2$, which leads to:

$$x_g = \sqrt[3]{\frac{3}{4}} = 0.90856 \tag{4}$$

The two functions are plotted together in Figure 1. Each function has its advantages for testing. The function $f(x)$ is well behaved in the sense that it has only one minimum, but it starts out very flat, which can cause trouble for some search methods. The function $g(x)$ really has two minima, one of which occurs at $x = 0$. According to our originally stated criterion, which is that we will prefer the leftmost minimum, we should, by all rights, accept this value immediately. However, several readers have suggested that a good minimum-finder should be able to step along the func-


imagine



Much more than connecting buyers and sellers . . .

For Product Development Companies:

- A more cost-effective and efficient sourcing process
- Worldwide, pre-qualified expertise and resources
- Supplier base expansion controls
- Secure and anonymous RFP postings
- Emerging technology communities
- Reduced time to market

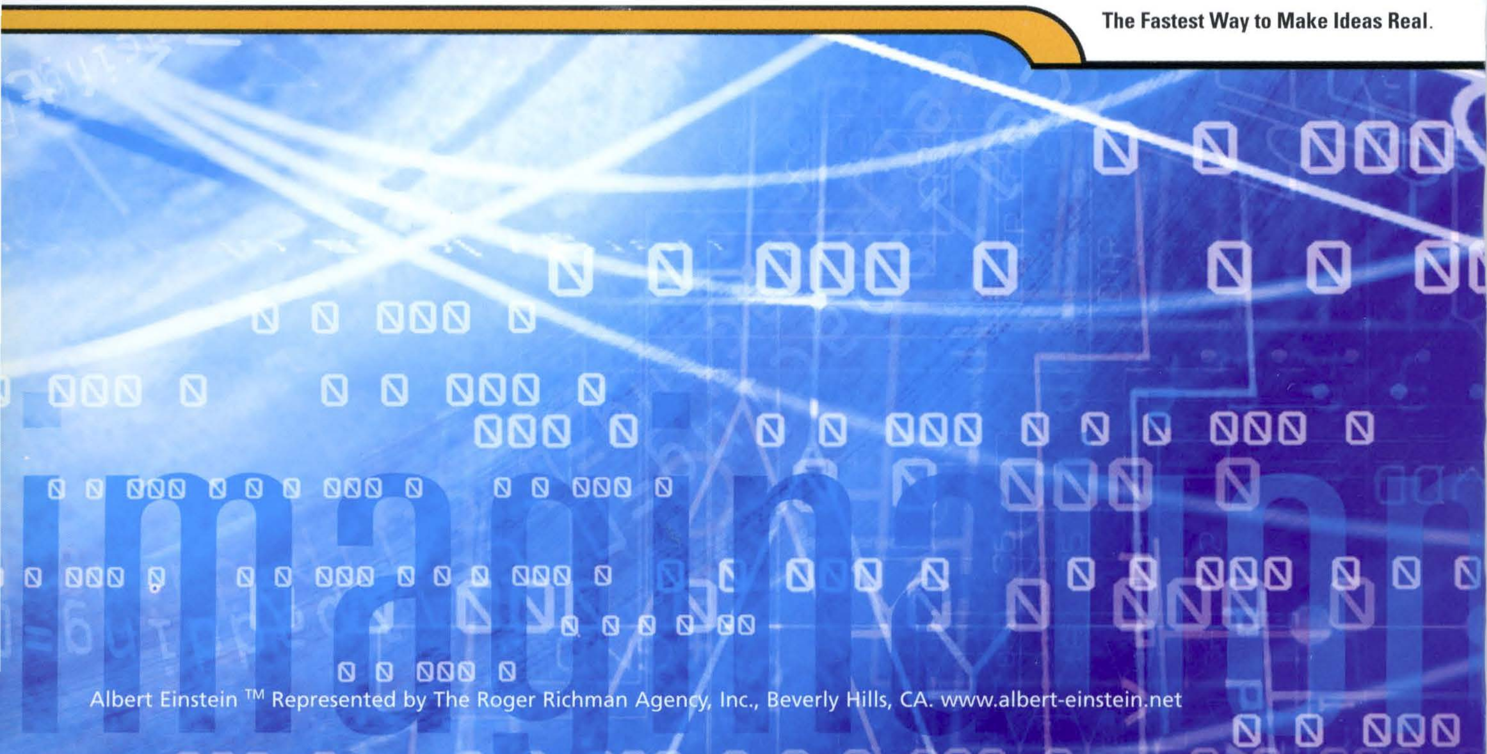
For Service / Intellectual Property Providers:

- Reduced cost of sales
- Opportunities to win the *right* projects
- High visibility to product development companies
- Access to potential symbiotic partners
- Millions in sales and marketing resources deployed on your behalf
- Emerging technology communities

And still there's more. Log in and learn. WebPRN.com

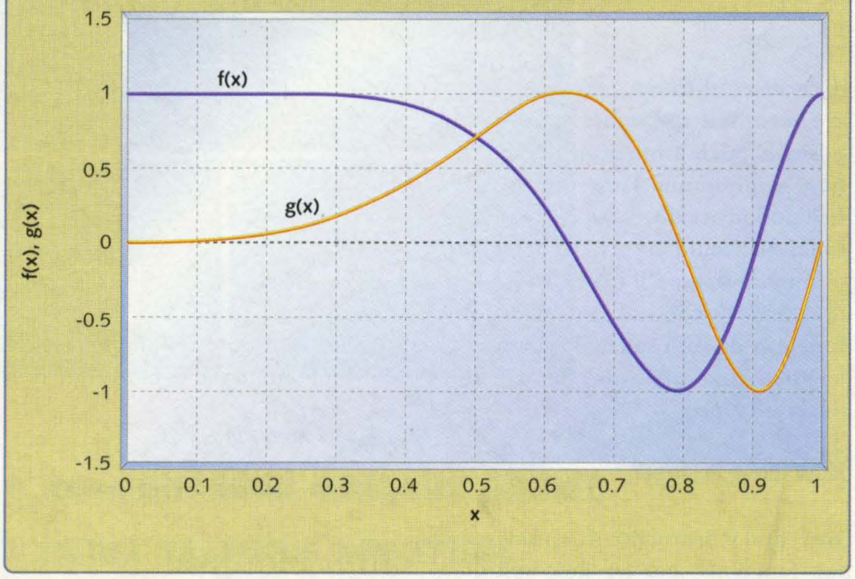
webPRN.com
PRODUCT REALIZATION NETWORK

The Fastest Way to Make Ideas Real.



Albert Einstein ™ Represented by The Roger Richman Agency, Inc., Beverly Hills, CA. www.albert-einstein.net tion until it begins to have a negative slope (assuming that this ever happens), and ignore lower starting values. I thought their argument persuasive, and $g(x)$ is the result. A robust minimum-finder should return the leftmost value if the function never turns south at all, and whatever minimum occurs after that, if it does. The routine I gave last month does do this.

FIGURE 1 Test functions



Getting rigorous

Both functions in Figure 1 share one common characteristic, which is that they are *differentiable*. In mathematical terms, this means that the function, and all its derivatives, are finite, no matter how high an order we choose to go. In layman's terms, the function is smooth, with no surprising discontinuities or other craziness. A key characteristic of such functions is that they can be expressed in terms of a Taylor series:

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + \frac{1}{2!}(x - x_0)^2 f''(x_0) + \frac{1}{3!}(x - x_0)^3 f'''(x_0) + \ldots \quad (5)$$

(where the primes imply the derivatives of successively higher orders, with respect to x). In words of few syllables, if we have $f(x)$ at some value x_0, plus all its derivatives (a potentially infinite number), also evaluated at x_0, we can synthesize the function. Note that the series is exact, not an approximation, as long as we use all the non-zero derivatives of $f(x)$. If the function happens to be a polynomial of some order N, the series will not be infinite, since all derivatives higher than Nth order will be zero.

The choice of the value x_0 is arbitrary, though in practice we try to choose a value that's near the region of interest. That's because, also in practice, we tend to approximate the function by truncating the series.

You may recall my saying, in previous columns, that all well behaved functions look like parabolas when we look at a region sufficiently close to the minimum. Equation 5 gives us rigorous support to that assertion. Look at those coefficients for each term, which have the general form:

$$\frac{1}{n!}(x - x_0)^n \quad (6)$$

If we choose a value of x_0 that's sufficiently close to the true minimum, and restrict our range of focus to a sufficiently narrow range, the value of $x - x_0$ remains small. Its higher powers, then, are even smaller, and this smallness is greatly aided by the appearance of the $n!$ term in the denominator. For these reasons, terms of order higher than two become negligible, and we can truncate the Taylor series to:

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + \frac{1}{2!}(x - x_0)^2 f''(x_0) \quad (7)$$

This function doesn't just look like a parabola; it *is* a parabola. Therefore our statement that all functions with minima look like parabolas is rigorously true if the function is differentiable and the magnitude of $x - x_0$ is sufficiently small.

Now we can also give a rigorous definition for the location of the minimum, if one exists. The usual conditions are given as:

$$f'(x_{\min}) = 0$$
$$f''(x_{\min}) > 0 \quad (8)$$

These two equations state mathematically what we know to be true, intuitively, for a mimimum. At the minimum, the slope is flat (that is, zero), and the function curves upwards in both directions as we leave the minimum. This statement is about as rigorous a definition of the minimum as you're likely to find. Oh, if you look hard enough, you can find a pathological function that has a true minimum, but the conditions of Equation 8 don't hold. For example, we can invent an "artificial" function such that not only the second derivative, but all higher

Having had this idea that functions look much like parabolas near their minima, it seems only natural to try to fit the data by assuming a parabolic shape.

derivatives through, say, the ninth, are zero, but the tenth derivative is positive. Such a function really does have a minimum. However, in practice nobody really cares, because the function would be so flat near the minimum that we'll never find it anyhow. For all practical purposes, if the function doesn't satisfy Equations 8, we'll throw up our hands and declare failure.

How close is close?

We'll find it instructive to consider our test functions, and see how well they match the "looks like a parabola" description. The derivatives of these functions turn out to be:

$$\begin{aligned}f'(x) &= -6\pi x^2 \sin(2\pi x^3) = -6\pi x^2 g(x)\\ f''(x) &= -12\pi x \sin(2\pi x^3) - (6\pi x^2)^2 \cos(2\pi x^3)\\ &= -12\pi x g(x) - (6\pi x^2)^2 g(x)\end{aligned} \tag{9}$$

Similarly,

$$\begin{aligned}g'(x) &= 6\pi x^2 \cos(2\pi x^3) = 6\pi x^2 f(x)\\ g''(x) &= 12\pi x \cos(2\pi x^3) - (6\pi x^2)^2 \sin(2\pi x^3)\\ &= 12\pi x f(x) - (6\pi x^2)^2 g(x)\end{aligned} \tag{10}$$

Because $f(x)$ and $g(x)$ are related as cosine and sine, you probably won't need much convincing that $g(x) = 0$ when $f(x)$ is at its minimum, and vice versa. One look at Figure 1 should help. Nevertheless, for the sake of rigor, we'll go ahead and evaluate the functions.

$$\begin{aligned}g\left(\sqrt[3]{\frac{1}{2}}\right) &= \sin\left(2\pi\left(\sqrt[3]{\frac{1}{2}}\right)^3\right)\\ &= \sin\left(2\pi\frac{1}{2}\right)\\ &= \sin(\pi) = 0\end{aligned} \tag{11}$$

Similarly,

$$\begin{aligned}f\left(\sqrt[3]{\frac{1}{2}}\right) &= \cos\left(2\pi\left(\sqrt[3]{\frac{3}{4}}\right)^3\right)\\ &= \cos\left(2\pi\frac{3}{4}\right)\\ &= \cos\left(\frac{3\pi}{2}\right) = 0\end{aligned} \tag{12}$$

Thus Equations 9 and 10 become, at the minimum values of the function:

$$\begin{aligned}f'(x_f) &= -6\pi x_f^{\,2} g(x_f) = 0\\ f''(x_f) &= -(6\pi x_f^{\,2})^2 f(x_f) = (6\pi x_f^{\,2})^2\end{aligned} \tag{13}$$

$$\begin{aligned}g'(x_g) &= 6\pi x^2 f(x_g) = 0\\ g''(x_g) &= -(6\pi x_g^{\,2})^2 g(x_g) = (6\pi x_g^{\,2})^2\end{aligned} \tag{14}$$

Substituting the values of x_f and x_g gives:

$$\begin{aligned}f''(x_f) &= 141.0031837\\ g''(x_g) &= 242.1125289\end{aligned} \tag{15}$$

So our approximate functions are:

$$\begin{aligned}f(x) &\approx -1 + 70.50159184(x - 0.793700526)^2\\ g(x) &\approx -1 + 121.0562645(x - 0.908560296)^2\end{aligned} \tag{16}$$

How accurate are these approximations? To find out, let's evaluate them at values of x that are fairly far from the true minima; far enough, say, to cause a difference of 5% in the full function range. That is, the function should be –0.9 instead of –1.0. My calculator says that the variations in x should be about 0.038 in $f(x)$, and 0.029 in $g(x)$. At these values (for $f(x)$, a range of nearly 10% in x), the differences between the approximate and exact functions is on the order of 0.5% or less. Obviously, if we narrow the range of x down further, the range of error will also drop dramatically. Surely this justifies our assertion that, in the vicinity of the minimum, all differentiable functions look like parabolas.

A little curve-fitting

Having had this idea that functions look much like parabolas near their minima, it seems only natural to try to fit the data by assuming a parabolic shape. Suppose we have three points, denoted by the ordered pairs:

$$\begin{aligned}P_0 &= [x_0, y_0]\\ P_1 &= [x_1, y_1]\\ P_2 &= [x_2, y_2]\end{aligned} \tag{17}$$

What we'd like to do is find the parabola that passes through them. We know that the equation of a parabola is a quadratic equation, with the general form,

$$F(x) = a + bx + cx^2 \tag{18}$$

Conceptually, we can require that this equation satisfy the constraints that:

$$\begin{aligned}F(x_0) &= y_0\\ F(x_1) &= y_1\\ F(x_2) &= y_2\end{aligned} \tag{19}$$

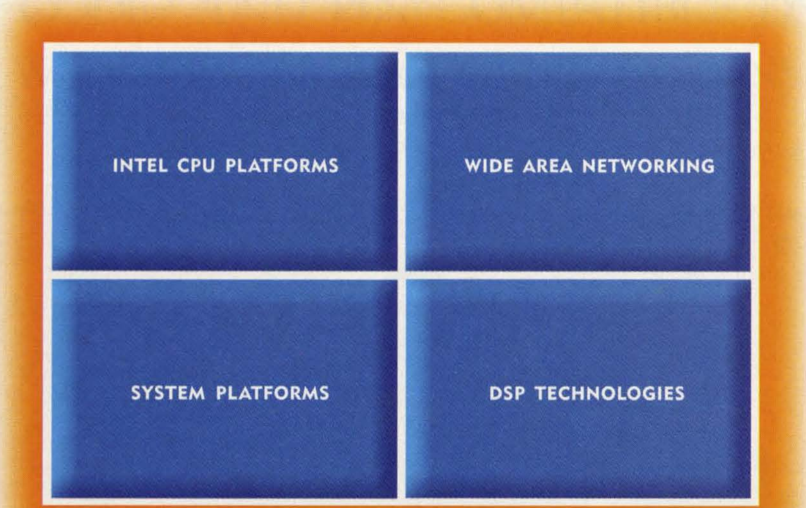
This gives us three equations to solve for the three unknowns a, b, and c.

You're welcome to try this approach, and you *will* get an answer, given enough perseverance. But you'll be doing a lot more work than is necessary. A better approach is to use Newton's form for the quadratic:

$$F(x) = a + (x - x_0)[b + c(x - x_1)] \tag{20}$$

The nice part about this form is that the entire x-dependent part vanishes when $x = x_0$, and the quadratic part vanishes at $x = x_1$. This makes it much easier to evaluate the coefficients. (Hint: file this little trick away for future reference. You will find that it saves you tons of work.)

Using this form, Equations 19 become:



HERE'S A NO-NONSENSE MESSAGE ABOUT OUR TOTAL TELECOM SOLUTION. DON'T BE FOOLED BY ITS SIMPLICITY.

Only RadiSys offers the total telecommunications solution.

A solution this complete is hard to pull off. A lot of vendors offer different parts of telecom solutions. Only RadiSys has the vision and technical expertise to deliver the total, integrated solution across both PCI and CompactPCI architectures. RadiSys has acquired Texas Micro, and is now positioned to redefine how the market is buying OEM embedded telecommunications systems. Now you can let us put the total solution together. Download our comprehensive white paper on SS7, and find out how simple it is to get the total solution from one source at www.radisys.com, or call 1-877-837-6859.

- **INTEL CPU PLATFORMS**
  State of the art Pentium,® PII, PIII, IXP and StrongArm solutions
- **SYSTEM PLATFORMS**
  Enterprise and carrier class CompactPCI and PCI system enclosures and backplanes
- **WIDE AREA NETWORKING**
  Field-proven frame relay, SS7, ATM, T1/E1, X.25, IP and other protocols and interfaces
- **DSP TECHNOLOGIES**
  Advanced TI C6x voice processing solutions-hardware and algorithms

Download FREE white paper today!

www.radisys.com/SS7

Intel® Applied Computing Platform Provider

RadiSys.

$$
\begin{aligned}
&a = y_0 \\
&a + (x_1 - x_0)\,b = y_1 \\
&a + (x_2 - x_0)\,[b + (x_2 - x_1)\,c] = y_2
\end{aligned}
\tag{21}
$$

From the first equation, of course, we get the value of a directly. Substituting it into the second gives:

$$
b = \frac{y_1 - y_0}{x_1 - x_0} \tag{22}
$$

We'll find life a lot simpler here if we introduce the familiar Δ-notation, also called the forward difference:

$$
\Delta x_n = x_{n+1} - x_n \tag{23}
$$

Then:

$$
b = \frac{\Delta y_0}{\Delta x_0} = m_0 \tag{24}
$$

which we recognize as simply the slope of the straight line between P_0 and P_1.

The third of Equations 21 may now be written:

$$
y_0 + (x_2 - x_0)\,[b + (x_2 - x_1)\,c] = y_2
$$

or

$$
c\Delta x_1 = \frac{y_2 - y_0}{x_2 - x_0} - \frac{\Delta y_0}{\Delta x_0} \tag{25}
$$

At first glance, the terms like $y_2 - y_0$ don't seem to fit very well into our difference operator scheme; but look again. We can always write:

$$
\begin{aligned}
y_2 - y_0 &= y_2 - y_1 + y_1 - y_0 \\
&= \Delta y_0 + \Delta y_1
\end{aligned}
\tag{26}
$$

and similarly for the x's. Thus:

$$
c\Delta x_1 = \frac{\Delta y_0 + \Delta y_1}{\Delta x_0 + \Delta x_1} - \frac{\Delta y_0}{\Delta x_0}
$$

With a little mathematical manipulation, we can write:

$$
\begin{aligned}
c\Delta x_1 &= \frac{(\Delta y_0 + \Delta y_1)\Delta x_0 - (\Delta x_0 + \Delta x_1)\Delta y_0}{(\Delta x_0 + \Delta x_1)\Delta x_0} \\
&= \frac{\Delta x_0 \Delta y_1 - \Delta x_1 \Delta y_0}{(\Delta x_0 + \Delta x_1)\Delta x_0} \\
&= \frac{\Delta y_1 - \dfrac{\Delta x_1}{\Delta x_0}\Delta y_0}{\Delta x_0 + \Delta x_1}
\end{aligned}
$$

Then:

$$
\begin{aligned}
c &= \frac{\dfrac{\Delta y_1}{\Delta x_1} - \dfrac{\Delta y_0}{\Delta x_0}}{\Delta x_0 + \Delta x_1} \\
&= \frac{m_1 - m_0}{x_2 - x_0}
\end{aligned}
\tag{27}
$$

This formulation gives us an easy way to generate the coefficients of the polynomial. Take the differences of successive y's to get slopes (m's), and the differences of successive slopes to get c. (Confession: if you didn't see this particular move coming, don't feel bad. I've been doing this kind of thing for over 40 years, and I just saw the trick a few months ago! Newton saw it a bit earlier.)

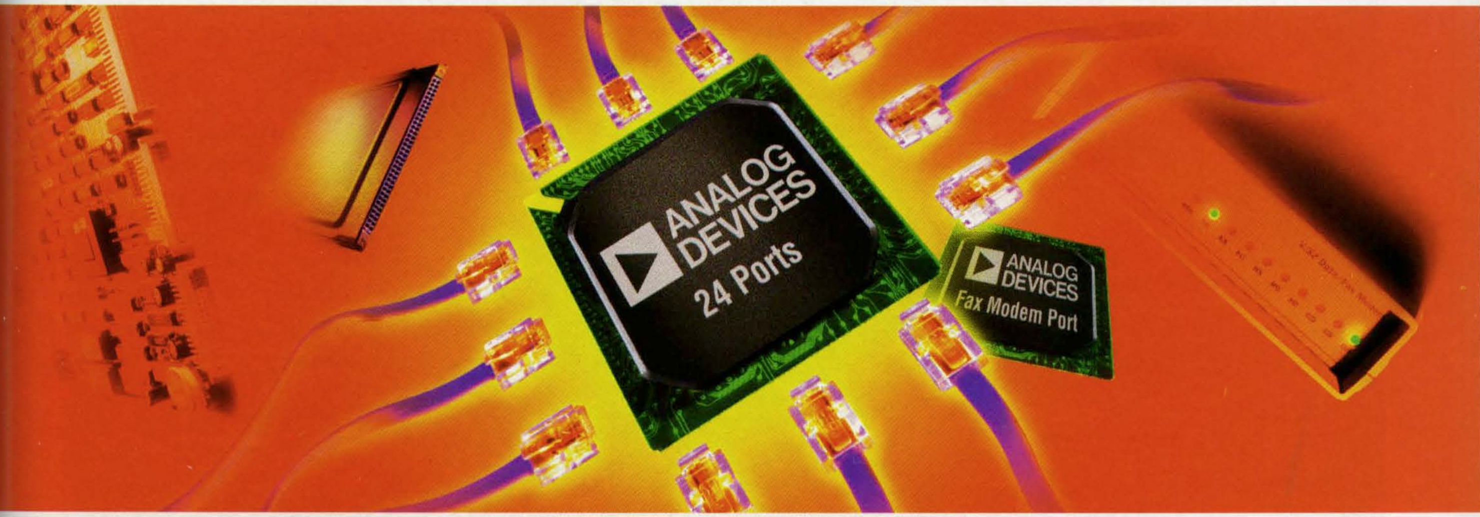
Okay, let's assume that we have the equation for the function-approximating parabola. How do we get the location of the assumed minimum? That's simple: we differentiate the equation, and require that the slope be zero. From Equation 20:

$$
\begin{aligned}
F'(x) &= \left[b + c(x - x_1)\right] + (x - x_0)c \\
&= b + c\left(2x - (x_1 + x_0)\right) = 0
\end{aligned}
$$

Solving this for x, we get:

$$
\begin{aligned}
&2x_{\min} - (x_1 + x_0) = \frac{-b}{c} \\
&2x_{\min} = x_1 + x_0 - \frac{b}{c} \\
&x_{\min} = \frac{1}{2}\left(x_1 + x_0 - \frac{b}{c}\right)
\end{aligned}
\tag{28}
$$

WITH OUR CODE- AND PIN-COMPATIBLE DSPS, YOUR BEST WORK LIVES FOR GENERATIONS.



The ADSP-218x 16-bit DSP family. Fixed-point processing made easy.

The only thing more difficult than writing code is rewriting it—unless you're designing with 16-bit DSPs from Analog Devices. Our roadmap of over 250 code- and 100 pin-compatible 16-bit DSPs makes your design challenge easier. Start with the exact speed, power, and memory version you need from our popular ADSP-218x family. Complete your initial code faster using our simple algebraic syntax for assembly programming, and hardware features like zero overheading looping. Switch to a smaller memory size as the code matures, even if you are in production, without a hardware restart. Or offer multiple features and price points from a single design. Whether you need the world's smallest DSP form factor, power consumption as low as .4mA/MIP, SRAM integration up to 2 Mbits, or high-performance data converters for easy integration, no one can provide you with more choices and design advantages than the DSP supplier with the unlikely name. Analog Devices.

Introducing the ADSP-2186M.
A 15-year tradition of compatibility. This newest family member offers:

- *75 MIPS for a $4.95* price per unit*
- *100-lead LQFP and 10mm x 10mm mini-BGA packages*
- *320 Kbits on-chip SRAM, with a pin-compatible roadmap to 2 Mbits*

For FREE samples, data sheets, and an evaluation board:
Visit www.analog.com/16-bit or call 1-800-ANALOGD.

ANALOG DEVICES

DSP SOLUTIONS THAT MAKE YOUR DESIGN CHALLENGE EASIER.™

*ADSP-2186M USD per unit in 25,000 unit volumes. Ad Code 6134

Before we just jump in and start writing code, we must tame the method to the point where we are 100% sure we can trust its conclusions, and even more importantly, we'll know when we cannot.

FIGURE 2 Homing in

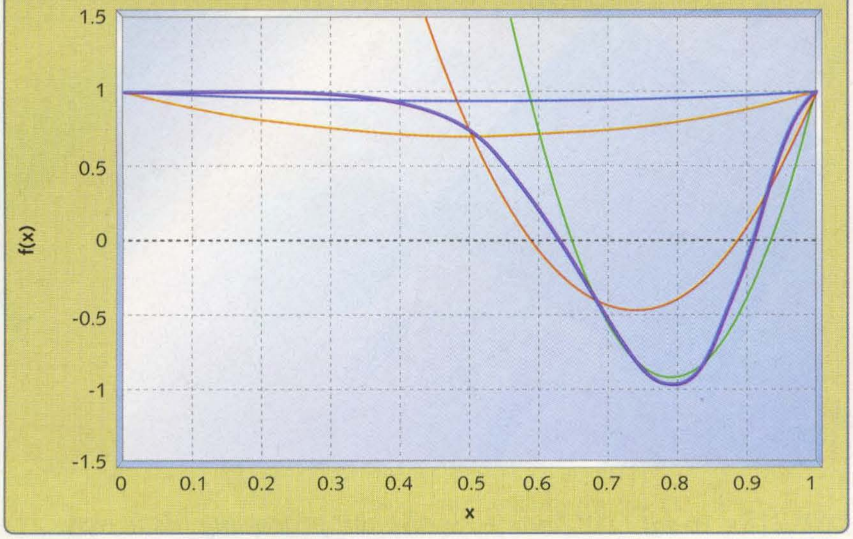


TABLE 1 First four points

x	f(x)
0	1
0.4	0.92023
0.5	0.70711
1	1

What does it all mean?

Now that we've done the math, what have we got exactly? What we have is a way to estimate the location of the minimum, given three points on the functional curve. We've already seen that most functions look like a parabola near the minimum, so our three points should give us a super estimate of the location of the true minimum. We have every right to expect that we should get much more rapid convergence than we would using a simple bisection algorithm. Do we actually get such rapid convergence? Let's find out.

We'll begin with our standard test function, $f(x)$, and bisect the interval to get a middle point lower than the other two (forget the Golden Section, at this point; it only complicates the math). We have, then:

$$x_0 = 0 \qquad y_0 = 1$$
$$x_1 = 0.5 \qquad y_1 = 0.70711$$
$$x_2 = 1 \qquad y_2 = 1$$

$$m_0 = -0.5858 \qquad m_1 = 0.5858$$

$$c = 1.716 \qquad x_{\min} = 0.5$$

Oops! Houston, we have a problem: the algorithm says that the x-value of the minimum is the same as x_1, which means that, as far as it's concerned, we have already found the minimum. We know that's not true, but how are we going to tell the algorithm that it's wrong?

You can see that we will always get this kind of result, if the center point is obtained by bisection, and the values of $f(x)$ at the other two points are equal. The algorithm will dutifully fit a parabola that's symmetric about x_1, and will steadfastly insist that it's found the minimum.

In retrospect, perhaps we would have been better off to use the Golden Section after all. At least we wouldn't have hit this crazy symmetric case. But, of course, there was method to my madness: I wanted you to see what kinds of problems we can encounter. It should be clear at this point that, while our method of fitting quadratics has the potential for greatly speeding up the search for the minimum, it is definitely not ready for prime time. Before we just jump in and start writing code, we must tame the method to the point where we are 100% sure we can trust its conclusions, and even more importantly, we'll know when we cannot. That process will take a little time.

Okay, so we're not quite ready to start committing the algorithm to code. Let's at least see what we can do by applying the method manually. We've seen that symmetric cases don't work, so we'll arbitrarily tweak the center point to a new value:

$$x_0 = 0 \qquad y_0 = 1$$
$$x_1 = 0.4 \qquad y_1 = 0.9202$$
$$x_2 = 1 \qquad y_2 = 1$$

$$m_0 = -0.1994 \qquad m_1 = 0.1329$$

$$c = 0.3324 \qquad x_{\min} = 0.5$$

Whoops! That didn't seem to help either. And you can easily see why. Think of the points P_0 and P_2 as the end points of a wooden bow, and the line between them as the bowstring. We can bend the bow up or down by choosing points that are left or right on $f(x)$, and therefore higher or lower in the y-direction. No matter how we do this, however, we cannot warp the bow so that its minimum is at anywhere but $x = 0.5$. As long as $y_0 = y_2$, the parabola will be symmetric about this value.

At this point, it may seem that we've gotten nowhere at all. But don't despair: we've made more progress than you think. Let's tabulate the data we have so far. Table 1 doesn't give us

TABLE 2 Quadratic fit, manual iteration

x_0	x_1	x_2	y_0	y_1	y_2	x_{min}	y_{min}
0	0.4	1	1	0.92023	1	0.5	0.70711
0.4	0.5	1	0.92023	0.70711	1	0.68532	-0.43639
0.5	0.68532	1	0.70711	-0.43639	1	0.73636	-0.80634
0.68532	0.73636	1	-0.43639	-0.80631	1	0.79172	-0.99973
0.73636	0.79172	1	-0.80631	-0.99973	1	0.79921	-0.99783
0.73636	0.79172	0.79921	-0.80631	-0.99973	-0.99783	0.79334	-0.99999
0.79172	0.79334	0.79921	-0.99973	-0.99999	-0.99783	0.79369	-1

much to go on, but, to quote Shakespeare's Tybalt in *Romeo and Juliet*, "T'is enough—t'will serve." In the last three entries, we have three points that satisfy our minimum-seeking criterion: the middle point is lower than the other two. That's all we need to start an iteration process.

In Table 2, I've repeated the computations of Equations 24 through 27, manually choosing the lowest point and its two neighbors as the three points for the next iteration.

We can see two things from Table 2. First, the method definitely works. Including the first probing values of 0.4 and 0.5, we get the solution in only seven iterations, which is a lot fewer than the 20 or so we were needing using bisection.

In Figure 2, I've shown the parabolas fitted by the first few iterations. It's easy to see that the method has definitely gotten the scent of the true minimum, and is rapidly homing in on it.

The second thing we can learn from Table 2 is that the method still needs a lot of work. Our first clue is the way the right-hand limit, x_2, seems stuck at 1.0 for so many iterations. Even when it comes unstuck, it seems to get stuck again at 0.79921.

Looked at in another way, during the time when the interval between x_0 and x_1 is reduced from 0.4 to 0.05, the overall interval from x_0 to x_2 remains stuck at about 0.264. This is not a good sign. One of the nice things about the bisection (also including the Golden Section search) methods was that we could count on the total interval being inexorably reduced during the search process. In fact, we used the width of this interval as our criterion to decide when we'd converged. As it stands now, the quadratic-fit method doesn't give us the same luxury.

A second problem is related to the first. In the last iteration, the distance between the points x_0 and x_1 is only 0.0016, which is only 20% of the total interval. I don't like the idea of any two of the three points getting too close together; it implies a risk of a poor parabolic fit because of roundoff error. I'd feel much more comfortable if we could make sure that both x_0 and x_2 are pulled in, in a roughly symmetrical manner.

What we have learned, then, is that while the method of fitting a parabola to the data is promising, we still have a lot of taming to do before it's ready for prime time as a robust minimum-finder. Since we were able to maintain a roughly symmetrical balance using bisection methods, it should come as no great surprise to you that the most robust methods are those that combine bisection and quadratic fit. That's the area that we'll concentrate on next month.

See you then. **esp**

Jack W. Crenshaw is a senior principal design engineer at Alliant Tech Systems Inc. in Clearwater, FL. He did much early work in the space program and has developed numerous analysis and real-time programs. He holds a PhD in physics from Auburn University. Crenshaw enjoys contact and can be reached via e-mail at jcrens@earthlink.net.

✓ Royalty Free
✓ Comprehensive Productline
✓ Focus on Service
All You NEED in an RTOS
www.atinucleus.com
Accelerated Technology INCORPORATED

SCOTT BRIGGS

Manage Your Embedded Project

As the embedded systems development effort shifts more toward software and schedules begin to slip, project management becomes key to increasing productivity.

Embedded systems are by far the most prevalent of all computer systems used in the world today. More than 99% of all microprocessors sold in 1998 were used in embedded systems. Advances in processor and memory technology and their associated low cost allow them to be considered for an enormous number of applications. This, coupled with the expectations of customers, has led to tremendous growth in the size and sophistication of embedded applications.

Embedded systems share the following characteristics. They generally are developed around custom hardware, require high quality and reliability, and frequently deal with real-time issues. The applications running on these systems are usually very large and complex. Examples of these systems include telecommunication switching systems, aircraft, spacecraft, weapons systems, and the multitude of applications in the car and home.

Several trends are causing the driving factors in embedded systems development to shift from hardware to software. First, processors continue to push the performance envelope, allowing more flexibility in the applications which run on them. Second, functions are being increasingly integrated on the silicon. Functions which in the past required several devices can now be found on a single device. Many development organizations have found that they can simply purchase whole boards from manufacturers, thereby eliminating expensive and very long hardware design cycles. On some projects, software development costs comprise about 90% of the total development costs.

Most software projects fail to meet their schedules. A 1998 survey indicates that only about 24% of all software projects succeeded, up from 9%

People represent the single largest factor in improving software productivity and quality. Issues include motivation, staffing, and teamwork.

in 1994.[1] A 1997 survey indicated that 80% of all embedded software projects fail to come in on time.[2]

In this article, I will present a number of techniques designed to improve the success rate of embedded projects. Three primary factors determine the success of a project: schedule, cost, and the quality of the delivered features. The ability of a company to optimize its project's schedule, cost, and quality objectives will give it a significant competitive advantage within its industry.

The development of software is one of the most complex activities ever undertaken. Despite significant advances in the methods and technologies for software development, it remains a people-oriented activity. As such, techniques that leverage the people and the way they develop software are the main factor in determining the success of a project.

I will focus on four areas of impact: people, process, product, and technology.[3] There are leverage points associated with each of these factors that are used to manage the risks to the success of a project.

People

People represent the single largest factor in improving software productivity and quality. Issues include motivation, staffing, and teamwork. Steve McConnell summarizes the research concerning variations in productivity of individuals and teams:[4]

- Greater than 10-to-1 differences in productivity among individuals with different depths and breadths of experience
- 10-to-1 differences in productivity among individuals with the same levels of experience
- 5-to-1 differences in productivity among groups with different levels of experience
- 2.5-to-1 differences in productivity among groups with similar levels of experience

Motivation

Most productivity studies have found that motivation is a stronger influence of productivity than any other contributing factor.[5] Hence, motivation can be your greatest ally on the road to a successful project.

We must now distinguish between motivation and movement as defined by Herzberg.[6] Motivation is a function of growth from getting intrinsic rewards out of interesting work. Movement is a function of fear of punishment or failure to get intrinsic rewards. Movement is the typical procedure used in animal training and its counterpart, behavioral modification techniques for humans. Examples of this abound in software companies as they strive to bring products to market in a competitive environment as quickly as possible. The behaviors that result from motivation and movement appear the same but have different long-term differences. Movement requires constant reinforcement and is focused on short-term results. Once the extrinsic reward disappears, so will the movement. Motivation, on the other hand, is based around personal growth being the ultimate reward. The benefit is long-term with no need of incremental extrinsic rewards.

Herzberg goes on to define the differences in motivating factors and hygiene factors.[7] Motivating factors stimulate growth and performance. Examples of these include achievement, recognition, the work itself, responsibility, advancement, and growth. Hygiene factors represent the basic conditions a person needs to do his or her job. These include company policy and administration, supervision, interpersonal relationships, working conditions, salary, and security. The conclusion to draw here is that the motivating factors are primary causes of satisfaction, and hygiene factors, at best, create no dissatisfaction. At worst, hygiene factors can create extreme dissatisfaction.

We can look at the motivating factors for software developers and their managers and come to some conclusions. While the data in Table 1 is a bit dated, it still offers significant insights. It should be clear that developers are

TABLE 1 Comparison of motivating factors

	General population	Software developers	Software managers
1	Achievement	Achievement	Responsibility
2	Recognition	Growth	Achievement
3	Work itself	Work itself	Work itself
4	Responsibility	Personal life	Recognition
5	Advancement	Technical supervision	Growth
6	Salary	Advancement	Relationship with subordinates
7	Growth	Relationship with peers	Relationship with peers
8	Relationship with supervisor	Recognition	Advancement
9	Relationship with peers	Salary	Salary
10	Relationship with subordinates	Responsibility	Relationship with supervisor

Adapted from Boehm, *Software Engineering Economics* (1981) and Herzberg, "One more time: how do you motivate employees?" *Harvard Business Review* (1987).

motivated by a desire to be challenged and prefer an environment where they can grow in their positions. This represents an opportunity for the organization to allow the developers to stay current with trends in technology and methods. Note that developers are motivated by the work itself. An environment that allows them to focus on the work free of interruption is critical. This also suggests that a constant redirection of work will eventually cause problems between developers and managers. Developers are also motivated by their personal lives. Don't expect them to live their lives at work. Last, the data clearly shows that developers are usually not motivated to move into a management position. More often than not, promoting a developer to manager is a mistake and the organization will suffer.

Staffing

Since software is developed by people, their selection for projects is of the utmost importance. Boehm presents five basic principles of software staffing.[8] There is the principle of top talent. The idea here is to use better and fewer people. Given the productivity implications above, significant differences exist between good, average, and mediocre people. The best people will be much more productive. Therefore, fewer people will be needed. However, the most productive people will also cost more. It seems that management will consistently pick the person who costs less, but this ignores the data concerning productivity.

The second principle deals with job matching. The idea is to fit the tasks to the skills and motivation of the people. In terms of technical skills, larger projects have many areas of specialization. It's important to select people to fill these positions based on their areas of expertise. For instance, it would not be appropriate to place a person skilled in developing higher level application software in a position which would require the development of embedded real-time firmware. The interpersonal skills of the people are also very important. Clearly, a person who is in a lead or managerial position needs to have effective communication and conflict resolution skills.

The third principle deals with career progression. An organization will benefit itself in the long run by helping its people to progress in their careers and keep current with the latest trends and technologies. It is also important not to force people to consistently work in areas where they are needed the most or have the most experience, but to let them explore other areas of interest. Of course, current business conditions will influence this activity significantly.

The fourth principle deals with team balance. The idea is to select people who will complement and harmonize with each other. The balance and coverage of a team are important considerations. Balance means that the team has a sufficiently diverse set of skills to achieve objectives. Coverage means that all critical skills needed are represented on the team. It is far better to have a team of average but solid performers dedicated to achieving results than it is to have one staffed with superstar developers engaged in adversarial relationships. The first team will easily outperform the second.

The fifth principle deals with phaseout: keeping a misfit on the team does not benefit anyone. Sometimes it becomes evident that a team member is not performing up to standards or is preventing everyone else from being productive. Perhaps the person is better as an individual contributor or would be better off leaving the team. The correct decision is to immediately deal with the problem and remove the person if necessary. In reality, this is more difficult than it seems. However, if the situation is allowed to languish, all team members will be affected, resulting in more problems than before.

Teamwork

As software continues to become more complex, teams are increasingly the mechanism that determines the success of a project. Only teams can deliver the productivity enhancements needed to create complex software in a timely and cost-effective fashion. While much work has been accomplished in enhancing individual productivity over the last two decades, it is the productivity of teams that will be the deciding factor in the future.

Katzenbach and Smith define a team as "a small number of people with complementary skills who are committed to a common purpose, performance goals, and approach for which they hold themselves mutually accountable."[9]

Most teams usually consist of a small number of people, often between five and seven.[10] Teams with less than five members risk not having the creativity needed, while those with more than seven can cause subgroups to form which increases the likelihood of conflict. Larger groups of people can form teams, provided each individual makes a significant commitment.

An effective team must have the right mix of skills. The first of these is technical or functional expertise. It is important that the team not be staffed by people of only one particular skill set. Problem solving and decision skills are also important so that team members can readily evaluate opportunities and problems and come to conclusions on how to proceed. Last, interpersonal skills are essential to group communication and resolving conflicts in constructive manners. A surprising conclusion here is that the team members do not need to have all of these skills at the beginning. The most effective teams quickly identify the gaps in skills and the specific training members need to acquire them. Individual accountability will promote the necessary learning.

A team's purpose and performance goals cannot be separated. A demanding performance challenge is an

Pentium® Debug: $4,995

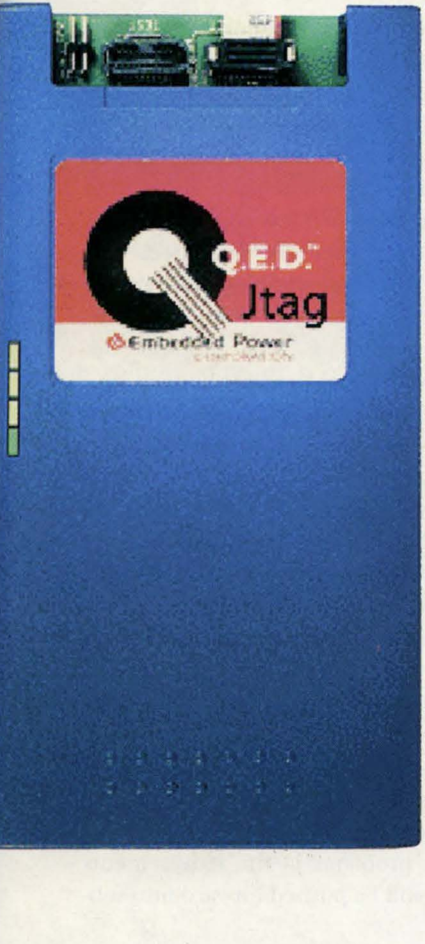
EmProve
Virtual Application Development and Prototyping (VAD-P™) environment

RTXC™
Royalty-free real-time OS
- TCP-IP stack
- File system
- Graphic system

BeaconSuite™
- Compilers
- Linkers & Locators
- Debuggers
- Simulators

Q.E.D. ™ Jtag

Sometimes the shortest distance between two points covers new ground.



Embedded Power's Q.E.D. tradition continues!

We bring you accurate, dependable hardware-assisted source-level debug on the Windows® NT/95/98 platforms.

Using our own VisualProbe™ or other debuggers, Q.E.D. Jtag provides:

- Kernel-aware debug with our RTXC RTOS
- Debug for all Pentium family members
- Full visibility of all CPU registers
- Support for your favorite toolchains, including CAD-UL, Borland, MetaWare® and Microsoft®
- Fast, efficient host communications. High throughput means fast debug functions and quick code downloads - even to FLASH.
- Requires NO fumbling with PC bus cards! Standard connection to your PC is via an IEEE 1284/EPP parallel port.
- Eliminates the need to CAD a JTAG port to your board. We offer a complete set of JTAG breakout adapters that make hardware connection a snap!

Stay on course through Pentium-class embedded development with Q.E.D. Jtag Debugger!

Pentium Processors	JTAG Port Pinout	Breakout Board
Pentium I	20	PGA 320
Pentium II	30	PGA 370/SLOT I
Mobile Pentium II	30	BGA/Micro BGA Intel breakout board
Pentium II Xeon	30	PGA 370/SLOT I
Celeron	30	PGA 370/SLOT I

www.embeddedpower.com

Embedded Power
CORPORATION

U.S. 281-561-9990
email: info@embeddedpower.com
Europe: +44 (0) 1256 474448
email: eurosales@embeddedpower.com

essential factor in team formation and must relate to the team's purpose or there will be confusion. Most teams get their mandate from management and it belongs to them both individually and collectively. An effective team is constantly shaping their purpose as this clarifies the implications to members. The purpose is translated into specific objectives, which keeps the focus on achieving results. The focus on results helps to push member status and personality issues away as well. Objectives also help the team to win the small battles that are essential to building commitment.

All team members must be committed to a common approach, a shared way of working together to accomplish their purpose. In order for this to happen, every team member must do equal amounts of real work and make substantial contributions. Members should agree on how this takes place; this helps to advance the team's performance objectives. There is also a need for social roles on a team, roles that provide support for members and keep everyone honest and headed in the right direction. These roles usually evolve of their own accord on effective teams.

For a team to exist, there must be mutual accountability. This means both individual and team accountability. Commitment and trust among team members helps accountability to flourish. Accountability tends to grow as the team's purpose, goals, and approach are developed. In many ways, the quality of the team's results are highly dependent on this joint accountability.

In discussing teams, it is important to look at various groups of people to determine how they fit into the team model. These small groups include working groups, pseudo-teams, potential teams, real teams, and high-performance teams:[11]

- The *working group* is a group of people who are simply sharing information among themselves. There is no significant incremental performance need or opportunity that requires it to become a team
- The *pseudo-team* is a group for which a significant incremental performance need or opportunity could exist. However, the group has not focused on collective performance and is not trying to achieve it. Pseudo-teams have a smaller performance impact than working groups because their interactions detract from each member's individual performance without the corresponding joint benefits
- The *potential team* is a group for which there is a significant incremental performance need. The group is also making a substantial effort to improve its performance impact. Usually, the purpose, goals, and approach need more clarity. Mutual accountability does not exist yet
- The *team* is a group of people who satisfy all the basic characteristics described above. A significant performance impact occurs during the movement from a potential team to a team
- The *high-performance team* is a group which satisfies all the basic team characteristics with members who are also deeply committed to one another's personal growth and success. The high-performance team outperforms all other teams

Katzenbach and Smith also identify a number of common approaches to enhance the performance impact of teams:[12]

- Establish urgency and direction
- Select members based on skills and skill potential, not personalities
- Pay particular attention to first meetings and actions
- Set some clear rules of behavior
- Set and seize upon a few immediate performance-oriented tasks and goals
- Challenge the group regularly with fresh facts and information
- Spend lots of time together
- Exploit the power of positive feedback, recognition, and reward

Process

Process is defined as a systematic series of actions for doing software development. The purpose of a process is to maximize the allocation of resources to productive activities and minimize the impact of overhead activities.[13] The following benefits are obtained by having a well-defined and flexible process in place. First, rework is avoided completely or minimized where possible. Second, there is a focus on the activities that constitute software development: requirements management, analysis, design, implementation, and test. Third, the quality of the delivered product is enhanced since errors are detected at the stage where they occur, where they are the least costly to fix. Last, there is an emphasis placed on risk management early in the lifecycle.

The waterfall model is the most common life cycle model found in software development. This model describes the sequential steps necessary to develop a successful software product. The model also shows that defects are much less costly to correct the earlier they are identified. The waterfall model works well with small projects and projects where the problem domain is well understood. However, several assumptions in the model cause significant problems when the model is used to develop software on large projects. The model assumes that the problem space is completely understood and that all requirements have been identified. However, the requirements can never be nailed down completely since customer expectations, markets, and the software and hardware technologies constantly change. Second, the model assumes that a design can be developed which captures all requirements and constraints. Obviously, any unresolved problems in the requirements phase will be pushed forward into sub-

Kiss your ASIC good-bye.

Xilinx introduces the new Spartan™ -II family of FPGAs. With all the speed, density and programmability you could wish for in a low-cost logic solution, it'll make you think twice about using an ASIC ever again.

Up to 150,000 gates at 200Mhz-believe it!

Say hello to a new level of performance. The Spartan-II family delivers up to 150,000 system gates at speeds of 200Mhz and beyond, giving you design flexibility that's hard to beat. And these low-powered, 2.5-volt devices feature I/Os that operate at up to 3.3 volts with full 5-volt tolerance.

Spartan-II also features multiple Delay Lock Loops, on-chip RAM (block and distributed), and the versatility of Select I/O™ technology supporting over 16 high-performance interface standards.

All this in a chip that offers unlimited programmability, and can even be upgraded in the field.

The unbeatable low-risk, high-volume solution

You asked for it, you got it. The Spartan-II series is the result of more than fifteen years of FPGA design, and invaluable feedback from thousands of customers just like you.

Designed to eliminate the initial cost, long development cycles and inherent risk associated with conventional ASIC devices, Spartan-II FPGAs are the winning solution in the high-volume arena. That includes digital modems, set-top boxes, wireless communications, and consumer electronics.

Priced to go, go, go

When you consider the upfront expense, the risk of high design costs, plus the minimum volume requirements associated with the typical ASIC, you'll see that when you choose a Spartan-II solution for your next high-volume application, price is no longer an issue.

When we said we'd deliver 100,000 gates for under ten dollars, we meant it. That means you can get going faster, and more affordably than ever.

Why would you choose anything else?

The Spartan-II family is a groundbreaking advancement in semiconductor technology, offering up to 150,000 system gates, high performance, and more features per dollar than you ever thought possible in an FPGA. The value is simply unbeatable, and the benefits make your choice all the easier.

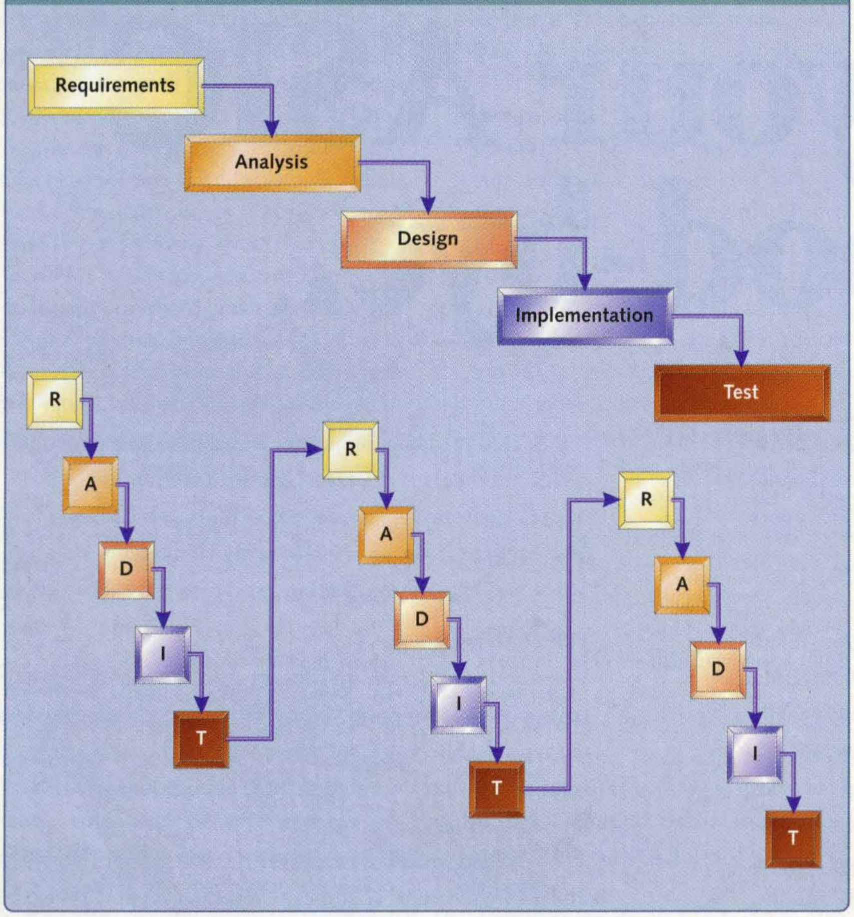
Find out more about Spartan-II FPGAs today. Visit our website, and we'll prove to you that it's time to kiss *your* ASIC good-bye.

SPARTAN™-II
FPGA

www.xilinx.com

XILINX®
The Programmable Logic Company℠

FIGURE 1 Waterfall and iterative models



sequent phases. Last, the model pushes risks forward, effectively masking them until it is too late to deal with them in any meaningful fashion. These risks usually manifest themselves at integration time.

Since the waterfall model works well on small projects, we can take a large project and divide the development activities into a series of small waterfall projects as Figure 1 shows. The figure compares the waterfall model with an iterative process showing three iterations. Notice that there are three distinct and smaller integration phases instead of a "big bang" integration. Each iteration resolves risks and brings us closer to a successful product.

The iterative process described here will follow the work now known as the Rational Unified Process (RUP).[14] This process identifies two distinct sets of activities in software development: research and development activities and production activities. It is from the separation of these two sets of activities that the phases of the iterative process are defined. The first stage is defined as the engineering stage and consists of the inception and elaboration phases. Team sizes during the engineering stage will be smaller due to the unprecedentedness of the work involved. The second stage is defined as the production phase and consists of the construction and transition phases. Team sizes during the production stage will be larger since construction and transition activities are well defined and much work can take place in parallel. Keep in mind that each phase may consist of several iterations. During the inception phase, the business case and scope of the project are defined. This is where the initial ideas are investigated and the functionality in the final product is identified. An initial architecture is defined based on the required functionality. The milestone associated with this phase is called the life cycle objective milestone.

During the elaboration phase, the goal is to refine the requirements and architecture to a stable point where the architecture can be baselined. A plan is also developed that identifies the iterations and evaluation criteria for each iteration during the construction phase. Since this is the phase in which risks are sufficiently mitigated, the plan should enable you to predict the schedule and cost of the remaining development. An executable prototype is also generated in this phase. At the end of this phase, the engineering activities are considered complete. The milestone associated with this phase is the life cycle architecture milestone.

The construction phase begins the activities associated with production. During this phase, all remaining components and features are developed and integrated into the product and then tested. The emphasis here is on managing resources and optimizing schedule, cost, and quality. If a project is big enough, the activities in this phase are executed in parallel. The goal of this phase is to have a beta release of the software to users to evaluate provided this does not expose the project to high risks. The milestone associated with this phase is the initial operational capability milestone.

The transition phase is where the software product is mature enough to be deployed to end users. This means that the product has met its quality objectives and user documentation exists. As end users exercise the software, issues will arise that require corrections or new releases. The goal of this phase is to have the end users selfsupportive in their use of the product. The milestone associated with this

Some things are instinctive. (Now including logic analyzers.)

Only* $3,200

Logic*Wave*

- Debug designs with 8- and 16- bit microcontrollers
- 100 MHz state analysis and 250 MHz timing analysis
- 128 K memory per channel
- 34 channels of logic analysis

Doing what comes naturally. That's the idea behind Logic*Wave*, Agilent Technologies' new logic analyzer. It may be the easiest logic analyzer you've ever used. How? For starters, it's PC-based, for familiar, simple operation. So you won't need to waste hours learning—and relearning—the interface. You can draw the trigger easily by selecting a trigger condition from a drop-down menu. And its low cost means you can actually afford one of your own. Because a logic analyzer isn't very helpful if you have to share it with ten other engineers.

ELECTRONIC PRODUCTS 1999 PRODUCT OF THE YEAR AWARD

TEST & MEASUREMENT WORLD BEST IN TEST 2000 AWARD WINNER

Experience Logic*Wave* for yourself. Visit our web site for a FREE download of the actual interface software. Or call us to get a FREE demo on CD-ROM. And talk to another engineer who'll answer your questions about Logic*Wave's* ease of use. Although, after you've used it once, you probably won't have any.

www.agilent.com/find/bi

1-800-452-4844, Ext. 6918

Agilent Technologies
Innovating the HP Way

*U.S. list price.

FIGURE 2 Iterative process life cycle

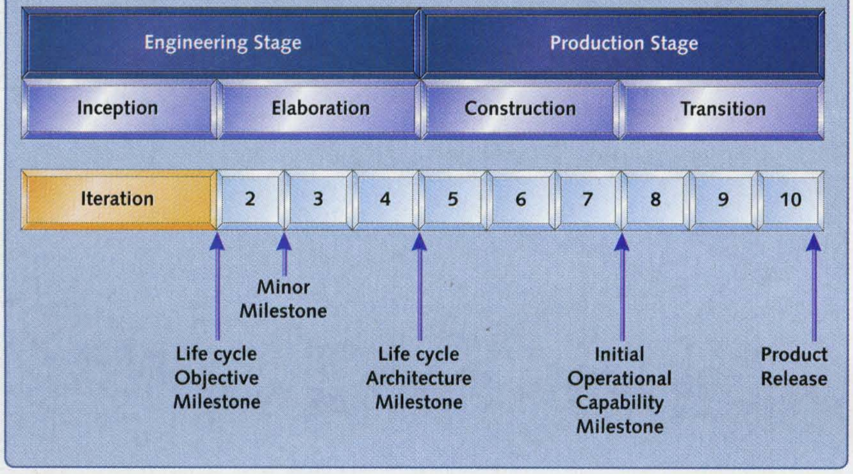


TABLE 2 COCOMO effort and duration estimates

Product Size (KLOC)	Effort (staff months)	Time (months)
0.1	0.2	1.4
1.0	2.8	3.5
10	44	8.0
20	102	11
50	306	16
100	703	20
200	1,616	27
500	4,852	38
1,000	11,147	49

phase is the product release milestone. Figure 2 shows an example of an iterative process life cycle with 10 iterations.[15] The number of iterations can vary. The major and minor milestones are shown.

Notice how the iterative process addresses the problems of the waterfall model. The more difficult features are developed first in the earlier iterations and refined in later iterations. Risks are effectively mitigated since there are now several integration periods to identify and resolve problems. Requirement and feature changes and additions can now be handled as the project proceeds through subsequent iterations. The iterations allow the developers to learn as they go and build up their expertise as they proceed. This will be especially effective with newer tools and development environments. As the iterations proceed, opportunities to identify commonality across the project will arise, an activity that should facilitate the reuse of components. Since the project will have been tested and integrated several times, the probability of significant errors remaining is low. This means that the overall quality of the product should be enhanced.

Architecture plays a critical and central role within RUP. The following are some of the definitions given to architecture:[16]

- The architecture is the organization of a software system
- The architecture consists of structural elements and their behaviors and interfaces
- Architecture consists of the composition of these elements into progressively larger subsystems
- Architecture also considers performance, reliability, reuse, and economic and technological constraints and trade-offs

In short, architecture covers nearly every aspect of software system design including hardware constraints. In the past, architecture has not been given its proper place as the foundation on which a software product is based. Many of the problems commonly attributed to the waterfall model are in reality problems with the architecture not being defined appropriately. For example, many system-level issues such as performance requirements do not surface as problems until the integration phase of a project. At this point, it is too late to do a reasonable job of satisfying these requirements within any reasonable timeframe. A sound architecture and an iterative process that continually mitigates risks are two techniques that decrease the probability of problems like this.

One of the big problems with architecture has been representing multiple views or dimensions. To accurately represent an architecture, multiple views covering different aspects of the system are needed. The concept of multiple architecture views is part of RUP and was first elaborated by Philippe Kruchten.[17] The five views are now summarized:

- The *logical view* addresses the functional requirements of the system, the feature content that the end user manipulates. It represents an abstraction of the design model and identifies major subsystems and classes
- The *development view* describes the organization and management of source code, data, components, and executables in the development environment. It seeks to address the issues of ease of development, reuse, and build-or-buy decisions
- The *process view* addresses the runtime behavior of the system in terms of its tasks, threads, and processes. It deals with issues such as performance, scaleability, throughput, and fault tolerance
- The *physical view* addresses how the executables and other runtime

Don´t gamble with success ...

Hitex deals you winning cards!

USB Agent - USB Analyzer

The USB Agent is a full-featured USB-Protocol analyzing instrument ...
A development tool that captures, analyzes and intelligently displays all USB-signals. Right performance - Right price!

In-Circuit Emulators for ...

MX430 family:

For MSP430 microcontrollers, these emulator systems support all members of this TI family of low-power microprocessors, including the ultra-low-power MSP430x11x derivatives. These instruments are proving to be very popular.

C166 family: DProbe167 and DBox167

A modular family of powerful in-circuit emulator systems for the Siemens C16x variants featuring the most advanced adaptation technology: PressOn. Our C161 system with a 64 K trace buffer module is available for less than $3,500. Also supports ST-10 microcontrollers.

68HC12 family: DProbeHC12

For 68HC12 processors; supports maximum speed and all operating voltages; plug and play (no jumpers, no switches); additional BDM interface cable; flash programming built-in; PlugOn trace with 32k frames. Move up to super advanced features with DBoxHC12.

8051 family: MX51/AX51

True real-time emulation for more than 500 variants from all manufacturers. Including derivatives like Atmel 89S8252, Intel 8x931x and Siemens C505C.

and more:

251, 68k, CPU-32, 80x86, 68HC11, Pentium®Processor

hitex

DEVELOPMENT TOOLS

Hitex USA
710 Lakeway Dr., Suite 280
Sunnyvale, California 94086

	Tel.:	(800) 45-HITEX
	Tel.:	(408) 733-7080
	Fax:	(408) 733-6320
	E-mail	info@hitex.com
Hitex Germany	Tel.:	(0721) 9628-133
	Fax:	(0721) 9628-149
Hitex UK	Tel.:	(01203) 69 20 66
	Fax:	(01203) 69 21 31
Hitex Asia	Tel.:	(65) 74 52 551
	Fax:	(65) 74 54 662

www.hitex.com

"The best emulator I have ever used!"

Follow this URL to a special promotion:
http://www.hitex.com/hitools/deal.htm

Benefit from our world-wide support:

Austria Tel: +43-1-259727/20 Belgium Tel: +31-77-3078438 PR China Tel: +86-10-62383376 Czech Republic Tel: +420-2-66316661 Denmark Tel: +45-43-424742 France Tel: +33-1-39611414 India Tel: +91-22-8520817 Israel Tel: +972-3-7657666 Italy Tel: +39-0434-633500 Malaysia Tel: +603-7167591 Netherlands Tel: +31-77-3078438 Pakistan Tel: +92-51-822075 Poland Tel: +48-22-7556983 Russia Tel: +7-812-3252792 Singapore/Indonesia Tel: +65-7496168 South Korea Tel: +82-2-645-0386 Spain/Portugal Tel: +34-91-6401234 Sweden/Norway/Finland Tel: +46-87-404580 Switzerland Tel: +41-1-3086666 Taiwan Tel: +886-2-26983456 Turkey Tel: +90-212-4472700 PBX components map to the underlying computing platforms. This view deals with the systems engineering aspect of architecture

- The last is the *use-case view.* This view contains several key use-cases which are used to drive the discovery process with regard to the other views. As such, it integrates the other views into a coherent architecture which drives the remaining work. The use-cases also serve to validate the other views later

Product

The most significant point to understand about software development is this: the complexity of software does not scale linearly with size. Complexity is defined as product size. The effort required to build a software product increases faster than the size of the product. This means that we can decrease the time it takes to develop the product by reducing its size. We can quantify these relationships by considering the Constructive Cost Model (COCOMO).[18] COCOMO is a

FIGURE 3 Relationship between productivity and product size

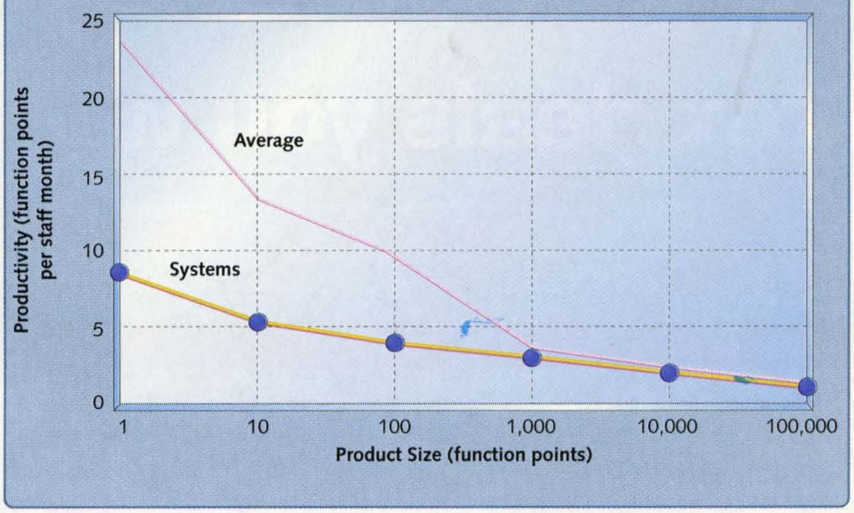
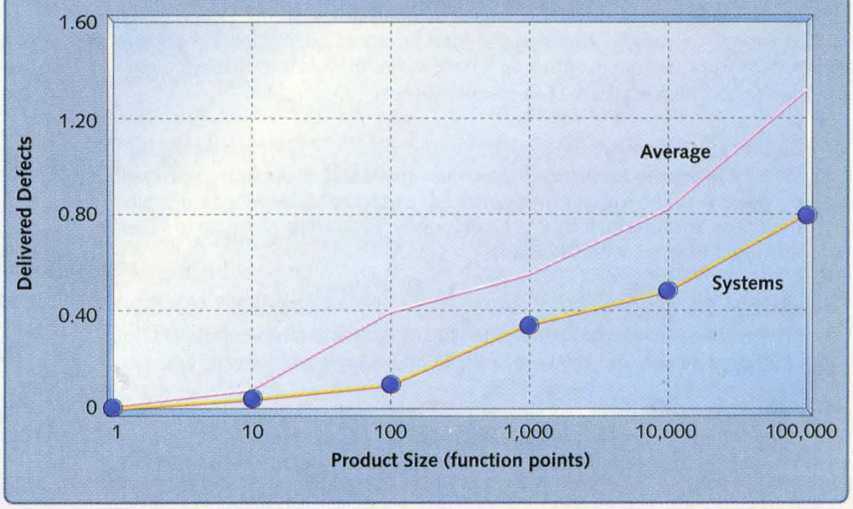


FIGURE 4 Relationship between product size and delivered defects



NEW
PC-IN-THE-LOOP
PROTOTYPING
PRODUCTS

Don't just predict your real-time system's performance. Prove it.

New PC-in-the-Loop products enable real-time rapid prototyping of embedded designs on standard x86 PC hardware in any form factor.

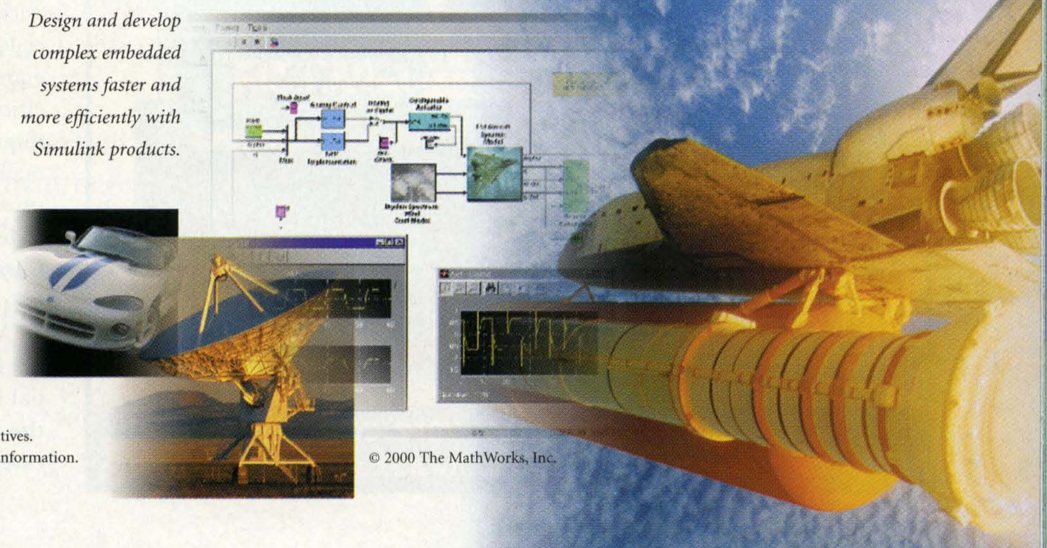
With the new PC-in-the-Loop™ products, you can accurately test the performance of real-time embedded system designs on a low-cost PC long before your final target hardware is available. Simulink code generation products convert block diagrams to real-time C code for rapid prototyping. You can then go directly to your embedded target with our production-quality code.

Learn how PC-in-the-Loop products enable rapid prototyping right on your own desktop.

To get your Embedded Systems Technical Kit, for pricing information, or to buy online now, visit www.mathworks.com/espc.

SIMULINK®

Design and develop complex embedded systems faster and more efficiently with Simulink products.



© 2000 The MathWorks, Inc.

The MATH WORKS Inc.

Visit www.mathworks.com/espc
or call 508-647-7000

We have a worldwide network of international representatives.
Visit our Web site at www.mathworks.com/eur for more information.

model that estimates the effort and duration of a software project based on its estimated size in thousands of lines of code (KLOC). Table 2 shows data derived from the COCOMO embedded mode equations. Notice how the effort increases much faster than the product size.

We can also look at the impact of product size on such aspects as schedules, productivity, and delivered defects. Table 3 shows that as product size increases, an increasing number of projects fail to meet their schedule commitments or are simply cancelled. Note especially the significant jump in late and cancelled projects as the product size increases from 100 to 1,000 function points. Our ability to manage the complexity of the product begins to fail within this range.

Figure 3 shows how productivity and product size relate within the U.S.[20] In general, as product size increases, productivity drops. This is due to the additional overhead that larger projects have and is primarily due to the growth of interpersonal communications and system integration.[21] Note that the productivity of embedded systems is lower than the U.S. average.

TABLE 3 Relationship between product size and schedule[19]

Product size (function pts)	Early Projects	On-time Projects	Late Projects	Cancelled Projects
1	6.00%	90.00%	3.00%	1.00%
10	8.00%	85.00%	5.00%	2.00%
100	10.00%	78.00%	8.00%	4.00%
1,000	5.00%	68.00%	17.00%	10.00%
10,000	3.00%	40.00%	32.00%	25.00%
100,000	1.00%	35.00%	34.00%	30.00%

Figure 4 shows the relationship between product size and delivered defects.[22] Notice how the number of defects increase with product size. Notice also how the number of defects is lower for embedded systems, reflecting the increased focus on quality.

We see that productivity decreases and defects increase as product size increases. Notice that the effects are nonlinear. The conclusion to draw from this is that smaller, less complex projects deliver higher quality software and do it more quickly.

Given the above implications about product size, an obvious way to simplify the product is to write less software. This can be accomplished in several ways. First, object-oriented techniques and higher order languages like C++ can be used to write less code via abstraction and inheritance. Second, commercial components and libraries can be used instead of developing the same functionality in-house. Of course, maintaining and upgrading commercial components and libraries involves work.

A significant decision for embedded systems is whether you need a real-time operating system (RTOS). The first decision is to determine if your application needs the services an RTOS can provide. If it does, the next decision is whether you buy a commercial RTOS or write your own. A proprietary RTOS can better address the specific needs of an application but requires significant cost to develop and maintain. A commercial RTOS will usually be cheaper, more thoroughly tested, and include customer support. It will also likely provide an integrated set of tools for development.

Another way to simplify the product is to develop only the most essential features. A rule of thumb such as the 80/20 rule can be followed. An example of this is to develop the 80%

RESUME

HIRED

+ Interview ✓
* References ✓
+ Qualifications ✓
+ Call Tomorrow

Name: HI-TECH C

Address: 7830 Ellis Rd., Suite 105, Melbourne, FL 32904

Phone: 800 735 5715

Fax: 407 722 2902

Email: hitech@htsoft.com

Web site: http://www.htsoft.com/

DOB: Jan 1st, 1984

Salary: Peanuts. Really. But a small signing bonus would be nice.

Skills: Highly experienced in the translation of C programs into machine code for a wide variety of microntrollers. Can produce working code up to ten times faster than a human assembler programmer. Skilled in debugging complex programs.

Education: George Institute of Technology, Cornell University, University of California, State University of NY, University of East London, MIT, Sydney University, University of Waterloo, Worcester Polytechnic, Ecole Polytechnique, Fachhochschule Hannover and many others too numerous to mention here.

Employment: Currently employed at Microchip Inc, Cisco Systems, NASA, Federal Reserve Board, British Railways, Intel, Siemens, Motorola, Philips, Garmin, Hitachi and way too many more to list here. I like to spread myself around.

Languages: C, assembler.

Architectures: 8051, PIC, 68000, 68HC11, 6805, V8, Z80, 8086, H8/300.

Personal: Very easy-going, undemanding but highly supportive of co-workers. Not a self-starter, I prefer to work in a team, small or large. I'm tireless and will work 24 hours a day if required, even skipping lunch breaks.

Interests: Wow, where do I start - I've been involved in rocketry, space, entertainment, defence, railroads, motor vehicles, domestic appliances, toys, you name it!

Goals: I want to be on your desktop. I want to make your application sing. I want to bang your bits, clear your RAM and crunch your calculations. Just give me a chance - I promise you won't regret it!!

of the features that take 20% of the time.[23] An even better way is to develop the feature set in iterations. The more difficult features are developed first and given more iterations to be refined. This process allows the risk inherent in the more difficult features to be mitigated across the iterations.

Technology

A variety of technologies is available to enhance software development. Object-oriented techniques and higher order languages such as C++ first come to mind. There has not been much use of either one within the embedded world due to performance and memory constraints. However, as the cost of processor cycles and memory continue to decrease, they should be looked at in detail. The long-term benefits of abstraction and reuse of designs and code are simply too great to ignore. It must also be noted that significant learning curves must be overcome in the use of both. It is also worthwhile to note that C++ takes many fewer lines of code to express equivalent functionality in C or assembly.

TABLE 4 COCOMO 2.0 scale factors

Scale factor	Definition
Precedentedness of project	The degree of domain experience of the development organization
Development flexibility	The degree of relaxation or conformity in the process
Architecture risk resolution	The degree that the technical feasibility is demonstrated before entering into the production phase of the life cycle
Team cohesion	The degree of cooperation among stakeholders
Process maturity	The process maturity level as defined by the Software Engineering Institute

The concept of modeling is becoming more important in software development. Models are used because they help us to understand the problem domain we have to deal with. Once we have a good understanding of the problem domain, we can make tradeoffs and come up with an optimized solution to the problem. These models are maintained by the organization and represent its intellectual property.

Modeling has been given a significant push forward with the introduction of the Unified Modeling Language (UML). UML is a rigorous graphical language that allows the user to visualize, specify, and document the artifacts of a software system. It stands a very good chance of becoming the communication medium of choice for architects, designers, and programmers. UML allows one to effectively specify the architecture views in the 4+1 view model discussed earlier.

It is important to note that none of the tools and techniques described above offer productivity enhancements that would cause them to be viewed as a silver bullet. However, their use in a consistent and disciplined manner will go far in allowing organizations to enhance their productivity and competitiveness.

COCOMO 2.0 software cost model

The COCOMO 2.0 software cost estimation model is an update to the earlier COCOMO model and takes into account modern software practices.[23] It will be useful to look at the factors that this model takes into account. COCOMO 2.0 consists of three different cost estimation models that can be mapped to an iterative life cycle. The applications composition model corresponds to prototyping efforts to resolve high-risk issues. This model would be used

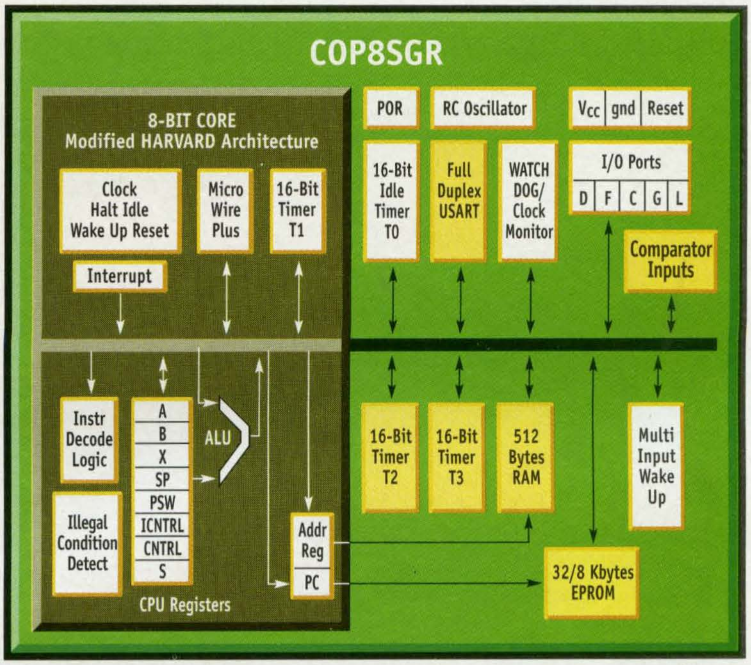

COP8SGR
8-BIT CORE
Modified HARVARD Architecture
Clock Halt Idle Wake Up Reset
Micro Wire Plus
16-Bit Timer T1
Interrupt
Instr Decode Logic
Illegal Condition Detect
A
B
X
SP
PSW
ICNTRL
CNTRL
S
ALU
Addr Reg
PC
CPU Registers
POR
RC Oscillator
Vcc
gnd
Reset
16-Bit Idle Timer T0
Full Duplex USART
WATCH DOG/ Clock Monitor
I/O Ports
D
F
C
G
L
Comparator Inputs
16-Bit Timer T2
16-Bit Timer T3
512 Bytes RAM
Multi Input Wake Up
32/8 Kbytes EPROM

TABLE 5 Early design/post-architecture cost drivers

Early design cost driver	Combined post-architecture cost driver
Product complexity	Required reliability, database size, product complexity, documentation
Required reuse	Required reuse
Platform difficulty	Execution time constraint, main storage constraint, platform volatility
Personnel experience	Analyst experience, programmer experience, language/tool experience
Personnel capability	Analyst capability, programmer capability, personnel continuity
Facilities	Use of software tools, multisite development
Schedule	Required development schedule

TABLE 6 Judging the impact of techniques

Technology	Learning Curve (months)	Initial Productivity Results, %	Final Productivity Results, %	Defect Removal Results, %	Schedule Reduction Results$%	Return on Investment
Reusable code	6.0	-10.0	25.0	10.0	-20.0	>20%
Reusable design	6.0	-10.0	25.0	10.0	-17.5	>20%
Quality estimating	1.0	2.5	7.5	20.0	-7.5	>20%
Cost estimating	1.0	5.0	12.5	5.0	-12.5	>15%
Formal inspections	1.0	10.0	20.0	35.0	-15.0	>20%
Measurements	2.0	-2.5	12.5	20.0	-12.5	>20%
Prototypes	1.0	12.5	18.0	10.0	-10.0	>20%
Risk management	2.0	-1.0	5.0	5.0	-5.0	>15%
Project management	1.0	-2.5	5.0	0.0	-2.5	>15%
Configuration control	2.0	2.5	7.5	5.0	-2.0	>10%
Integrated tools	6.0	-12.5	15.0	10.0	-7.5	>15%
OO programming	6.0	-15.0	25.0	10.0	-7.5	>15%
OO design	6.0	-20.0	12.5	10.0	5.0	>5%
SEI CMM Level 5	18.0	-5.0	20.0	17.5	-10.0	>20%
SEI CMM Level 4	12.0	-5.0	15.0	12.5	-5.0	>15%
SEI CMM Level 3	6.0	-5.0	10.0	10.0	-5.0	>10%
SEI CMM Level 2	3.0	-5.0	10.0	10.0	-5.0	>10%
SEI CMM Level 1	2.0	-2.5	0.0	0.0	0.0	0%

to estimate work that involved exploratory analyses typical at the beginning of new and complex projects. This model maps to the engineering stage (inception phase) of RUP.

The early design model corresponds to the exploration of alternative software and system architectures. This model would fit within the engineering stage of RUP (elaboration stage) where there would be a good understanding of the requirements and architecture.

The post-architecture model corresponds to the actual development and maintenance of a software product. This model would fit into the production stage of RUP (construction and transition phases). At this point, the requirements and architecture should be stable.

The overall cost estimate equations for the early design and post-architecture phases follow.[25] For the early design model:

$$\mathit{Effort} = 2.45\ E(\mathit{ARCH})\ (\mathit{Size})^P$$

For the post-architecture model:

$$\mathit{Effort} = 2.45\ E(\mathit{APP})\ (\mathit{Size})^P$$

where

$$P = 1.01 + 0.01\ W_i$$

Effort is given in staff months and Size in function points for the first equation and in thousands of source lines of code in the second equation. The exponent P is the scaling factor applied to both equations and is a weighted sum of five factors as given by Table 4.

$E(APP)$ is the product of the seven early design effort adjustment factors (cost drivers) shown in the left column of Table 5. Note that the early design factors are derived from the composite post-architecture cost drivers as shown in the right column of Table 5.

$E(ARCH)$ is the product of the 17 post-architecture effort adjustment factors (cost drivers) shown in the right column of Table 5.

The effects of the five scale factors are exponential in nature. As such, they represent your greatest leverage points on the road to a successful software project. They are also consistent with the themes discussed in this article.

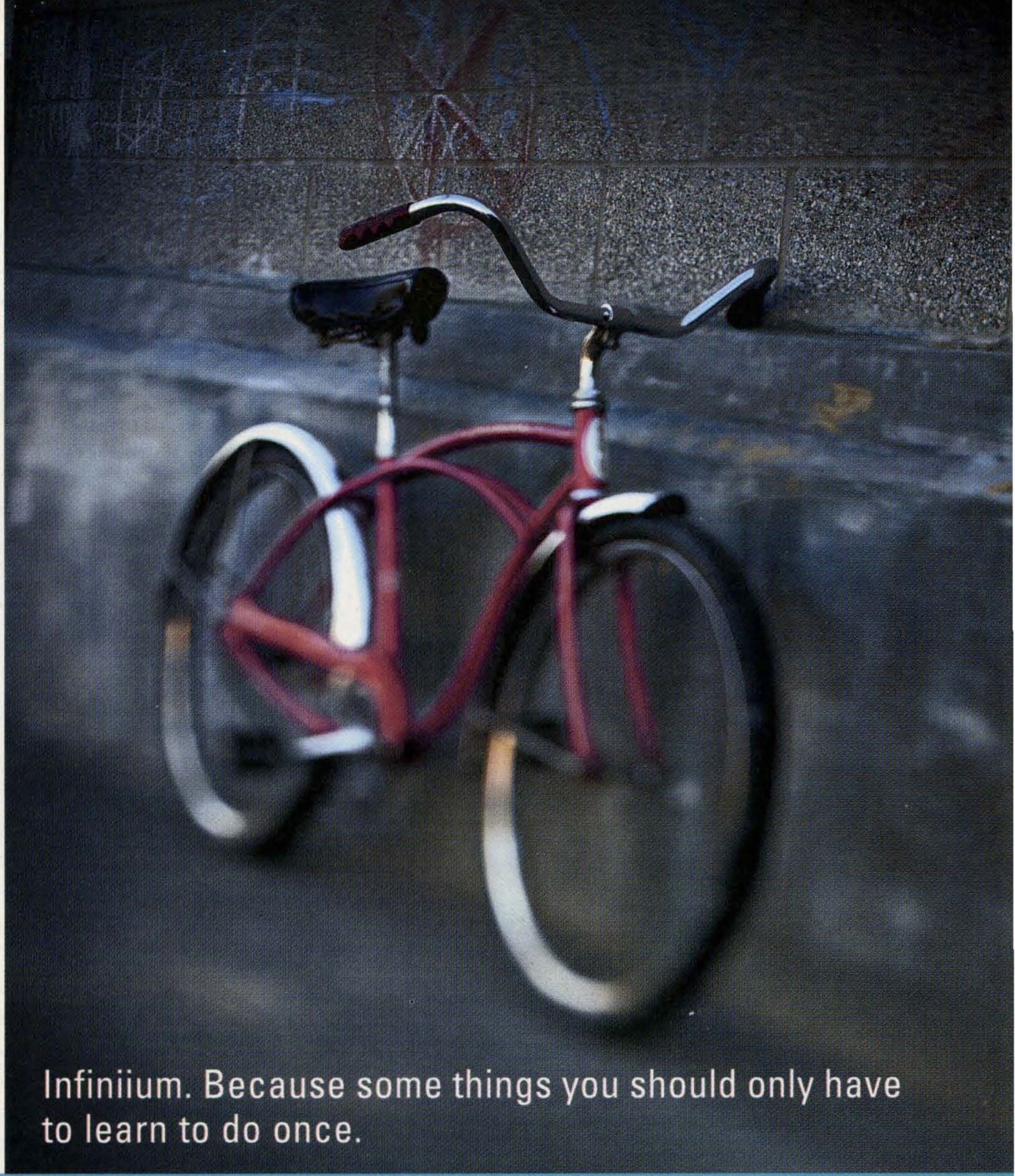
Technology impacts

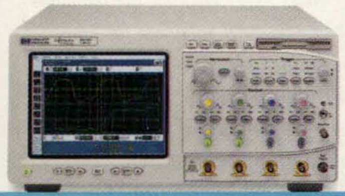
I have discussed a number of techniques designed to increase the probability of success of software projects. Table 6 lists some recent data that can be used to judge the impact of these techniques.[26] The data come from the knowledge base of Software Productivity Research. The margin of error in the data in Table 5 is high, but we can learn some things from the general trends present.

Notice how the vast majority of technologies listed require significant learning curves and productivity results that are initially negative. This is important information not usually taken into account by the vendors or users of the technologies. Many examples exist of where technologies are abandoned when faced with this initial up-front cost. However, the data clearly show the long-term benefits of these technologies.

We see that reusable designs and code have significant up-front costs replicating the research to date in this area. However, the end productivity results and ROI clearly justify the effort here as do the positive schedule impacts.

Formal inspections represent one of the best ways to enhance the quality of software products. There is little


Infiniium. Because some things you should only have to learn to do once.



from $9,995*

Agilent Infiniium Oscilloscopes

- 500 MHz, 1 GHz and 1.5 GHz models
- Up to 8 GSa/s sample rate
- 2- and 4- channel models
- Advanced triggering
- Communication mask testing
- Voice control (English only)

The idea is to debug your design, not to figure out how to use your scope. High performance doesn't have to mean difficult to fathom. Not with an Infiniium oscilloscope from Agilent Technologies.

All those soft keys and menu layers have been replaced with something more engineer friendly. There's an analog-like front panel that's straightforward and simple (the green knobs control the green waveform on the screen, for example). The Windows 98® interface handles advanced measurements with drag and drop ease. And LAN connectivity makes sharing your work easy.

If you need communication masks for compliance testing, Infiniium offers those, too. And should you have questions only another engineer can answer, you can call one on our toll-free number.

So even if you're a fan of someone else's scope, you might want to discover first-hand how good an Infiniium really is. A free, no-obligation demo makes that a lesson easily learned.

www.agilent.com/find/Infiniium1

1-800-452-4844,** Ext. 6757

Agilent Technologies
Innovating the HP Way

*U.S. list price.
**In Canada, call 1-877-894-4414, Ext. 6447. ©1999 Agilent Technologies ADGPP003/ESP
Windows 98 is a registered trademark of Microsoft Corporation.

upfront cost with tremendous ROI and final productivity and schedule reduction results. Notice how formal inspections can remove 35% of the defects at a particular stage. This means that a combination of design and code inspections can immediately give you a defect removal rate of 70%. Measurements are also a good way to enhance quality but only if the measurements are used to tune the existing processes.

The data on prototypes show why this activity is an important part of any process. An early prototype gives leverage with both the schedule and the quality of the product as well as being a highly productive activity. Notice how there is little up-front cost here since this work will have to be done regardless of when it is accomplished.

The data on object-oriented (OO) techniques are interesting. Both OO and ROI have extremely high up-front costs but longer-term OO programming shows much better productivity and ROI. It seems that the learning curve for OO design is prohibitive, causing many projects to either abandon it or go back to other better-known techniques.[27]

Last, we see the increasing productivity, quality enhancements, and ROI that adherence to the Software Engineering Institute's Capability Maturity Model (CMM) brings. **esp**

Scott Briggs is currently a manager of software development in the switching systems group at Alcatel USA. He holds a BSEE from Washington University and an MBA. He has 10 years experience in the development of hardware and software for embedded systems in the defense and telcommunications industries.

Resources

Boehm, Barry W., "An Overview of the COCOMO 2.0 Software Cost Model," Software Technology Conference, April 1995. This paper can be found at *sunset.usc.edu/COCOMOII/cocomo.html*.

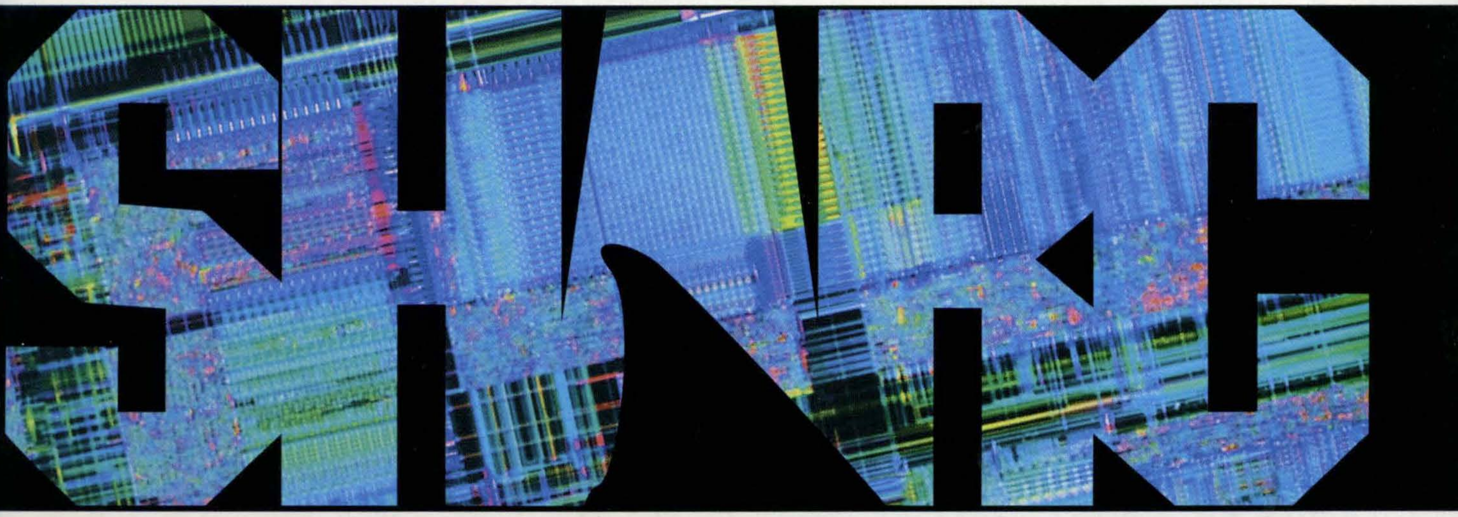
Boehm, Barry W., Bradford Clark, Ellis Horowitz, Ray Madachay, Richard Shelby, and Chris Westland, "Cost Models for Future Software Life Cycle Processes: COCOMO 2.0," *Annals of Software Engineering*, 1995. This paper can be found at *sunset.usc.edu/COCOMOII/cocomo.html*.

Boehm, Barry W. *Software Engineering Economics*. Englewood Cliffs, NJ: Prentice Hall, 1981.

Herzberg, Frederick, "One more time: How do you motivate employees?" *Harvard Business Review*, September-October 1987, pp. 109-120.

Jones, Capers. *Applied Software Measurement*. New York: McGraw-Hill, 1996.

Katzenbach, Jon R. and Douglas K. Smith.


SHARC

The Wisdom of Teams. New York: HarperCollins, 1993.

Kruchten, Philippe, "The 4+1 View Model of Architecture," *IEEE Software*, November 1995, pp. 42-50.

Kruchten, Philippe. *The Rational Unified Process: An Introduction*. Reading, MA: Addison-Wesley, 1999.

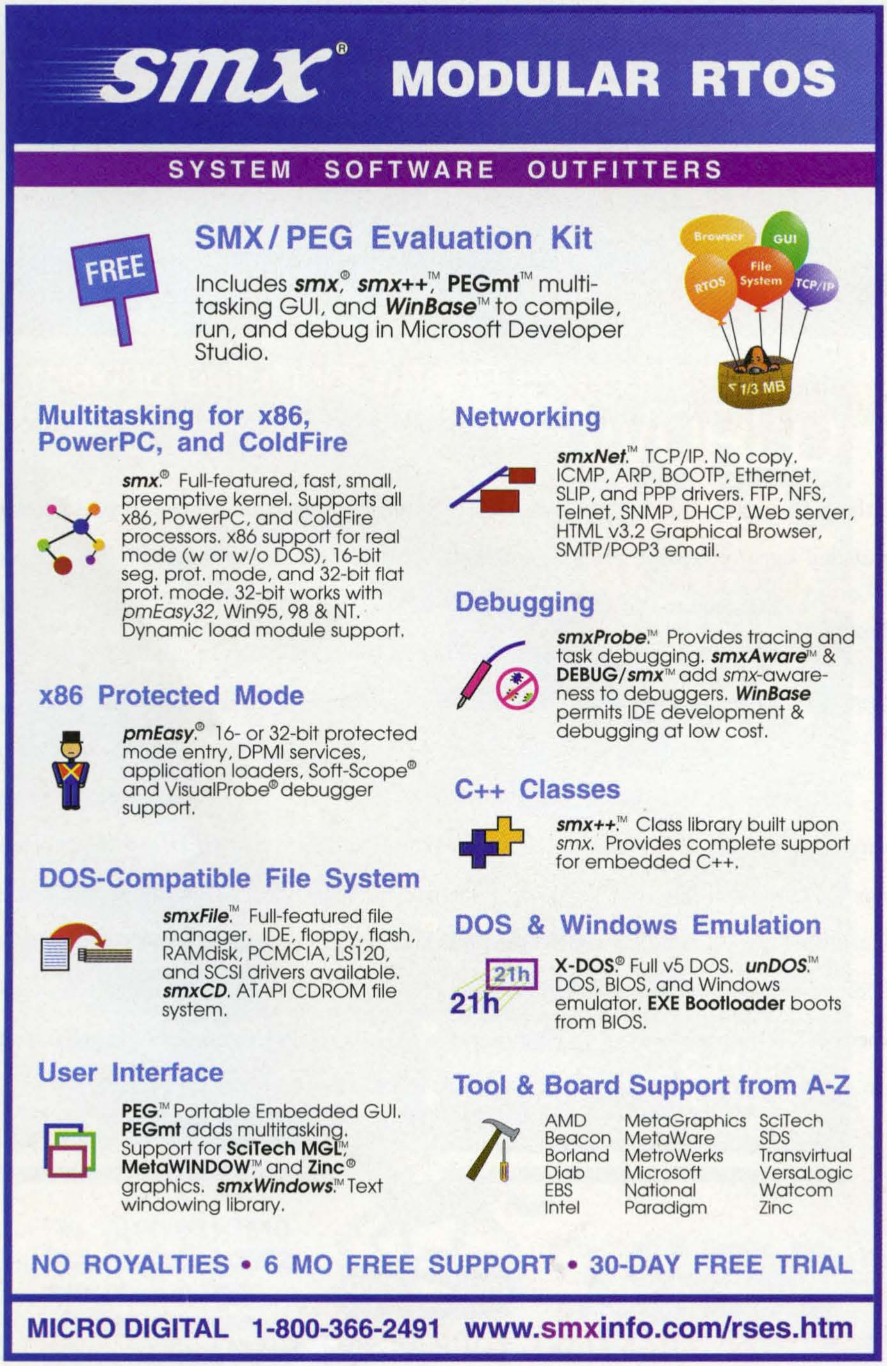
McConnell, Steve. *Rapid Development*. Redmond, WA: Microsoft Press, 1996.

Royce, Walker. *Software Project Management: A Unified Framework*. Reading, MA: Addison-Wesley, 1998.

Rymer, John R., "Optimizing Software Teamwork," Whitepaper for Rational Software Corp., April 1999. This paper can be found at *www.rational.com*.

References

1. From "Optimizing Software Teamwork" by John R. Rymer.
2. From "Embedded Y2K" by Jack G. Ganssle (*Embedded Systems Programming*, February 1999, p. 97)
3. I first read about this way of breaking up the areas of impact into the four dimensions of people, process, product, and technology in Steve McConnell's book *Rapid Development*.
4. McConnell, p. 14.
5. Boehm, *Software Engineering Economics*, p. 672.
6. Herzberg, "One more time: How do you motivate employees?" p. 118.
7. Herzberg, pp. 111-113.
8. Boehm, pp. 667-672.
9. Katzenbach and Smith, *The Wisdom of Team*, p. 45.
10. I first heard of this range for team size in an introductory management class in 1999.
11. Katzenbach and Smith, pp. 90-92.
12. Katzenbach and Smith, pp. 119-127.
13. Royce, Software Project Management, p. 41.
14. Kruchten, *The Rational Unified Process: An Introduction*.
15. Adapted from Kruchten, Figures 4 through 6.
16. Kruchten, pp. 80-81.
17. Kruchten, "The 4+1 View Model Architecture."
18. See Boehm's *Software Engineering Economics*.
19. From private communication with Capers Jones.
20. Data adapted from Jones' *Applied Software Measurement*, p. 213.
21. Boehm, "An Overview of the COCOMO 2.0 Software Cost Model."
22. Data adapted from Jones' *Applied Software Measurement*, p. 232.
23. McConnell, p. 17.
24. Boehm et al., "Cost Models for Future Software Life Cycle Processes: COCOMO 2.0."
25. Royce, pp. 274-281
26. Adapted from Jones' *Applied Software Measurement*, pp. 240-241.
27. Jones, p. 237.



8051

68HC11

Z80

Z180

68HC12

Mcore

683xx

PowerPC

68K

x86

68HC05

ColdFire

And
Many
More

The Fastest BMD Ever...

Background Mode Debugger
PowerPC, 683xx, 68HC12, ColdFire

The speeds of this ultra-fast BMD will astound you. Full source-level debugging features are now at your fingertips at a fraction of the cost of in-circuit emulators. If you're looking for more debugging power, such as trace and performance analysis, we also offer the HMI ProTools line of in-circuit emulators for a wide range of processors. Call today.

HMI
HUNTSVILLE MICROSYSTEMS, INC.
Power
Stop
Run
Background Mode Debugger Plus USB

...Just Got Faster

- High-Speed USB Connection
- Supports Entire Device Families
- Supports Flash Programming
- Extremely Affordable
- Source-Level Debugger Included
- 128 Software Breakpoints
- Supports Most Major Compilers
- Ideal for Production Environments
- Manipulate Registers
- Single-Step Source or Assembly
- Connect Multiple BMDs to One PC

Avocet: The Complete Solution

- Over 20 Years of Experience with Embedded Development Tools
- Compilers, Assemblers, Simulators, RTOS, In-Circuit Emulators for Hundreds of Processors
- Expert Technical Support Staff
- We will get you any tool you need

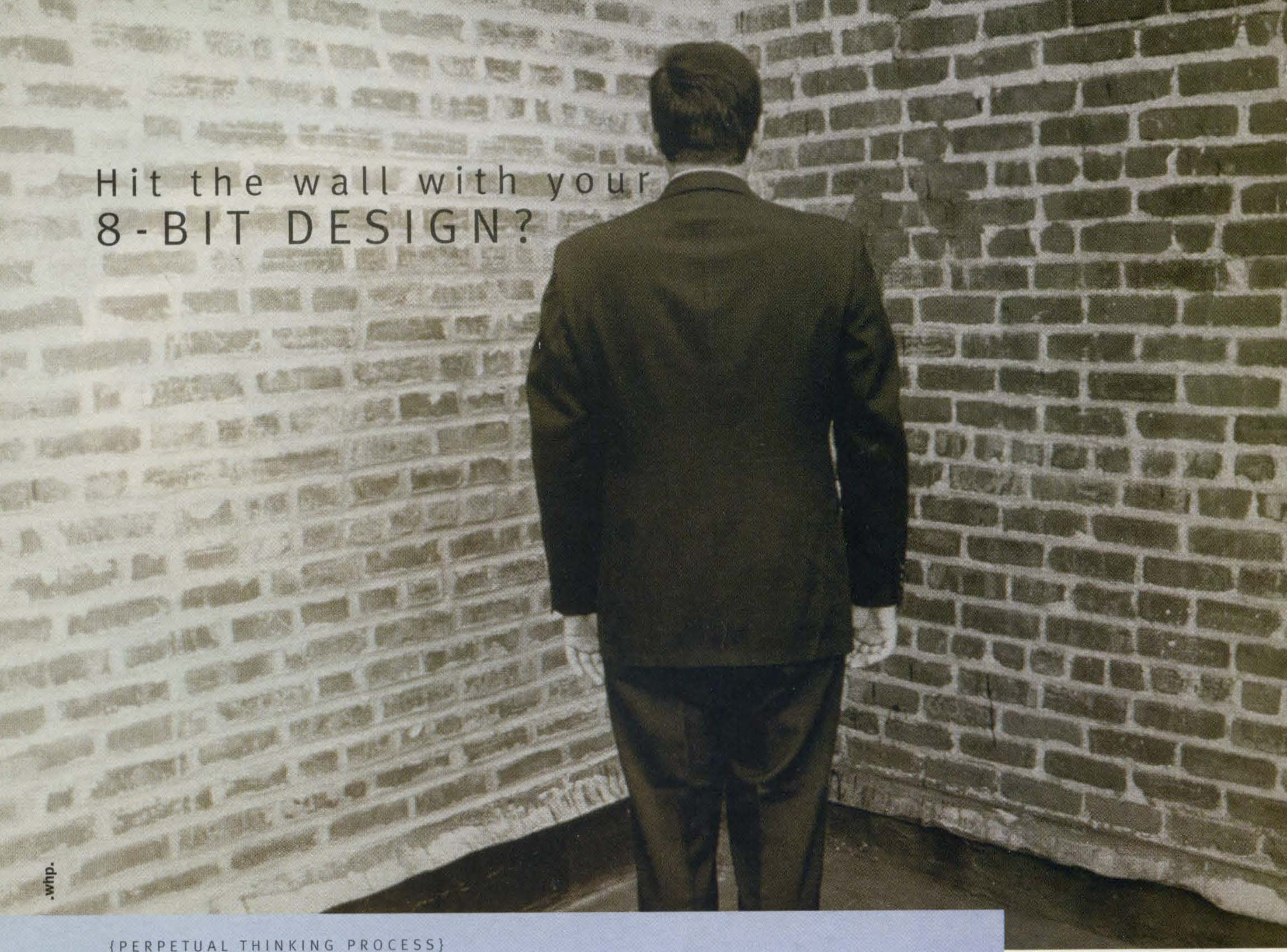
(800) 448-8500

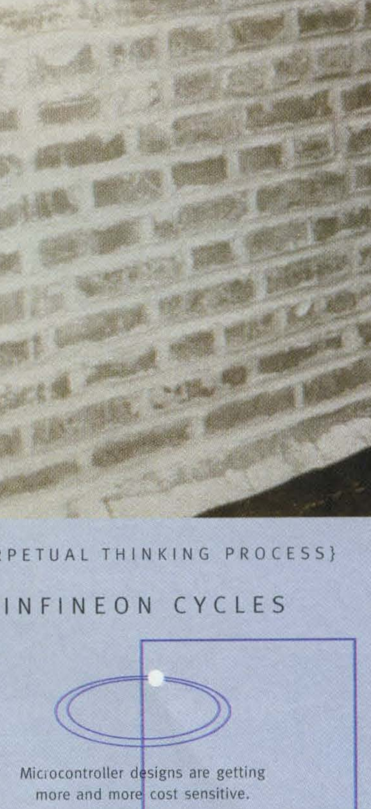
www.avocetsystems.com

sales@avocetsystems.com

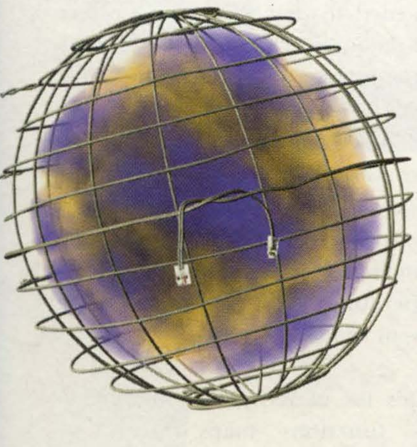
Support for Hundreds of Processors.
Hundreds of Products.
One Phone Call.
Avocet.

AVOCET SYSTEMS, INC


Hit the wall with your
8-BIT DESIGN?
.whp.


Microcontroller designs are getting more and more cost sensitive.
Microcontroller designs require more and more performance.
Engineers are faced with tighter and tighter timelines during design cycles.
Innovative and comprehensive solutions are the answer.
Move up to Infineon.

Michael Barr



TCP/IP or Not TCP/IP?

iternet Appliance Design

in this issue

The question is not that simple, unfortunately. TCP/IP is the name given to a large collection of related networking protocols. It would be unusual, particularly in an embedded application, to use all of these protocols at once. So the real question is: what subset of TCP/IP will you include in your product, if any?

It's now been more than five years since I developed my very first internet appliance. Of course, it wasn't really an appliance in the consumer understanding of that word. But it was the first time that anyone at my company had attempted to use the TCP/IP suite of protocols to connect proprietary equipment.

The product in question was a piece of telecommunications equipment designed to connect remote land-based telephone networks via satellite. One or more of these satellite gateways would be attached to each land-based network, thereby allowing telephone calls from one network to the other to be routed through a satellite channel.

None of the TCP/IP protocols was required to communicate with the satellite, nor were any of them required to communicate with the land-based telephone network. Therefore, we really could've used just about any protocol we wanted for communication between the internal subsystems of our product. We chose a subset of TCP/IP.

Why TCP/IP?

With literally hundreds of internally and externally developed communications protocols to choose from, we had no compelling reason to choose TCP/IP. This was 1995 after all, and the Internet was just beginning to take hold outside of universities and the Department of Defense; the World Wide Web was still in its infancy; and a war between Ethernet and Token Ring was raging fiercely. If I'd been a stock market investor back then, I probably would've bet all my savings on a networking company called Novell. (Thankfully, I didn't have any money to invest at the time.)

What we needed on that project was a way to connect one or more of these satellite gateways to a local PC-based monitoring and control station. (Take a look at Figure 1 to get a feel for the overall context.) Most PCs were not on networks then and Windows 3.1, the dominant PC operating system, had only limited TCP/IP support—in the form of the Winsock add-on. But this was still far better than the PC support for proprietary network protocols. So we decided to use TCP/IP for the communication between the subsystems of our own product (specifically, the satellite gateway and the PC).

The point was not to put our product onto the Internet (with a capital 'I') or even the customer's corporate intranet. In fact, the Ethernet cabling connecting our satellite gateways and PC-based monitoring station was going to be restricted to that purpose. The customer wouldn't necessarily even know what protocols were involved. So it really didn't matter what protocols we chose, so long as they did the job and did it well.

By the end, we'll have built a substantial body of source code that you can easily port to any embedded platform processor.

Embedding TCP/IP

Back then, independent embedded TCP/IP stack vendors didn't exist like they do today. We were familiar with a couple of RTOSes that included TCP/IP stacks (yes, VxWorks, LynxOS, and others included TCP/IP that far back), but the RTOS we'd selected did not. Our RTOS was just a small kernel that didn't include any kind of networking support at all.

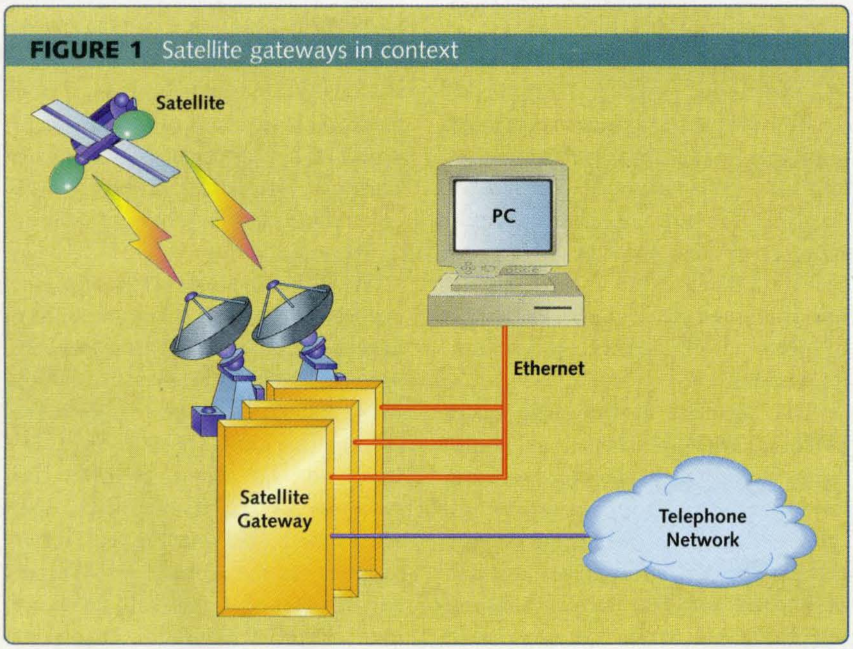


FIGURE 1 Satellite gateways in context

Our options were, therefore, quite limited. We could either port the BSD TCP/IP stack ("free" source code from the University of California at Berkeley that formed the basis of most TCP/IP implementations at the time) or write our own. A preliminary estimate of the work involved in porting BSD revealed that the source code (and resulting executable) was huge and would not be easily reduced to just the subset we needed or ported to our kernel.

We really only needed a narrow subset of the full TCP/IP suite, so we decided to develop our own more limited protocol stack from scratch. In fact, what we needed wasn't a TCP/IP stack at all. What we really needed for this particular application was a UDP/IP stack.

Figure 2 shows the interrelationships between the member protocols and applications in a typical TCP/IP stack. As you can see from the figure, UDP and TCP are "peer" protocols (ICMP and IGMP are also their peers), in the sense that they are at the same protocol level. The vast majority of application programs and higher-layer protocols depend on either UDP or TCP, but not both.

You see, communications between our satellite gateway and PC-based monitoring and control station was pretty simple. The list of communication requirements was limited to the following:

- Each satellite gateway needed to be individually addressable by the PC
- At boot-time, each satellite gateway needed to determine the address of the local PC and receive a set of boot parameters
- At boot-time, each satellite gateway needed to be able to download new firmware, if any, from the PC and store it into local flash memory
- During normal operation, each satellite gateway needed to send its operating statistics and various event reports to the PC

This list of required communications functions maps nicely to UDP/IP, ARP, BOOTP, and TFTP. In other words, all of these demands could be met by a UDP/IP-only protocol stack augmented with ARP support and BOOTP and TFTP application software. So that's what I set out to develop.

Road map

My plan for the next several months is to recreate as much of the development of this UDP/IP stack as possible in the pages of this column. I'll write new source code as we go and provide it to you. Along the way, I hope to make clear the what and why of UDP/IP and related protocols and their implementations. Of course, I'm also hoping to have some fun writing code. That's something I don't get to do much in this job and it's the one thing I miss most about being away from the day-to-day of embedded systems development.

By the end, we'll have built a substantial body of source code that you can easily port to any embedded platform processor. Mind you, the result of this will not be a complete TCP/IP implementation like those you can buy commercially. It probably wouldn't compete very well with a commercial UDP/IP stack either. Those guys have a lot of different customers to satisfy and the time to do things right and stay up to date on the latest protocol advancements. But it may be that my

THREADX

Being Royalty Free Is not Enough.

You work under a lot of pressure to meet production demands and schedules. So the RTOS you choose has to have more than one outstanding quality to make you feel comfortable.

ThreadX, the leading royalty-free, source-code RTOS gives you more, much more...Like state-of-the-art technology with preemption-threshold and a picokernel design that automatically scales to your needs. Processor support for most processor families. Documentation so clear and concise that ThreadX's User Guide constantly receives kudos from our customers. And such a wide array of tools that chances are there is ThreadX-aware debugging already built-into your favorite tool chain.

Call **888.THREADX** or visit **www.threadx.com** and find out more about ThreadX, the RTOS with all the features to make you comfortable.

the comfortable RTOS

Processors Supported

- Win32 NEW!
- ARC
- ARM
- StrongARM
- Thumb
- Power PC
- ColdFire/68K
- x86
- Hitachi SH
- MIPS
- NEC V8xx
- TinyJ
- SPARC
- M-Core
- TriCore
- SHARC
- TMS320C6x
- TMS320C54x

eL
Express Logic, Inc.

tf **888.THREADX** • *fx* **858.613.6646** • **www.expresslogic.com**

FIGURE 2 The interrelationship of the TCP/IP protocols and applications

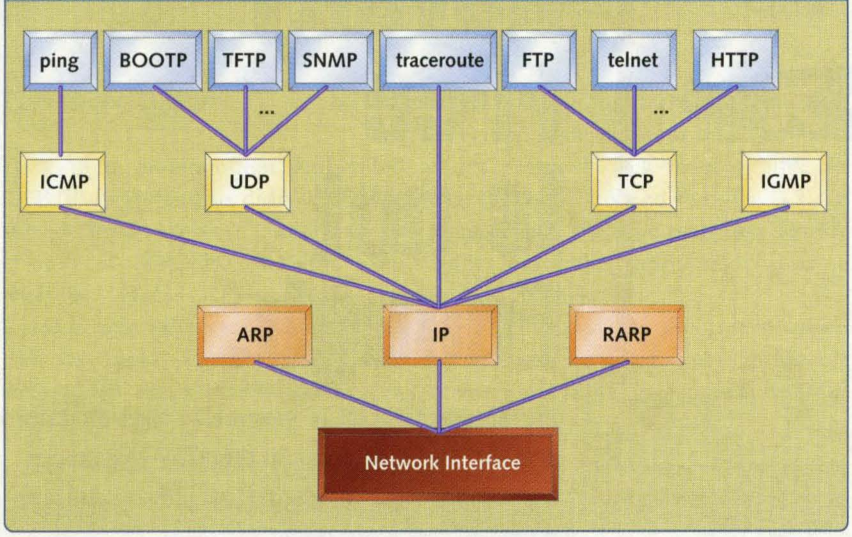
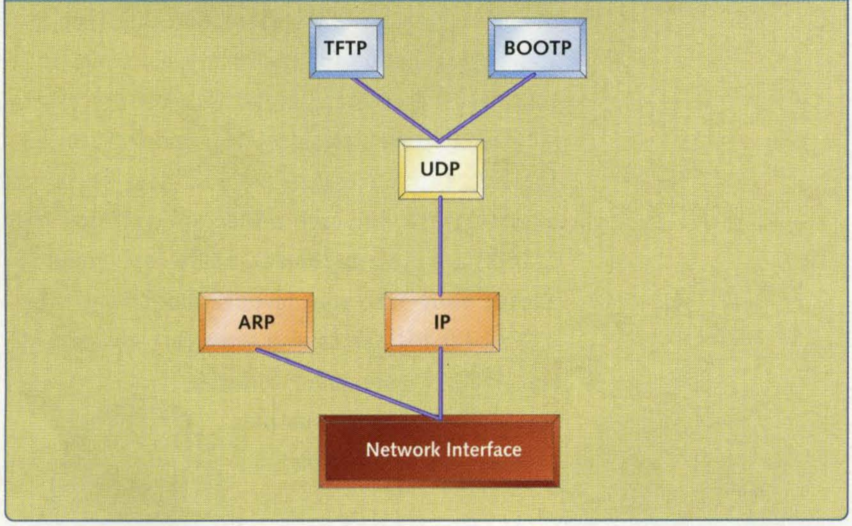


FIGURE 3 A smaller UDP/IP-only protocol stack



stripped down UDP/IP stack will be just enough for a real-world embedded application like yours or it may simply enable you to be a more intelligent consumer.

Figure 3 shows the relationship between the various protocols I will implement. The "network interface" at the bottom was originally Ethernet. I'll tell you what you need to know about Ethernet, but will eventually also show you how to use SLIP or PPP in its place. TFTP and BOOTP, at the top of the stack, are the application programs. Both are client applications that communicate with server counterparts hosted on a remote system.

This month's feature article by Christopher Leidigh will show you how BOOTP can be used to bootstrap a network-connected device. As you'll learn, BOOTP has been somewhat displaced in recent years by its modern descendent, DHCP. However, the two protocols are very similar.

Between the network interface and the client applications are the protocol layers, IP and UDP. Here, I'll concentrate only on those implementation details necessary to execute the BOOTP and TFTP client applications. In the UDP case, that won't be much different than a full implementation of the protocol. After all, UDP is a much simpler protocol to understand and use than its transport layer peer TCP.

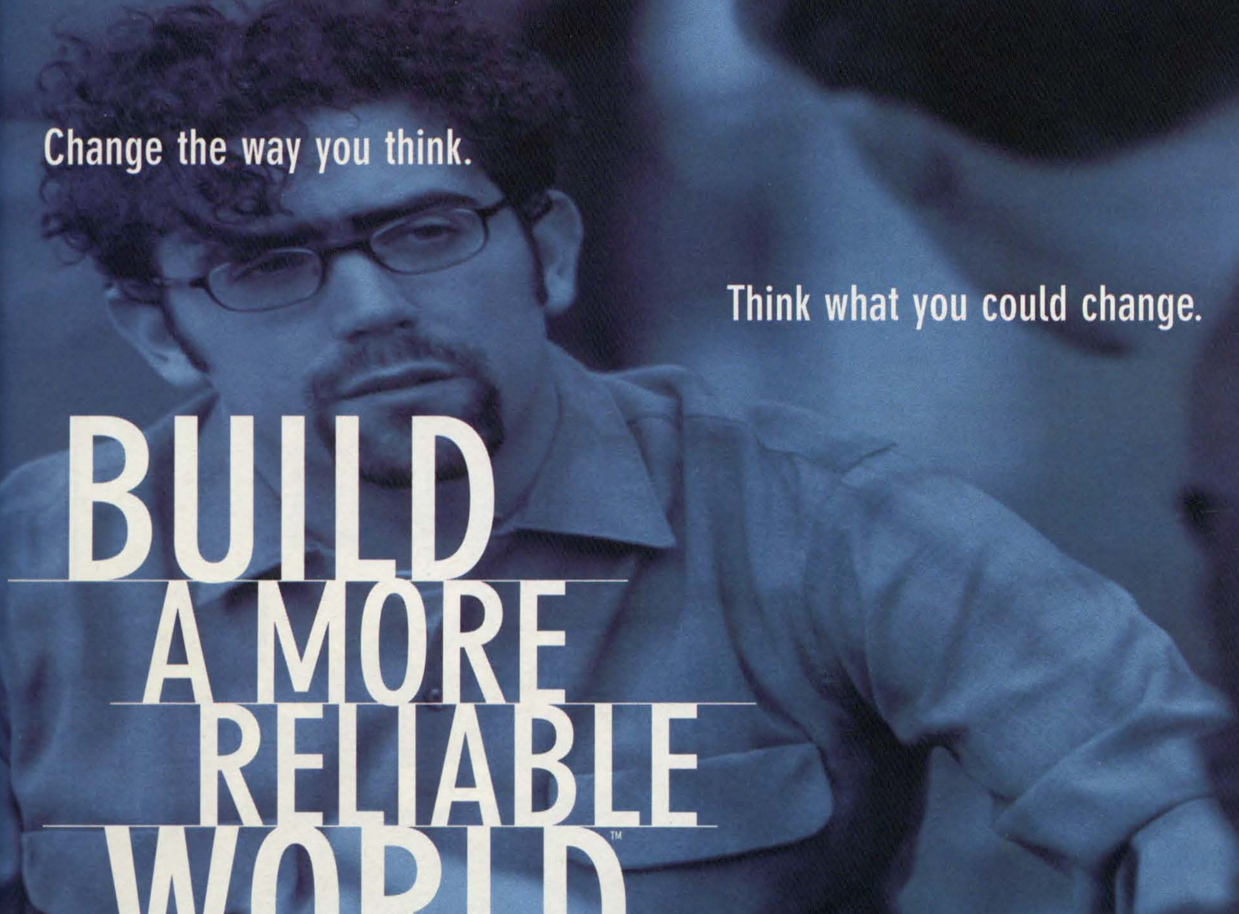
UDP provides only for the "connectionless" exchange of information between two devices on the network. Since there is no "connection" to track and maintain and there is no guarantee of successful communication, the UDP protocol is stateless.

The implementation of the IP protocol layer, on the other hand, can get complicated very quickly. The more features you add the more you need to research and understand the best implementation techniques invented. So I've really cut back on my list of supported features. The things my IP layer supports, it will do well. But it will not include every feature of a general-purpose IP implementation.

Even SNMP, a far more complicated client-server application than either BOOTP or TFTP, requires only a UDP/IP stack. Unlike HTTP, it is not built upon the "connection-oriented" communications embodied by TCP. I'll explain why as we get further into the subject. For now, you'll have to settle for an article by Kedron Wolcott that compares the remote management capaabilities of SNMP and HTTP, then shows you how to use web-based management techniques to leverage existing SNMP support in a product.

Next month I'll begin the UDP/IP stack development in earnest. In the meantime, stay connected... **esp**

Michael Barr is the technical editor of Embedded Systems Programming. *He holds BS and MS degrees in electrical engineering from the University of Maryland. Prior to joining the magazine, Michael spent half a decade developing embedded software and device drivers. He is also the author of the book* Programming Embedded Systems in C and C++ *(O'Reilly & Associates). Michael can be reached via e-mail at mbarr@cmp.com.*

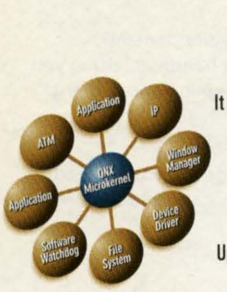


Change the way you think.

Think what you could change.

BUILD A MORE RELIABLE WORLD™

START WITH A MORE RELIABLE OS



It goes against conventional thinking. But to build more products in less time — without sacrificing reliability — you need more than good tools. You need a better OS.

An OS with an architecture so advanced it lets you pinpoint memory faults in minutes. Prototype without system rebuilds. Add features without retesting your entire software suite. Upgrade systems without rebooting. And respond faster to what customers want.

Now imagine. What if this same OS was backed by two decades' experience in the embedded market. An outstanding reputation for technical support. And an OS company as reliable as its technology. Think how that could change your world, and your customers' world.

For the better.

www.qnx.com

Call about our eval system:
800 676-0566 ext. 2323

QNX® RTOS

Build a more reliable world™

Register for QNX2000 - Our 10th International Technology Conference - Vancouver, Canada - May 14-17

OSE

– Conventional RTOS? ...sorry!

OSE™, THE NEW GENERATION REAL-TIME OPERATING SYSTEM, WILL NOT LET YOU DOWN!
OSE 1

OSE THE NEW GENERATION RTOS!

FASTER

Streamlined for extreme reliability and speed, OSE's kernel and TCP/IP stacks blow away the competition. From the RTOS to the tools, OSE sets tomorrow's standards for small size and high performance.

TOUGHER

As the only fault-tolerant RTOS, OSE supports mission-critical real-time systems, allowing complete non-stop recovery from hardware and software failures AND hot swaps – critical for high-availability functionality.

LONGER-LASTING

OSE is heterogeneous, scalable, and distributable, protecting your investment by allowing your application to grow from one CPU to hundreds.

MORE POWERFUL

OSE's kernel offers automatic supervision, dynamic reconfiguration and integrated error handling, letting you focus on your core competency: designing applications.

SIMPLER

OSE's powerful ultra-efficient message-based architecture lets you write nearly every bit of application code using only eight system calls.

SAFER

OSE is the world's only RTOS that is safety certified to the demanding specifications of IEC 61508 . OSE is also being certified according to the stringent DO 178-B.

PROVEN

Millions of products worldwide are already taking advantage of OSE , including the top brands in telecommunications and process control. OSE is the RTOS of the future.

www.enea.com

ENEA OSE SYSTEMS INC
5949 SHERRY LANE , SUITE 625, DALLAS TX 75225. PHONE: 214-346-9339. FAX: 214-346-9344. EMAIL: info@enea.com

KEDRON WOLCOTT

Exposing MIB Data to a Web-based Interface

Web-based management is essential these days. With the right architecture and an SNMP MIB inheritance library, legacy management code can remain untouched, allowing you to realize tremendous design and product life cycle benefits.

Although the Simple Network Management Protocol (SNMP) and Web-based management (WBM) are vastly different management paradigms, an SNMP MIB inheritance library, along with a backplane-based architecture, allows them to be combined in such a way that the advantages native to each methodology can be fully exploited. Furthermore, if a product is designed thoughtfully, using a backplane, SNMP code can be leveraged across multiple management paradigms for comprehensive access and control capabilities. In addition to WBM, for example, management via Java applications/applets, JavaScript, XML, and command line interfaces (CLI) can be quickly and easily achieved. Lastly, and most importantly, with a backplane-based architecture in place, no new development effort for a given device is required. Developers can fully leverage hundreds, and potentially thousands, of hours of pre-existing work. This expedites product time-to-market and provides the ability to safely migrate to next-generation network management protocols.

To show how a backplane-based architecture along with an SNMP inheritance library can provide these benefits, I will first explain the general principles behind both SNMP and WBM, and then explain how those principles can be seamlessly integrated to provide a robust, extensible Web-based management solution using previously defined SNMP MIB variables.

SNMP operation

SNMP is a well-established protocol framework that has been in published form since 1990. It has been widely adopted in the datacom industry, and close to

SNMP is a well-established protocol framework that has been in published form since 1990.

FIGURE 1 Management via SNMP

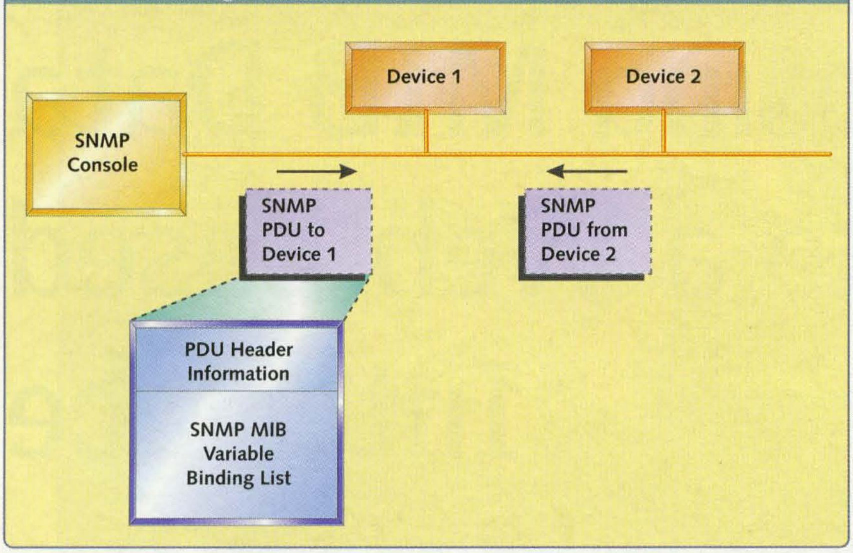
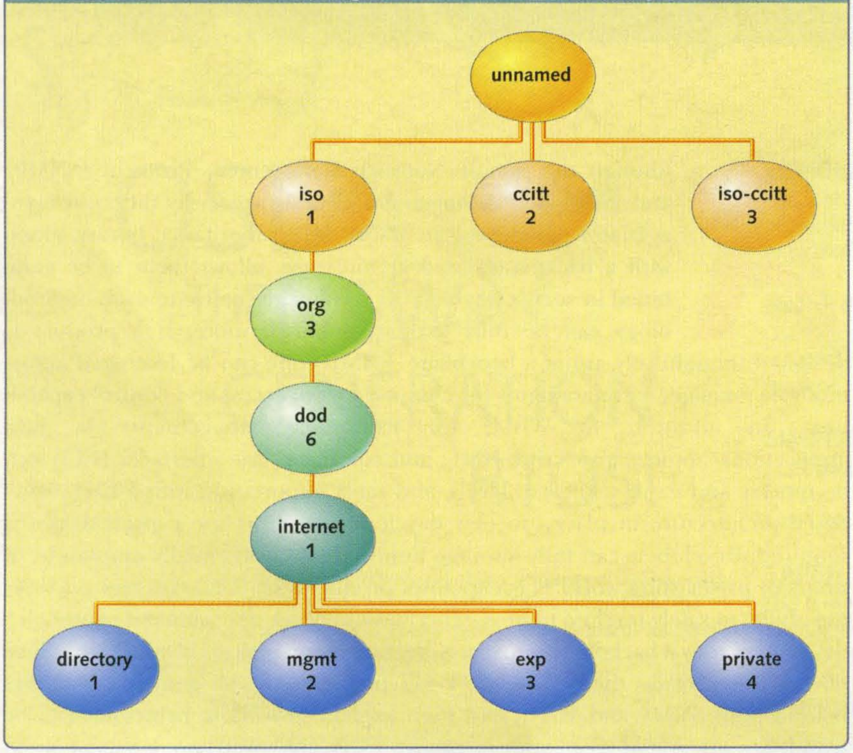


FIGURE 2 Structure of management information hierarchy



100% of shipping network equipment is manageable via SNMP. Furthermore, its rapid adoption in the telecom industry ensures that SNMP will continue to be a de facto management standard in all packet-based networks.

SNMP enables a system administrator to simultaneously manage a vast number of heterogeneous network elements (NEs), or devices, through an external management console such as HP's OpenView. This management is accomplished through the manipulation of Management Information Base (MIB) objects, which consist of a series of variables that reside on each device, and describe the state of the NE. The administrator can direct the SNMP console to monitor and alter the MIB objects on any device, thereby controlling its behavior. (Throughout the remainder of this discussion, I will use the terms "MIB variables" and "MIB objects" interchangeably.)

The console performs this manipulation of a device's MIB objects by sending out SNMP GET Protocol Data Units (PDUs) and SNMP SET PDUs. Each PDU encodes within it a list of the relevant MIB variables and values, if appropriate. This behavior is shown in Figure 1.

The SNMP agent, which resides on the NE itself, receives those PDUs, parses out the list of variables, and consults its management information base about those variables. The management information base, likewise, is found on the NE itself.

This parsing of the MIB objects within the PDUs is achieved through the use of the Structure of Management Information (SMI), which is a published, freely available, standard, and universal method of specifying a MIB object. The MIB objects in the PDUs are encoded both by the console and by the agent, according to the rules of SMI. And since SMI is a recognized standard, it is known both by the agent and by the console a priori to any NE management via SNMP.

Figure 2 illustrates how SMI works with respect to SNMP MIB variables. All MIB objects are laid out first according to the SMI hierarchy shown. That is, *all* MIB objects in the world of SNMP have the following prefix:

```
iso.org.dod.internet
```

FIGURE 3 Structure of management information through the system group

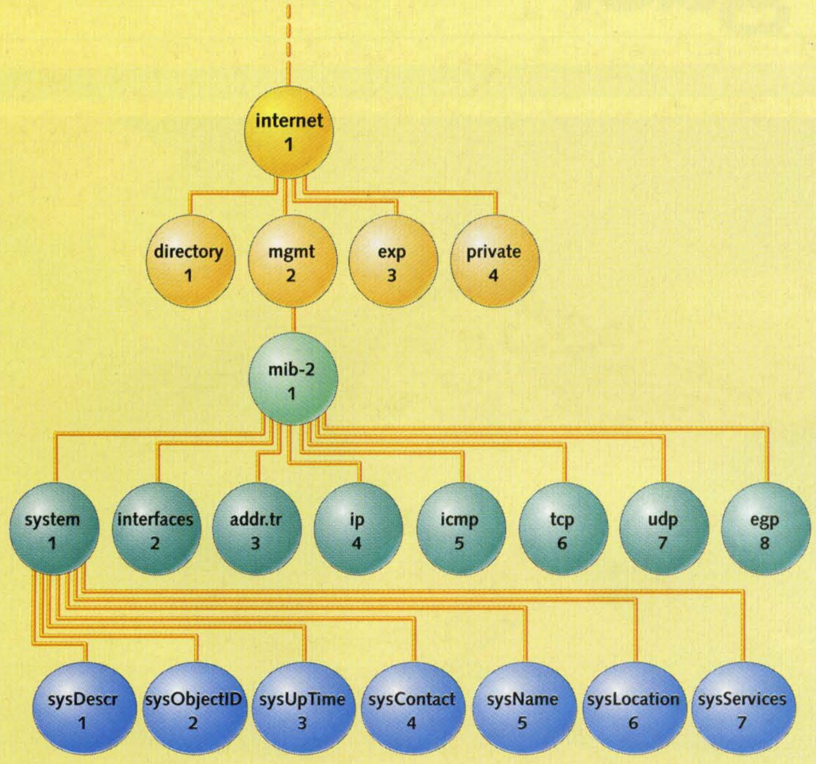
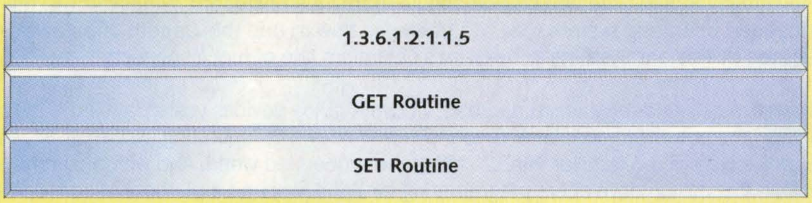


FIGURE 4 MIB object for sysName



or, in numerical form:

1.3.6.1

The other groups shown in Figure 2, `ccitt` and `iso-ccitt`, have never had any MIBs specified under them that have gained any kind of meaningful acceptance. Likewise, there have never been any other standard subgroups defined under the `iso`, `org`, or `dod` groups (so you will not be seeing any `MIB` objects starting with 2.1.4.2, for example, anytime soon).

Continuing on, suppose you wanted to specify a particular SNMP MIB object, say `sysName`. `sysName` is part of the `system` group, which, in turn, is part of `MIB II`. `MIB II`, which is part of the `mgmt` group shown in Figure 2, is a standard set of MIB objects, published in RFC form and implemented on virtually every packet-based NE that has SNMP. Both the console and the agent have a priori knowledge of the `sysName` variable.

To specify `sysName`, the SMI hierarchy shown in Figure 2 is merely taken out to deeper levels, and is shown in Figure 3.

Thus, `sysName` has the following full specification:

```
iso.org.dod.internet.mgmt.mib-
2.system.sysName
```

or, in numerical form:

1.3.6.1.2.1.1.5

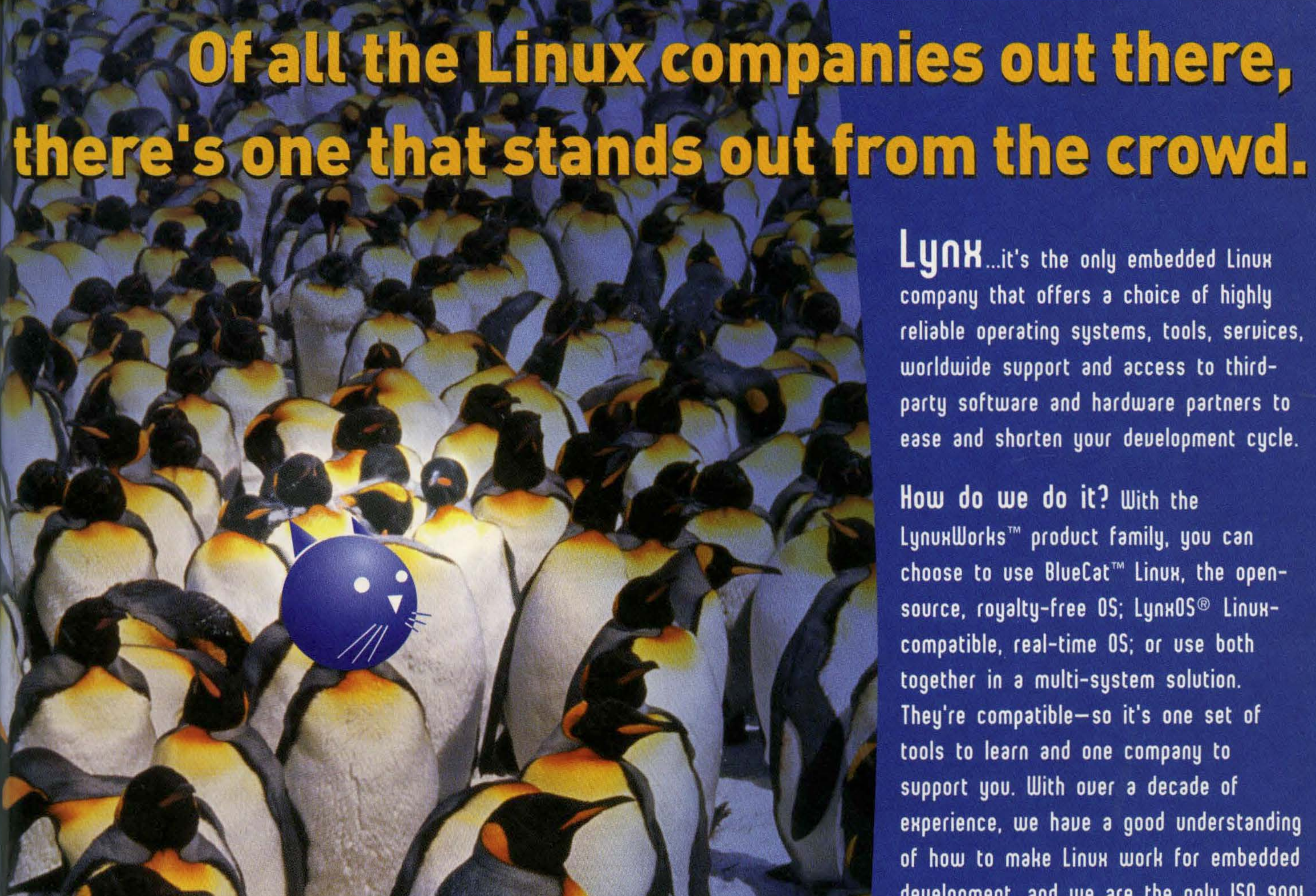
This specification is called an object identifier (OID). It consists of a base OID, 1.3.6.1.2.1.5. In addition, it has an instance of 0 appended onto it, for a total specification of 1.3.6.1.2.1.1.5.0. The 0 enumerates that the first and only instance of `sysName` should be returned, since `sysName` is known to be a scalar SNMP variable.

Thus, when an SNMP console wants to "get" or "set" the value for `sysName`, it sends out an SNMP GET or SET PDU with the OID 1.3.6.1.2.1.1.5.0 encoded within it.

The NE, in turn, will parse through the OID and consult its management information base. The management information base is a set of mappings between MIB variable OIDs and appropriate GET and SET routines. These routines provide read and write access to that data. An example of the mapping for `sysName` is shown in Figure 4.

Depending on the nature of the request, the SNMP agent then calls either the GET or SET routine associated with the relevant variable passes the instance variable 0 to the routine, and from there is able to monitor or update the state of its host's system name, as shown in Figure 5.

As these diagrams indicate, in order for an NE to be manageable by SNMP, it must have an SNMP agent that is capable of receiving, parsing, building, and sending SNMP PDUs, and it must have a management information base. The management information base, in turn, must have a significant number of MIB objects, with



Of all the Linux companies out there, there's one that stands out from the crowd.

Lynx...it's the only embedded Linux company that offers a choice of highly reliable operating systems, tools, services, worldwide support and access to third-party software and hardware partners to ease and shorten your development cycle.

How do we do it? With the LynuxWorks™ product family, you can choose to use BlueCat™ Linux, the open-source, royalty-free OS; LynxOS® Linux-compatible, real-time OS; or use both together in a multi-system solution. They're compatible—so it's one set of tools to learn and one company to support you. With over a decade of experience, we have a good understanding of how to make Linux work for embedded development, and we are the only ISO 9001 registered embedded Linux company.

What do you get out of it? Whether you're developing for telecommunications, post-PC consumer appliances, the Internet, military/aerospace or office automation, you'll gain a faster time to market, you'll never have to worry about compatibility or reliability, and you'll end up with a high-quality product that makes your customers happy.

And that, makes you stand out from the crowd.

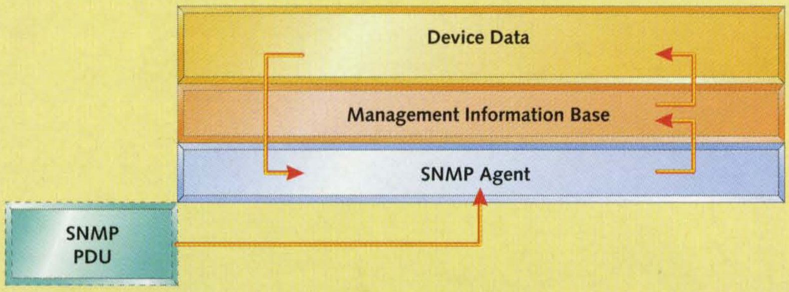
Lynx—the Leader in Embedded Linux **www.lynx.com**

Lynx

BlueCat™
Embedded Linux from Lynx

While WBM is a new management paradigm (it has only been around since 1995), it has gained remarkable mindshare and is rapidly becoming a standard mechanism by which NEs are managed.

FIGURE 5 SNMP agent behavior



their corresponding GET and SET routines, in order for the console to monitor and control the device in a meaningful fashion.

Commercial quality SNMP agents are provided by a number of vendors, including SNMP Research and Integrated Systems (ISI)/Epilogue (now part of Wind River Systems). In addition, a number of free SNMP agents have been developed, primarily at CMU or MIT.

However, the real "meat" of SNMP—the design, definition, and description of a device's MIB objects—must be done for each device, in almost a unique fashion. The reason for this is that the MIB objects need to access real data, which means there must be GET and SET routines written that access real variables used in the firmware. And since there are as many different ways of representing data as there are engineers, these GET and SET routines for *each* MIB variable are (relatively) unique to each device code line.

For example, the ISI/Epilogue agent requires a variation of the following system call to construct the MIB object for `sysName`:

```
LEAF_ASYS(sysName_leaf, "sysName",
VT_STRING, RW, SCALAR, it_exists,
get_sysName, set_sysName, std_
next, (char*)sysName, 0x0000,
0xFF, 0xFF);
```

Because of this, the developers at the network equipment vendor must *themselves* perform the task of writing the GET and SET routines (the `get_sysName()` and `set_sysName()` functions shown above)—there is no easy way out and no code that you can buy for MIB variable design and implementation.

Furthermore, that task is not limited to five or 10 MIB variables—NEs have encoded in them, at an absolute minimum, the roughly 200 MIB objects described in `MIB II`. In addition, many of these devices have literally hundreds of other MIB objects encoded within them, representing a development effort measured in the thousands of hours and hundreds of thousands of dollars.

However, even this does not fully describe the task. Most of the other MIB objects in the device are proprietary, which means two things:

- You have to have a knowledgeable engineer—a MIB-smith—to design and create the proper SNMP MIB objects
- The MIB-smith must generate a text document, written in a pseudo-programming language called Abstract Syntax Notation 1 (ASN.1), that fully describes the proprietary MIB on the NE

The first item is necessary because you want to have MIB objects that describe the state of your device in a meaningful way. The second item is necessary because you have to provide knowledge of the MIB objects in your device to the system administrators and their SNMP consoles. Otherwise, the SNMP console will not know the proper encoding for the proprietary MIB data, and how it fits in with the SMI. While the console will know the encoding for `MIB II` items, such as `sysName`, it will not know the encoding for anything unique to your device, and the a priori SMI knowledge mentioned earlier will not, in fact, exist. This will make the proprietary items on your device unmanageable.

So while SNMP is popular and likely to remain the de facto management paradigm for all packet-based networks into the foreseeable future, a non-trivial amount of engineering resources need to go into developing and maintaining a viable SNMP implementation on a NE.

Web-based management operation

While WBM is a new management paradigm (it has only been around since 1995), it has gained remarkable mindshare and is rapidly becoming a standard mechanism by which NEs are managed.

WBM allows a system administrator to manage an NE by simply "pointing" the Web browser at the device. The Web browser then makes a HyperText Transfer Protocol (HTTP) GET request to the device. From there, the device sends—over the same HTTP connection—a series of HyperText Markup Language (HTML) pages. These pages, which are nothing more than simple text files, tell the Web browser how to create the user interface (UI) of the device.

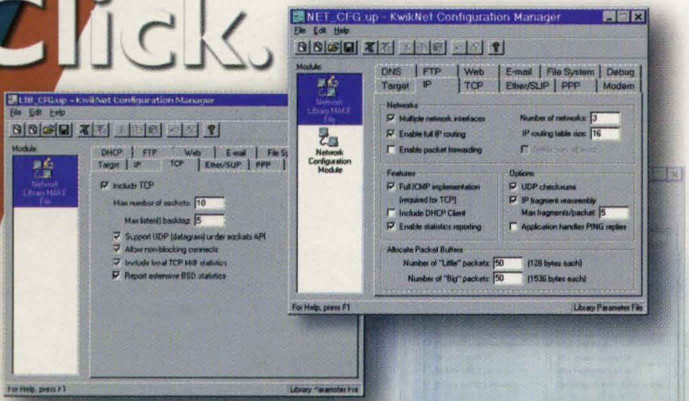
With the explosion of the World

KwikNet™

Adding network features to your embedded products?

KwikNet makes it quicker, easier, and less expensive.

Just Point and Click.


File Edit Help
Module:
Network Library MAKE File
Network Configuration Module
DHCP | FTP | Web | E-mail | File Sy
Target | IP | TCP | Ether/SLIP | PPP
DNS | FTP | Web | E-mail | File System | Debug
Target | IP | TCP | Ether/SLIP | PPP | Modem
For Help, press F1
Library Parameter File

LIB_CFG.up - KwikNet Configuration Manager

☑ Include TCP
Max number of sockets: 10
Max listen() backlog: 5
☑ Support UDP (datagram) under sockets API
☑ Allow non-blocking connects
☑ Include local TCP MIB statistics
☑ Report extensive BSD statistics

NET_CFG.up - KwikNet Configuration Manager

Networks
☑ Multiple network interfaces
☑ Enable full IP routing
☐ Enable packet forwarding
Number of networks: 3
IP routing table size: 16

Features
☑ Full ICMP implementation (required for TCP)
☐ Include DHCP Client
☑ Enable statistics reporting

Options
☑ UDP checksums
☑ IP fragment reassembly
Max fragments/packet: 5
☐ Application handles PING replies

Allocate Packet Buffers
Number of "Little" packets: 50 (128 bytes each)
Number of "Big" packets: 50 (1536 bytes each)

After more than 20 years of leadership in providing real-time multitasking kernels for mission-critical embedded applications, **KADAK** brings the same reliability and ease of use to TCP/IP stacks and web servers. With **KwikNet**.

When you use **KwikNet**, there is no need to spend months of time and incur the expense to manually create and test your TCP/IP stack.

We've already tested it.

And **KwikNet's** Configuration Builder lets you speed through the configuration complexities inherent in any TCP/IP stack!

Plus, as with any product from **KADAK**, you are given a royalty-free site license including source code.

You will also appreciate ***KwikNet's****:*

- portability to any RTOS with the minimum required features
- modular ANSI C code which has been tested on a wide range of target processors with 12 popular compilers

And if you add ***KwikNet's*** *Web Server option, you'll enjoy even more benefits.*

Find out all that it can do for you! Visit us at www.kadak.com

KwikNet TCP/IP Stack and Web Server

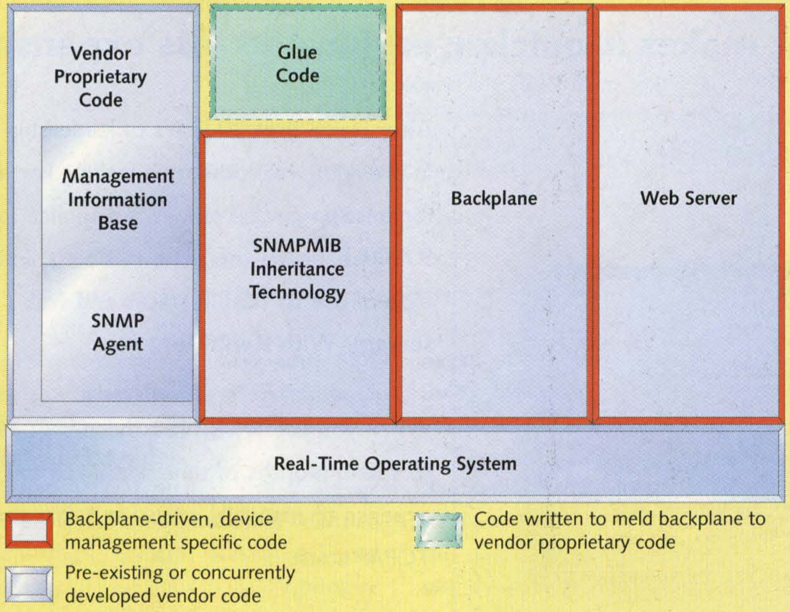
KADAK RTOS and Network Products

KADAK Products Ltd.
206-1847 W. Broadway
Vancouver, BC, Canada V6J 1Y5
Tel: (604) 734-2796 Fax: (604) 734-8114
Email: amxsales@kadak.com

KwikNet and AMX are trademarks of KADAK Products Ltd.

A backplane-based architecture allows a developer to link an HTML page to underlying data in a fashion that completely separates the HTML from the back-end C code.

FIGURE 6 Architecture for extensible, scaleable Web-based management solutions


Vendor Proprietary Code
Management Information Base
SNMP Agent
Glue Code
SNMPMIB Inheritance Technology
Backplane
Web Server
Real-Time Operating System
Backplane-driven, device management specific code
Pre-existing or concurrently developed vendor code
Code written to meld backplane to vendor proprietary code

Wide Web (WWW) in the last four years, a huge number of new technologies have been developed, including Dynamic HTML, eXtensible Markup Language (XML), JavaScript, and Java. All of these technologies serve greatly to enhance the capabilities of the "Web experience," both on the browser side and the server side. As a result, HTML files can now describe a UI that effectively mirrors, and often surpasses, the UIs found in conventional management applications.

Regardless of the technologies employed in the device's UI, those HTML pages have to be linked, somehow, to the underlying device data. That is, those pages have to be able to monitor and alter the host-specific dynamic data.

Thus, in order to implement a WBM solution, a vendor needs to have two things: an HTTP server embedded in his device—which is the main "packet engines" of WBM—and a mechanism for linking the HTML pages to the underlying data.

As with SNMP agents, commercial HTTP servers can be purchased or free versions can be ported. However, again like the SNMP paradigm, the real "meat" of WBM comes in linking the data in the device to its UI.

In the case of SNMP, the "meat" consists of linking the device data to the MIB objects. In the case of WBM, the "meat" consists of linking the device data to HTML pages.

A backplane-based architecture allows a developer to link an HTML page to underlying data in a fashion that completely separates the HTML from the back-end C code. By completely separating HTML from C code, the developer can hand off GUI design to the marketing department or an outside design firm to develop the device's UI *independently, without affecting the back-end code.* This allows for professional quality HTML UIs to be developed rapidly without tying up the embedded software developers. Furthermore, this it easy to modify the UI for OEM channels, localize it for foreign markets, and so on.

Lastly, by coupling a backplane-based architecture with an SNMP MIB inheritance library, a company can, in WBM, reuse all of the GET and SET routines written for its SNMP MIB variables without writing a line of code. This architecture allows device vendors to leverage the thousands of hours of work already spent in writing MIB variables and code, in a WBM scheme that is both flexible and scaleable for future business development.

How a backplane and SNMP MIB inheritance library work together

Figure 6 shows an architecture that provides a complete separation of the HTML UI from the back-end C code, and which allows a developer to leverage pre-existing SNMP MIB variable code without affecting embedded proprietary code.

A backplane-based architecture fits into a given embedded system as show in Figure 6. We assume that the typical embedded system has two primary parts: A real-time operating system (RTOS) and, more importantly, a vendor's proprietary code.

The RTOS provides the system with low-level services such as memory allocation, network services (TCP/IP), and semaphores.

The vendor's proprietary code provides the system with its unique value, the device's "reason for being." In the vendor's code, there is assumed to be an SNMP agent—either developed in-house or purchased commercially. In addition, there is a set of MIB objects, which access the device data for management via SNMP.

The glue code consists of a series of "Read" and "Write" wrapper routines (access methods) for the data found in the vendor's proprietary code, data that is not accessible via the SNMP MIB.

The Web-based management solution works with any and all types of HTML `FORM` elements.

FIGURE 7 Web-based management solution Web server architecture

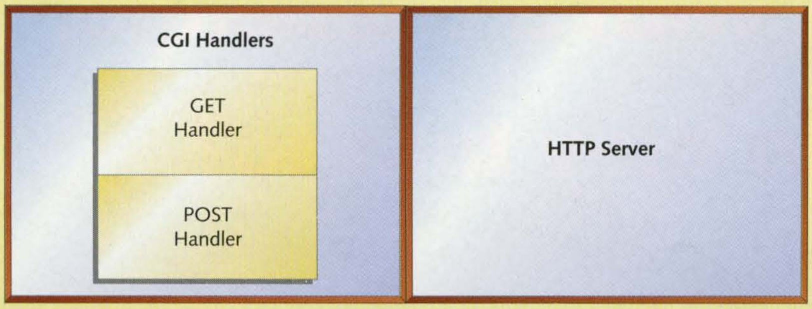
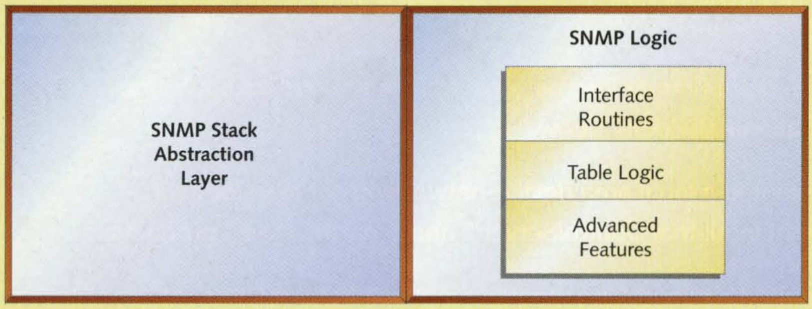


FIGURE 8 SNMP MIB inheritance technology architecture



The glue code is a necessary part of many WBM implementations. As you will see in the example detailed below, the backplane works in a very general sense—it is not limited to just one presentation media. As such, there is much data that the backplane works with that is wholly inappropriate to SNMP, such as dynamic data to specify HTML pages. Also, there is data that is not conveyed in a meaningful sense via SNMP, such as a dynamic log of entries.

Lastly, since there are as many different ways of representing data as there are individual engineers, a viable Web-based management solution needs the data access written in a special format that it can understand.

This Web-based management solution consists of three parts:

- A backplane. This is a database of mappings between special HTML tags and function pointers to the relevant "Read" and "Write" routines found in the glue code. The backplane and the special HTML tags are discussed at length below
- The Web server. This module consists of a fully functional HTTP 1.1/1.0 server plus a set of common gateway interface (CGI) handlers. CGI is a standard approach to handling data exchanges between an HTTP server and ancillary programs. The Web server architecture is shown in Figure 7
- The SNMP MIB Inheritance Technology. This module is a general-purpose code library that handle SNMP specific queries from the backplane. It consists of an SNMP abstraction layer (so that it can effectively mate with a wide variety of different SNMP stacks), conversion routines, and logic to work with advanced SNMP principles, such as SNMP tables. The architecture is shown in Figure 8

Basic operation

The easiest way to describe how the backplane achieves the complete abstraction of HTML from the underlying C code while leveraging preexisting SNMP MIB variables, is through a pair of examples. The examples are simplified for the sake of this discussion, but the concepts they illustrate are complete.

To show how this set of technologies works as a whole, we will first go through an example where the backplane accesses some data that is only available via a glue code routine. Next month, we will go through a parallel example where the data is available from a pre-existing SNMP MIB object.

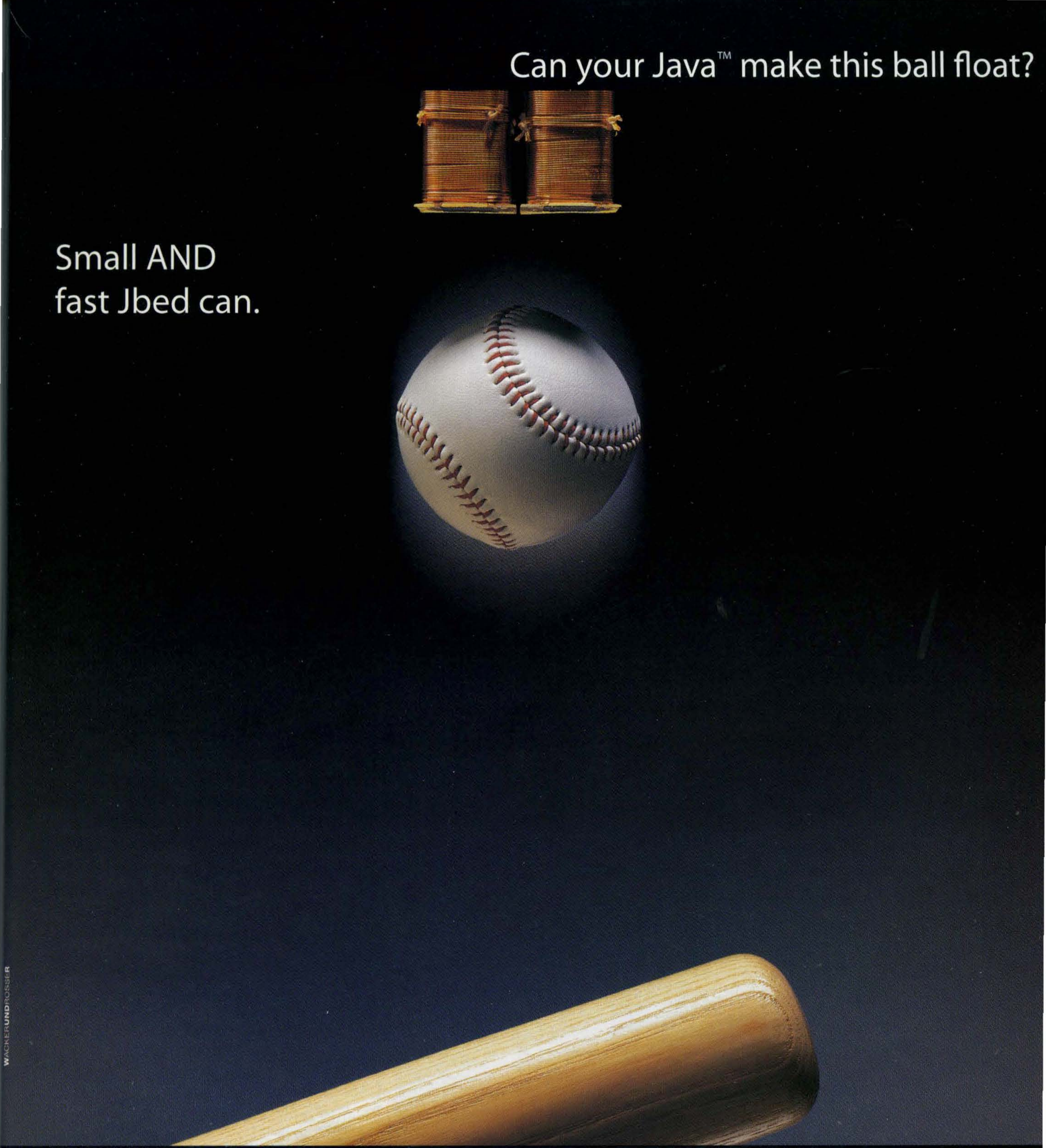
Backplane operation with glue code routines

Assume that the following HTML is located in a file named `MYFILE`, and `MYFILE` is stored on an embedded network device using such a backplane architecture:

```
<INPUT TYPE="text" SIZE="20"
NAME="userName"
VALUE="$%userName#$">
```

In this instance, the HTML referenced above is a `FORM` element that is used to describe a one-line text box. A text box is a common element in Web-based management because it allows an end-user to read and modify a particular string (be it a device name, an IP address, and so on) through the Web browser. The Web-based management solution works with any and all types of HTML `FORM` elements.

Note also that this is nothing more than standard HTML (albeit with an interesting `VALUE` field), and as such it can be written or modified by any ordinary HTML editor.


Can your Java™ make this ball float?
Small AND
fast Jbed can.
WACKERUNDROSSER

The GET handler, after making any other tag substitutions that might be in the HTML file, then sends MYFILE out to the Web browser via the HTTP server.

FIGURE 9 File loaded into GET CGI handler

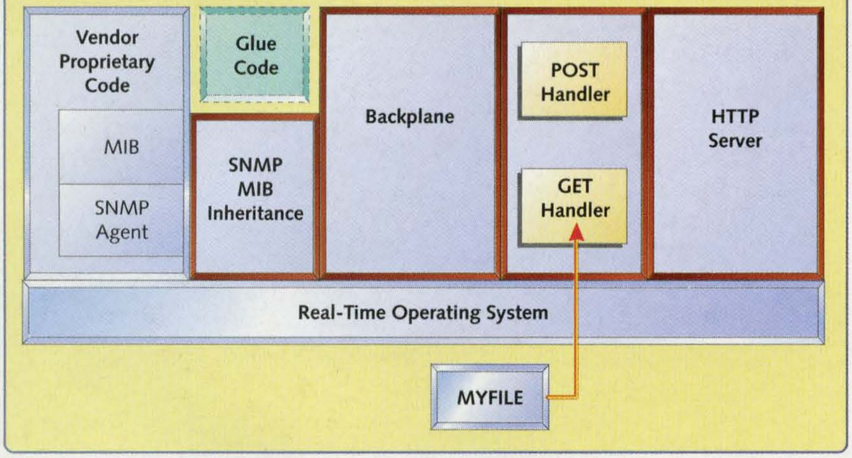


FIGURE 10 Backplane entry

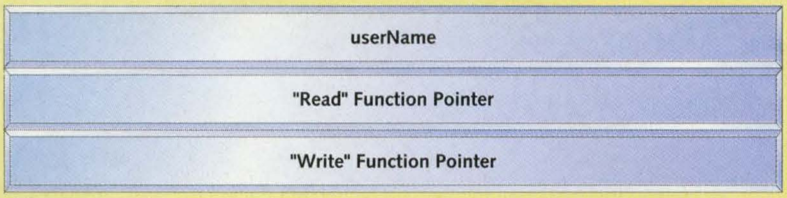
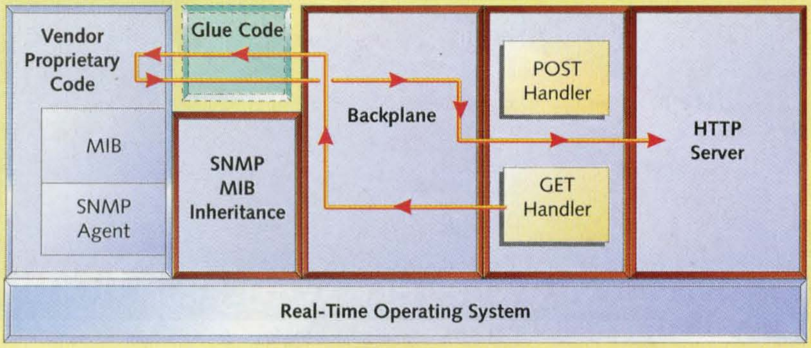


FIGURE 11 HTTP GET operation



When a Web browser asks for (that is, makes an HTTP GET request for) the URL that corresponds to `MYFILE`, the HTML in that file is first loaded into the GET CGI handler, as shown in Figure 9.

The GET handler then steps through the HTML character-by-character until the handler runs into "`$%`," the notation for the beginning of the special HTML tag. This pair of symbols, which does not appear in ordinary HTML, signifies to the GET handler that the following text is not standard HTML. In this particular example, "`$%userName#$`" is the special tag.

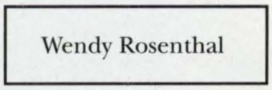
At this stage, the GET handler stops parsing the HTML and looks up `userName` in the backplane, which is essentially an online database of the special tags. The backplane returns a database entry to the GET handler, as shown in Figure 10.

The GET handler now has a pointer to a "Read" glue code routine, as shown in Figure 11, which it calls. The "Read" routine accesses the vendor's proprietary code, finds the relevant data, and copies it into the HTML that is streamed out to the remote browser.

Assuming that the "Read" routine retrieves and copies in the data "Wendy Rosenthal" in lieu of "`$%userName#$`" the following HTML is constructed:

```
<INPUT TYPE="text" SIZE="20"
NAME="userName" VALUE=" Wendy
Rosenthal">
```

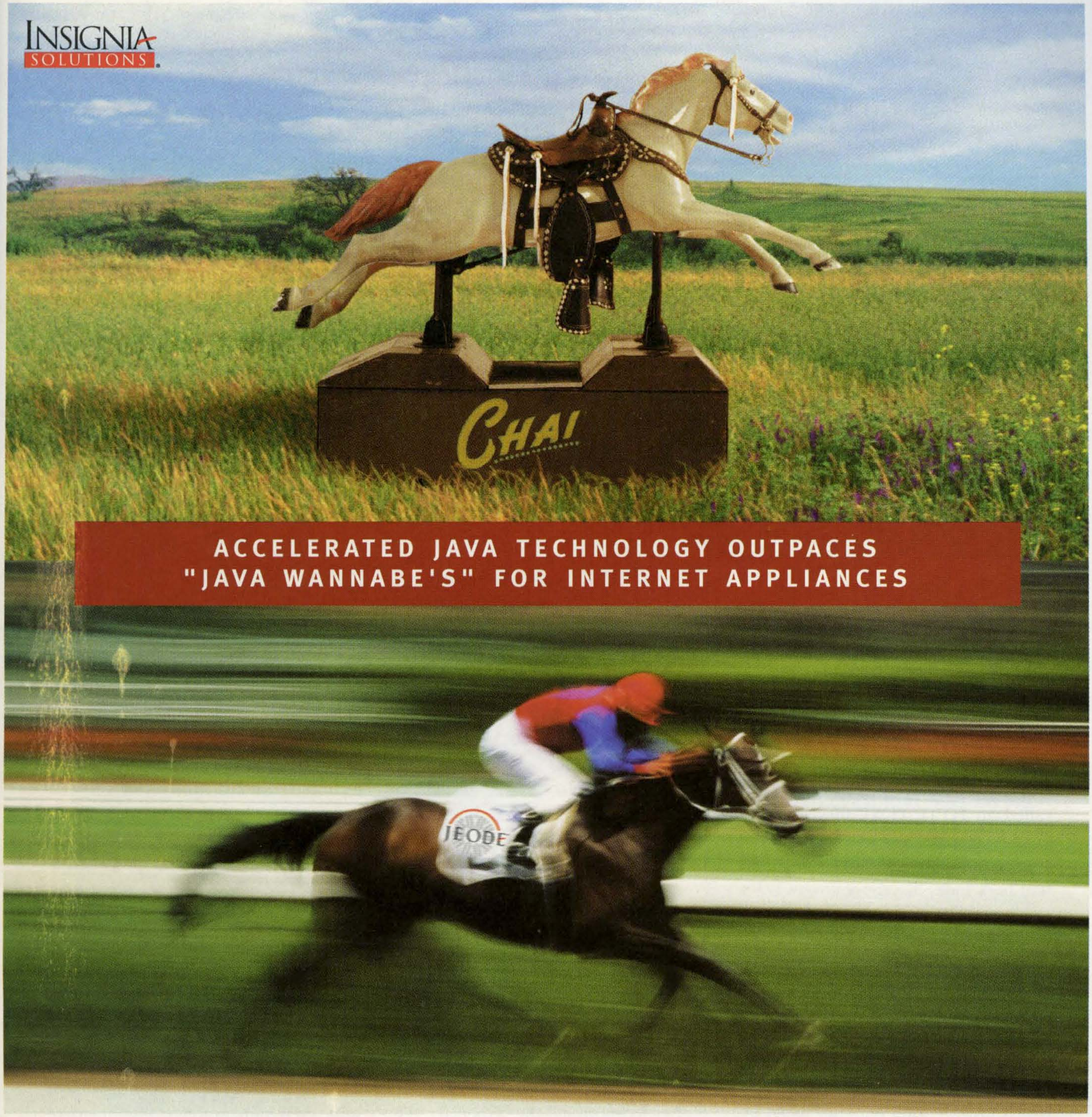
The GET handler, after making any other tag substitutions that might be in the HTML file, then sends `MYFILE` out to the Web browser via the HTTP server. The Web browser then displays the HTML in the example as:


Wendy Rosenthal

As mentioned earlier, this text box is an HTML `FORM` element. HTML `FORM` elements are essentially variables, where the name of the variable is specified in a `NAME` field, and the value of the variable is specified in a `VALUE` field. In the example above, the `NAME` of the text box is `userName`, and the `VALUE` is "Wendy Rosenthal."

Now, when an end-user wants to change the value in the text box, he or she simply highlights "Wendy Rosenthal" in the Web browser, hits the delete button, and then types

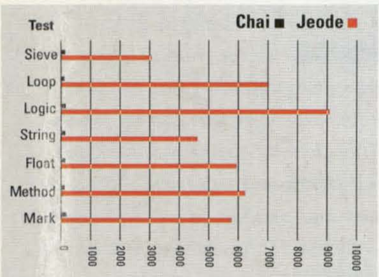
INSIGNIA
SOLUTIONS



ACCELERATED JAVA TECHNOLOGY OUTPACES "JAVA WANNABE'S" FOR INTERNET APPLIANCES

In the race for high performance Java™ solutions for Internet appliances and embedded devices, Java technology plus acceleration is the winner. Jeode™ is Insignia's accelerated, fully compatible and certified implementation of the PersonalJava™ and EmbeddedJava™ specifications. It leaves non-certified Chai® six lengths back in head-to-head tests (see chart). How? At Jeode's core is the fully compatible, Embedded Virtual Machine™ (EVM™), an accelerated runtime engine that delivers high performance through patented Dynamic Compilation. The EVM is robust, has a small memory footprint, and it's highly configurable. So get Jeode for accelerated Java technology. It's real. It's proven. And it's blazingly fast. Win big!

Embedded CaffeineMark 3.0



www.insignia.com/jeode or call **1.800.848-7677.**

JAVA
Sun Authorized
Virtual Machine

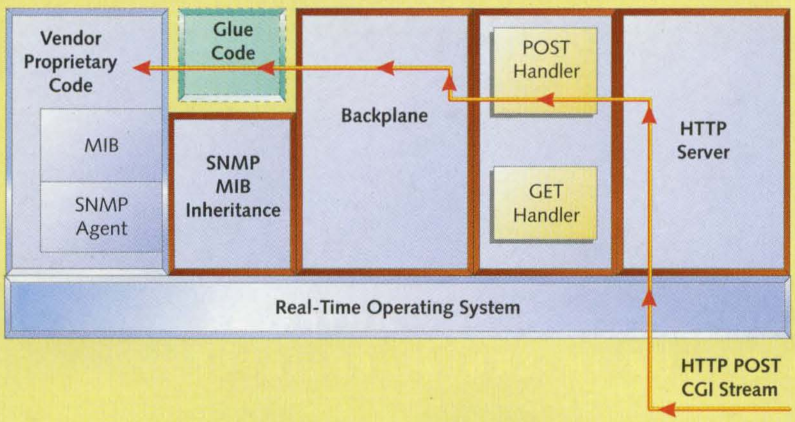
JEODE™

Accelerated Java Solutions
for Internet Appliances and
Embedded Devices.

Insignia and Insignia Solutions are registered trademarks and Jeode, Embedded Virtual Machine, and EVM are trademarks of Insignia Solutions, Inc. Sun, Java, EmbeddedJava and PersonalJava are trademarks of Sun Microsystems, Inc. CaffeineMark is a trademark of Pendragon Software. Chai is a registered trademark of Hewlett-Packard Company.

And just as a form letter template can be easily modified with any word processor, so, too, can the HTML template file be easily modified with any HTML editor.

FIGURE 12 HTTP POST operation



"Gayle Semrad." After that, the end-user hits the submit button.

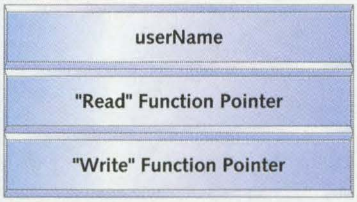
At this stage, the Web browser sends back a message to the Web server with a stream of data in it. The message is called an HTTP POST message, and the data is called a CGI stream. For the above example, the CGI stream that is sent back looks like the following:

```
. . . &userName=Gayle+Semrad& . . .
```

Notice how the name-value pair of the CGI stream is identified by `userName`, which is the name of the one-line textbox as shown in the HTML listing. The value part of the name-value pair is just what the end-user typed in. This is why the HTML `FORM` elements have a `NAME`—so that the appropriate element and its value can be readily identified in the CGI stream.

This CGI stream is received by the embedded Web server and passed to the Web-based management system's POST CGI handler. This is shown in Figure 12.

From there, the POST handler parses through the CGI stream. When it gets to `userName=Gayle+Semrad`, the POST handler then looks up `userName` in the backplane. The following entry is returned to the POST handler:



Now, the POST Handler has a pointer to a "Write" routine in the glue code, which it calls. The "Write" routine takes the "`Gayle+Semrad`" data, accesses the vendor's proprietary code, and makes the appropriate update. (The '`+`' sign is appropriately dealt with by the post-CGI handler code.)

By designing the special tag "`$%userName#$`" in the HTML file to match the `NAME` of the `FORM` element, `userName`, the same database entry in the backplane can be used bi-directionally, or both in the "outbound" direction and in the "inbound" direction.

This is the essence of how a backplane-based Web-based management system operates. Note that the HTML in `MYFILE` is completely unchanged. If the end-user hits "reload" on his/her Web browser, or if another Web browser were to access it, the GET operation would be performed again. This time, instead of copying in "Wendy Rosenthal," the GET handler would copy in "Gayle Semrad."

This model essentially takes the concept of a form letter—namely, a form template and a database that are merged to create the illusion of a personalized letter—and moves it into the realm of HTML/HTTP/CGI. The original template, in this case, is simply the HTML file `MYFILE`. And just as a form letter template can be easily modified with any word processor, so, too, can the HTML template file be easily modified with any HTML editor.

However, as demonstrated above, the HTML/HTTP/CGI mechanism takes the form letter model to a new level. By choosing the special tags so that they match HTML `FORM` element `NAME`s, one can essentially have a *bi-directional* form letter.

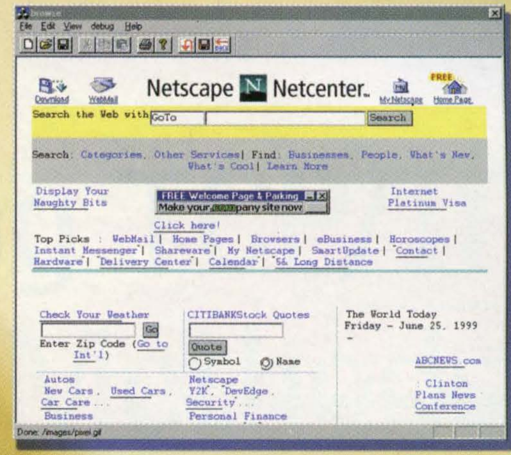
With this architecture, if a company wants to change the UI for rapid prototyping, for localizing to foreign markets, or for setting up new OEM channels, the only thing that needs to be modified is the HTML—and that can be modified by using any number of standard tools. *The rest of the system image can remain completely unchanged.*

Likewise, if file-based uploads are supported, then the UI of the device can be dynamically altered and engineering doesn't have to be involved in the process at all.

Next month, I'll show you how to make examples like these a reality for any SNMP-enabled system. We'll look at some even more powerful advantages of adding WBM-support this way. **esp**

Kedron Wolcott is VP of engineering for Rapid Logic, which he co-founded in 1996. With nearly 10 years of experience, he has expertise in Internet-based management for network devices. Before Rapid Logic, he developed routing, bridging, and frame relay software for Tribe Computer Works. Wolcott joined Tribe from the Kennedy Space Center, where he designed and implemented the data acquisition and analysis software and robot control software. He has earned engineering degrees from both MIT and Stanford.

Need Drop In Deeply Embedded Solutions Beyond Protocol Stacks?



Netscape Netcenter.

Search the Web with GoTo

Search: Categories, Other Services | Find: Businesses, People, What's New, What's Cool | Learn More

Display Your Naughty Bits

FREE Welcome Page & Parking
Make your company site now

Click here!

Internet Platinum Visa

Top Picks : WebMail | Home Pages | Browsers | eBusiness | Horoscopes | Instant Messenger | Shareware | My Netscape | SmartUpdate | Contact | Hardware | Delivery Center | Calendar | 5¢ Long Distance

Check Your Weather
Enter Zip Code (Go to Int'l)

CITIBANKStock Quotes
○ Symbol ● Name

The World Today
Friday - June 25, 1999
-

ABCNEWS.com
: Clinton Plans News Conference

Autos
New Cars, Used Cars, Car Care...
Business

Netscape
Y2K, DevEdge, Security...
Personal Finance

Actual NicheView WWW screen shot

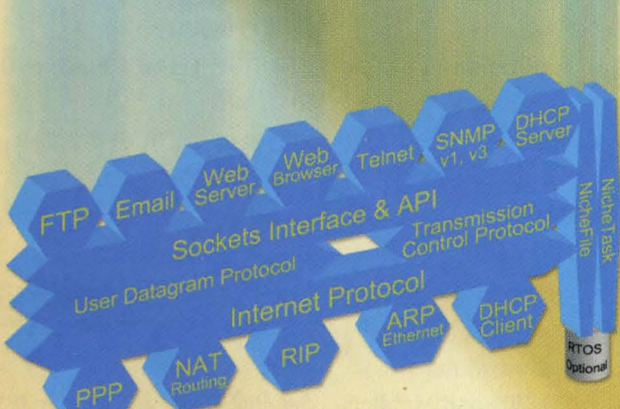
InterNiche's Drop In Networking Solutions Can Take You From Design To Production Quickly.

Our solutions include all you'll need for building embedded Web & DHCP servers, routing and more.

Our NicheView™ Deeply Embedded Browser is the preferred drop in solution for cost-conscience designs.

NicheView's unmatched patent pending implementation provides a fully-operational, fully-featured Browser in just 30K Bytes.

With support for Graphics and Plug-ins, your design is not limited by typical embedded browser technology.



RTOS & CPU independent packaged solutions designed for small footprint application needs.

InterNiche Technologies

1340 DeAnza Blvd., Suite 208
San Jose, CA 95129
1.408.257.8014
1.408.257.5692 FAX
sales@iniche.com
www.iniche.com

Embedded Solutions That Drive Change

WE CALM THE SEAS OF TECHNOLOGY

WHY PACIFIC SOFTWORKS ?

Since 1982, Pacific Softworks has been delivering the quickest and most stable Internet Protocols available in the industry. We're proud of our reputation as the leading supplier of protocol software.

Royalty Free or Royalty based licensing
End-to-End support from the Protocol Specialists
Ported and Tested Compatibility with any Processor, Kernel or RTOS
Flexible and Scalable solutions to protect your investment
Demonstrated reduction of time to market
Leading the industry with next generation products and solutions World-wide presence
...and, most importantly, we have Happy Customers!

Visit our website for more in-depth information on Fusion, the "off-the-shelf" FastTrack and other leading embedded solutions.
www.pacificsw.com , e-mail: sales@pacificsw.com

Pacific Softworks provides a single supplier solution ensuring that today's successes are the building blocks for tomorrow.

• **Protocols:**	TCP/IP (IP v4 and v6*), UDP/IP, DHCP, BOOTP
• **Applications:**	FTP, TFTP, TELNET, DNS, SMTP, POP3, RPC, NFS
• **Internet:**	PPP, SLIP, CSLIP, CHAP, PAP, MultiLink PPP*
• **Web:**	Embedded Web Server, Embedded Web Browser, Web Management
• **Network Management:**	SNMP v1/v2, MIB Compiler, SNMP v3*
• **Routing:**	RIP/RIP2, OSPF, BGP4, L2TP*, NAT
• **Security:**	IPsec*
• **Satellite and Mobile:**	Satellite TCP*, and Mobile IP*
• **WAN Protocols:**	ISDN, ATM, Frame Relay and X.25 (Integration with Third Party WAN Products)
• **RTOS:**	ThreadX™ from Express Logic
• **Development Tools:**	Porting Wizard and Installation Wizard

**product in development*
ThreadX™ is a registered trademark of Express Logic, Inc.

Pacific Softworks
our wave... your Success.

California Corporate Headquarters:
tel: 800.541.9508 fax: 805.499.5512
European Headquarters:
tel: 44.1494.432.735 fax: 44.1494.432.728
Japan/Asia-Pacific Headquarters:
tel: 81.3.5669.7722 fax: 81.3.5669.7723

CHRISTOPHER D. LEIDIGH

Configuring TCP/IP Hosts, Methods, and Protocols

Every TCP/IP-enabled device needs to be configured. Of course, there are several valid ways to go about it. This article will help you select the best configuration technique for your system.

Everything needs to be configured by someone. The fact of the matter is that configuration of computers and some embedded devices continues to be a hassle. Embedded systems can actually be some of the worst culprits in this regard, due to the fact that many tend to be "interface-challenged." (Need I mention the blinking "12:00" on your VCR?) Any system, embedded or otherwise, participating in a TCP/IP network requires at least some minimal configuration information. I'll refer to this as "IP configuration," given that the parameters are IP configuration elements. Currently, six basic configuration methods or schemes are in use. While I'm sure that others may exist, those presented here are used in the mainstream. The six methods are static local assignment, preset static unicast or multicast addresses, *Reverse ARP* (RARP), *gleaning*, *Bootstrap Protocol* (BOOTP), and *Dynamic Host Configuration Protocol* (DHCP). These will be discussed along with their uses and limitations.

Basic IP configuration information

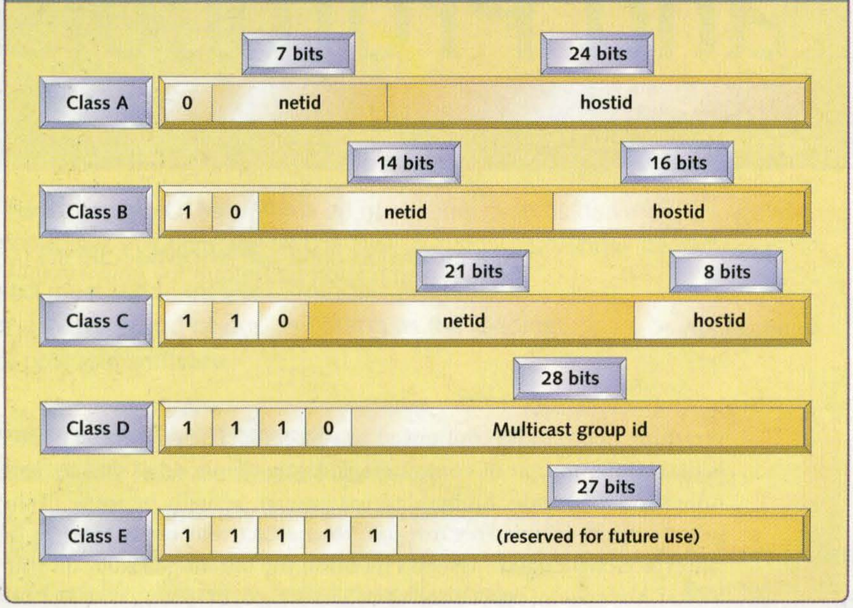
Any host "actively" participating in a TCP/IP network requires at least three basic IP configuration elements: a network-unique IP address, a subnet mask, and a default gateway IP address. By "actively" we mean to say that our host needs to be able to communicate with other nodes on the network using the TCP/IP protocols.

If a host were just a listening device, such as a network monitoring tool or probe, then no such information would be required. Such a device usually listens on a network in a passive, "promiscuous" mode (see Thomas Herbert's

"Introduction to TCP/IP," December 1999, p. 57) and records and displays all network traffic. For our purposes, we assume that we actually want to communicate. Therefore, we need to configure each network-connected device such that it has a unique IP identity.

Of the three elements, the *IP address* is the most important. This is the 32-bit address of the device and it must be unique on the network in which the host resides. If the host participates in a publicly routed network (that is, attached to the Internet), then the address must be globally unique. In general, this means that the address is within the range of addresses for an Internet-registered network. IP addresses are usually written in *dotted decimal notation* such as 159.215.6.40, with the dots separating bytes. IP addresses are actually composed of three parts: a *network id* portion, a *subnet id* portion, and a *host id* portion.

FIGURE 1 IP address classes



IP addresses are registered and assigned by the Internet Addressing Board (IAB) and are categorized into *network classes.* The different classes are listed in Figure 1. The class table shows that the different class networks define the maximum number of bits that can be used to describe the subnet and host portions of an address within a network. For example, the IP address 159.215.6.40 is on a Class B network with a network id of 159.215, the 6.40 represents the subnet/host id portion. (This decoding is accomplished with the aid of the *subnet mask.*)

The subnet mask is a separate IP configuration element that is a bit mask for the network id and subnet id portion of an address. While not a strict requirement, subnet masks are typically chosen on byte boundaries so that it is easy to read IP addresses and determine what subnet they are on. If 8-bit subnets are used for the example network address, then the subnet mask would be 255.255.255.0. Since the Class B address uses the top 16-bits (159.215), the subnet can be obtained by a bitwise OR of the IP address and subnet mask, and subsequently bitwise ANDing out the top 16 bits of the network id for a result of 0.0.6.0, the "six" subnet.

Subnets are used in TCP/IP to logically divide networks into smaller portions. Dividing networks is necessary to segment traffic and provide for a means of associating locations with subnets. It is the job of *routers* or *gateways* to move TCP/IP packets between IP subnets on an as-needed basis. For example, to send an e-mail from one company to another, a message would have to pass from Company A to the Internet and into Company B. Such a journey would typically involve many "router hops" as the message moved between Company A's subnets, portions of the Internet, then Company B's subnets.

The *default gateway* configuration parameter is used by TCP/IP hosts that need to communicate with other systems that are not on the same IP subnet. Packets destined for an off-subnet address will be forwarded by the sending host to the default gateway. The gateway (router) will then forward the packet in the appropriate manner based on information it has from routing protocols. A TCP/IP host need not know the path to other hosts outside the local subnet, it can leave this job to the default gateway. An isolated LAN situation would actually not need either the subnet mask or default gateway since everything would be on the same IP subnet. Such a situation might occur in a small, isolated office LAN.

These three pieces of information serve as the "minimal" configuration for a typical TCP/IP host. However, by no means do they represent even a fraction of the total possible configuration elements. Almost all other elements would be associated with specific features or TCP/IP applications implemented on a host. This extended configuration information can either have well-known defaults or be configured by a user remotely over the network. The main difference is that the IP address, subnet mask, and gateway are all specific and unique to a user's network and the actual host. Therefore, they cannot have defaults.

Static local assignment

There is no official networking term "static local assignment." Some texts do use the term "static assignment" for

BIG FEATURES
SMALL FOOTPRINT

U S Software products are compatible with:

x86	M*Core
386/486PM	ColdFire
680x0/683xx,	MIPS
DragonBall EZ	C166
PowerPC	CR16
ARM	8096/196
StrongARM	68HC11
SHx	68HC16
SPARCLite	H8/300
i960	8051/80251
NEC V85x	Z80/180

With thousands of embedded applications and 24 years in the business, U S Software continues to build cutting-edge, yet compact development tools.

Our software has a history of acclaimed performance and features:

- Compact – Even with industry-leading innovations our small footprint saves you valuable space.
- Fast – Optimized design ensures quick, efficient performance.
- Customizable – Source code is included to give you complete control.*

In the beginning there was silicon. Then, there was U S Software. And throughout, our engineers have crafted software and provided technical support designed to ensure your success.

Call us today at **800-356-7097** and entrust your next embedded project to the engineers and products at U S Software.

- **USNET®** Embedded TCP/IP protocols, Internet Access Package, SNMP and Embedded Webserver.
- **SUPERTASK!™** Multitasking RTOS development suite. ITRON 2.x/3.x API available.
- **USFILES®** Windows 95/DOS compatible file system with CompactFlash and CD-ROM support.
- **NETPEER™** Distributed networking platform with comprehensive desktop interface including GUI, database, multimedia and more.
- **SOFT-SCOPE®** x86 remote-target debugger for real and protected mode applications.

800-356-7097

WWW.USSW.COM

*Netpeer™ is delivered as a library.

U S SOFTWARE®
EMBEDDED EXCELLENCE

7175 NW Evergreen Pkwy. Suite 100 Hillsboro, OR 97124
TEL: 503-844-6614 • FAX: 503-844-6480 • EMAIL: info@ussw.com • www.ussw.com

this configuration method. However, "local" is added here to distinguish this method from the remote static methods described later. In general, most hosts will have a method for static local assignment of IP configuration information. However, it is assumed that there must be some form of a user interface to do so. While PCs have screens and keyboards, embedded systems are typically limited to serial ports, LCD screens and buttons, or even binary switches. For embedded hosts the most typical static configuration scenario would require a user to start a terminal program connected to a serial port and configure the device's parameters through a menu or command line interface.

The static local assignment method is by far the most widely used when considering all TCP/IP hosts. While certain categories of devices may utilize other methods to a greater extent, static local assignment is administratively the easiest to use and implement in most situations. Hosts can be centrally configured and then deployed to their final destinations where they can expect to boot up and be usable. Of course, other configuration information could be assigned in a local manner as well.

The key question is really when not to use static local assignment of IP configuration information. The answer to this question depends on two general issues: how and where the device is used and how it is deployed. For example, at the company I work for, we offer network management cards for many of our uninterruptible power supply (UPS) products. At a large customer's site, hundreds of these may be deployed at once. Consider two different deployment methods:

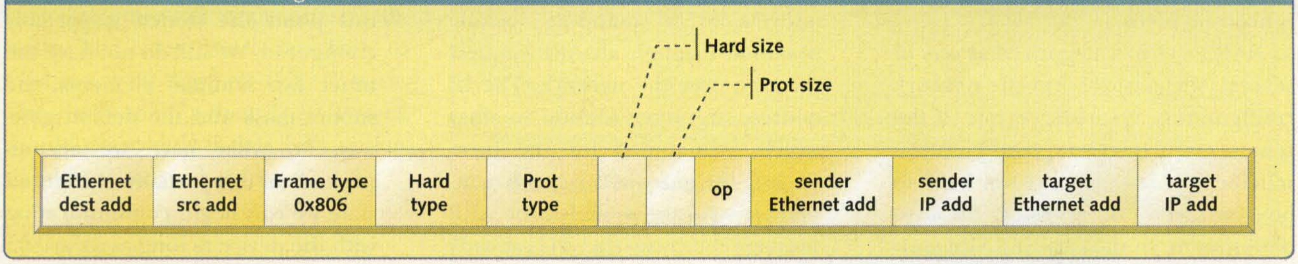
1. The customer has a centralized configuration strategy where equipment is staged and fully configured before it is deployed to its final location
2. The customer drop-ships "raw" unconfigured equipment to the deployment sites, where it is subsequently installed by field teams

The first scenario would require each device be configured one by one through the serial port, a time-consuming task when hundreds of units are involved. This method, however, requires only one central configuration team versus many experienced field teams. The second scenario could actually utilize either local static assignment by the field teams or one of the dynamic or remote methods explained later.

Static, preset unicast, or multicast addresses

Using factory preset addresses can eliminate the need for direct, local address assignment described above. These methods can accomplish over-the-network configuration in some circumstances. Assuming that a network host to be installed has a factory default address of 192.78.78.66, a user can temporarily change the address of their computer to be on the same pseudo subnet for the purpose of configuration, perhaps using address 192.78.78.67. After doing so, the user can use built-in access protocols, such as telnet, to access the new host and configure it appropriately for the local network. Following this, the user would change the configuration computer's settings back to normal so that both the new device and the computer could communicate on the operational network.

FIGURE 2 RARP message format



While this method eliminates cables and local interfaces, it is not always practical or desirable to have to change a computer's address information just to configure a new network device. For a very small deployment this may be reasonable, but it is not a possibility for a large rollout where installers may not be well trained. The requirement of being on the same IP subnet also precludes this method from being used remotely. Another limitation is that only one device can be added to the network and configured at a time to avoid simultaneous duplicate addresses (all the units would have the same default factory address).

A small variant to this method utilizes an IP multicast address as the factory pre-configured address. Using a multicast address has the advantage of not requiring an address change to the local computer used for configuration. The reason behind this is that IP multicast addresses are independent of IP subnets and, therefore, the configuration computer can communicate with the host's preset multicast address with no address change.

Reverse ARP (RARP)

In the "old" days (when I was still in college), many terminals that connected to TCP/IP networks were both diskless and without local interfaces. (Remember when X was the only kind of windows?) Not only did these systems lack local nonvolatile storage for applications, but they typically did not store their IP configuration parameters either. In these systems it was necessary to obtain IP configuration information "dynamically," at bootup. Reverse ARP (RARP) based on the *address resolution protocol* (ARP), was designed to accommodate such systems. The protocol is described in RFC 903. The ARP protocol is used to bind hardware addresses—also referred to as Media Access Control (MAC) addresses—to IP addresses. When a host needs to communicate with another TCP/IP host, it broadcasts an ARP message requesting the owner of the destination IP address to respond with its hardware address. The RARP protocol is just the reverse lookup; given a host's hardware address a request can be made for the corresponding IP address.

A RARP server is a process that can run on one or more host server systems in a local IP subnet. These servers contain configuration files that specify IP addresses for RARP clients. Each entry would contain an IP and hardware address pair. Host systems are configured to use RARP to obtain an IP address by broadcasting an RARP request at startup. The RARP message, shown in Figure 2, contains the client's hardware address and network type (Ethernet, Token Ring, or FDDI). Given that the request is sent as a link layer broadcast, all nodes will receive the request and the client doesn't need to know the addresses of RARP servers explicitly. Upon receipt, a RARP server will look up the client's hardware address in its tables. If a match is found, the server will send a directed response message, a RARP reply, specifying the IP address for the client. Multiple RARP servers can be used for redundancy or other reasons. If several RARP servers have an entry for the same client they will both reply to a request and the client will just process the first reply. For the sake of reducing network traffic, servers can be configured as primary and secondary. Secondary servers would only respond if the primary were down and a second client request was received.

While RARP is simple to implement and use, this configuration mechanism is only capable of supplying the IP address. The subnet mask and gateway would have to be configured in another manner. Why would such a mechanism be used given the limitations? The advantages of RARP lie in the fact that nodes need not be directly configured for deployment. Connecting cables and accessing configuration menus are avoided. In the case of embedded systems where interfaces such as keyboards and screens usually do not exist, this can be much easier for installation. Network managers can record the hardware addresses of hosts, place them in a RARP configuration file along with IP addresses and then boot up the hosts. Subsequent IP address changes can also be made by changing the RARP configuration files and rebooting the RARP client. When the client restarts, it makes another RARP request and receives the new address.

Given the fact that only an IP address can be obtained through RARP, additional information must be configured through other means such as a telnet session to the host. Telnet is a basic remote console program that usually provides access to a host's command line interface over the network. Through the use of this command line interface, a user could configure the remaining items.

RARP can be categorized as one of several dynamic, over-the-network configuration methods. As one of the tenets of a dynamic configuration scheme, the host should not have to be configured or accessed prior to deployment. In the case of RARP, only the hardware address need be noted for placement in the RARP configuration file. The broadcast of the request message avoids the need to know about specific RARP servers, which would require preconfiguration. An important limitation, however, is that the broadcast message will only be sent on the local subnet as routers do not forward broadcasts. This limits the use of RARP to the local subnet. That is both the clients and servers must be on the same subnet.

Gleaning

The method of IP address assignment through gleaning is not as widely known or implemented as the other schemes described here. One of the few devices to use such a scheme is the Axis line of print servers. Gleaning is also a dynamic method requiring no prior configuration of a device. It also takes advantage of the fact that hosts have unique hardware addresses.

There is no RFC describing the gleaning mechanism and, therefore, there is not necessarily one way to implement gleaning. The basic premise is that if a host is in an unconfigured state—that is with no IP address information—and it receives a link layer directed packet, the host can "glean" its IP address from the IP address in the packet. A step-by-step explanation is as follows:

1. The user places the hardware address of the host being setup into his/her workstation or PC's ARP cache through a command such as *arp –s 00:c0:b7:44:88:99 159.215.6.22.* This command, which is available in various forms on almost any Unix and Windows system, gives the PC the hardware address to IP mapping usually obtained through an ARP request message over the network. The IP address argument should be that which is intended for the host. Placing the entry in the cache manually *prevents* the need for the ARP message because the PC already has the mapping. Since the host being setup by this method does not have an IP address, it could not and would not respond to an ARP request.
2. Subsequent to "priming" the ARP cache, the user issues a *ping 159.215.6.22* command to send an ICMP echo request to the device. Normally, the PC would have to first issue an ARP request to find out the hardware address of 159.215.6.22 before sending the ICMP echo request message. Given that the ARP cache has the entry that was manually entered, it can issue the echo request immediately. The UDP/ICMP message that gets sent has a destination hardware address of *00:c0:b7:44:88:99* and an IP address destination of *159.215.6.22.* Since the link layer driver on a system will receive all packets explicitly addressed to the device's hardware address, the packet will be brought into the system. The link layer driver would then pass the packet to the first layer of the TCP/IP stack, the *network* or *IP* layer, which would then determine if the packet's destination IP address matches that of the device. At this point the special gleaning code detects that the device is unconfigured *and* has received a MAC and IP directed (unicast) for the ICMP echo request. These three factors allow the device to "glean" that it is being requested to set its IP address to that of the destination IP address of the ICMP echo request packet.
3. Once the device sets its IP address accordingly, it also goes out of the *unconfigured* state. This prevents gleaning from occurring again. At this point the device is partially configured. We still do not have the other two minimal elements, the subnet mask and the default gateway. So what have we accomplished? In this partially configured state we can begin communicating with the device as long as we are on a PC (or other host) on the same IP subnet. Following this restriction we actually alleviate the need for the subnet mask or default gateway because we are not involving off subnet communication (which requires the default gateway address). The effect of the lacking subnet mask is subtler. Usually the subnet mask is used to determine whether a target IP address is on or off subnet as described earlier. When the device has no subnet mask it just assumes all communication is on the same subnet and sends packets directly to the target addresses and not to a gateway.
4. A user can now *telnet* to or web browse to the host, depending

upon the protocols supported. Once connected, both the subnet mask and default gateway can be setup through a menu or command line method.

Gleaning is a clever way of doing a minimal configuration (IP address only) without the need for cables, prior setup, or any user installed software. The scheme only requires an ARP process that can accept manual entries and the ubiquitous PING program. In fact, other variants can be implemented that utilize telnet perhaps instead of PING. The designer of the device can choose a variety of accepted directed packets that signal the configuration.

It is important to note that in all circumstances either the link layer or network layer must have the gleaning code added to it that does the detection and configuration. This assumes that the designer of the device has access at that level directly, perhaps modifying existing stack source, or indirectly through appropriate callbacks. The latter case is not likely to exist in any off-the-shelf TCP/IP stack. Those who modify their TCP/IP stack should do so only if they fully understand how the stack works.

BOOTP

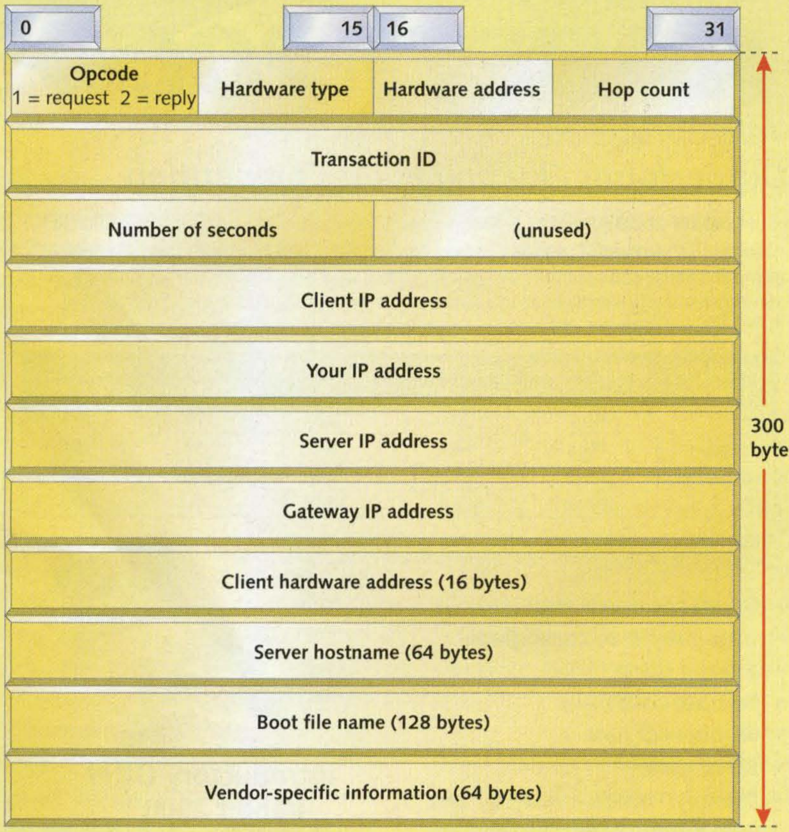
The limitations of RARP as a configuration method led TCP/IP developers to design a more flexible configuration protocol called BOOTP (Bootstrap Protocol). Again, the original intent was to enable a dynamic, over-the-network configuration method for diskless host devices. The BOOTP protocol overcomes the RARP limitation of only configuring IP addresses and is not limited by link layer broadcasts.

RFC 968 describes the BOOTP protocol in detail. The protocol is a simple client-server, request-response mechanism similar to RARP. BOOTP, however, utilizes UDP datagrams over IP. In its simplest form, a BOOTP client makes a BOOTP request with certain client information and a BOOTP server responds with the necessary configuration information for the client. The request message is sent as an IP broadcast datagram since the BOOTP server addresses are not known. The "well-known" BOOTP UDP port numbers are used by both client and server, 67 in the case of the server and 68 for the client. BOOTP servers contain a configuration file, typically called the "boottab" for *boot table*, that contains the list of preconfigured BOOTP clients. Each entry has a hardware address and one or more configuration elements such as

FIGURE 3 Boottab file entry

```
....................
Christopher_device:\
ha = 00:c0:b7:88:99:78:\
ip = 159.215.6.44:\
sm = 255.255.255.0:\
gw = 159.215.6.1
.....................
```

FIGURE 4 BOOTP message format



Opcode 1 = request 2 = reply	Hardware type	Hardware address	Hop count
Transaction ID			
Number of seconds		(unused)	
Client IP address			
Your IP address			
Server IP address			
Gateway IP address			
Client hardware address (16 bytes)			
Server hostname (64 bytes)			
Boot file name (128 bytes)			
Vendor-specific information (64 bytes)			

the IP address, subnet mask, and default gateway. In addition, it's possible to have other elements such as a host name, a boot file name and location, and so on. A typical boottab file entry is shown in Figure 3.

The BOOTP message format is given in Figure 4. The meanings of the fields are as follows:

- *Opcode* specifies request or reply (1 or 2, respectively)
- *Hardware type* indicates the requester's link hardware type (e.g., 10 for Ethernet)
- *Hardware address length* is the length of the hardware address (six for Ethernet)
- *Hop count* is used in the reply to tell how many routers (in a relay situation) the response has gone through
- *Transaction ID* is set by the client in the request and used in the server's reply to match request replies
- *Number of seconds* is the host uptime
- *Client IP address* is set by the client if it already knows its address (it may just seek a gateway)
- *Your IP address* is set by the server in the reply, to give clients IP addresses when they do not already have them
- *Server IP address* is the address of the server, set by the server
- *Gateway IP address* is the local router's IP address
- *Client hardware address* is set by client so that the server has easy access to it
- *Server hostname* is set by the server
- *Boot filename* is an optional boot file that the client can download from the server
- *Vendor-specific information* contains additional configuration options (up to 64 bytes)

Besides the standard information returned by BOOTP servers, the vendor-specific information section can contain additional configuration parameters. The first entry in this area, the "magic cookie" field, indicates whether vendor-specific options are present in the BOOTP response. When the magic cookie equals **99.130.83.99**, options are present. The vendor options are a series of entries including a one-byte *tag* followed by a length byte *n* and the actual *n* bytes of data. While the tags can be interpreted by the host, quite a few vendor-specific options have become fairly standard and are enumerated in RFC 1533. Two examples are shown in Figure 5.

BOOTP utilizes broadcast messages. This generally limits BOOTP clients and servers to be on the same IP subnet. Unlike RARP, though, BOOTP uses IP broadcasts. While these broadcasts are not forwarded across routers by default, a mechanism called a BOOTP Relay Agent can be run on routers if necessary.

BOOTP works well in those situations that configuration information is stored centrally and can be managed at the BOOTP servers. When changes are made, the clients can be rebooted to obtain their new information. The main limitation is that the hardware address of the devices must be tracked and entered, along with configuration information, into the BOOTP servers. If devices are deployed to incorrect locations, the corresponding configuration information will not work. In essence, while BOOTP can be consid-

Make Your Embedded Platform Soar! with

Treck Inc.

Real-Time Internet Protocols for Embedded Systems

We know that schedules are always tight, and that you do not have time to "port" Internet protocols into your embedded environment (let alone write them from scratch). That does not stop marketing people inside your company from asking for Internet functionality in your new product line that is rolling out in the next few weeks. Now you have someone to turn to.

High quality, high performance Internet protocols are what makes us the #1 choice for designers of embedded systems who need Internet connectivity. We design and implement our products specifically for embedded systems. That is why we are three times faster than our fastest competitor. Of course our Internet protocols do not require any porting to your specific platform. This means that you will never need to change a single line of our ANSI "C" code. You do not need an RTOS to use our Internet protocols, but if you have one, we will certainly work with it.

We are very committed to supporting you in your development effort. We know that you do not have time to wait for someone to call days later to get a support answer. That is why we provide the "finest" support in the industry. You can always count on Treck Inc. to help you every step of the way.

Our base product (Treck Real-Time TCP/IP), includes Sockets, TCP, UDP, ICMP, IP, Ethernet, ARP, SLIP and Ping.   Treck also provides support for PPP, DHCP, BootP, FTP, and Telnet. All of our Internet products arrive at your door in source code form and are always royalty free.

Call us today to see how Treck Inc. can help you!

Treck Inc.

(800)340-6648 or (513)688-0553

or visit our web page at www.treck.com

ered a dynamic, over-the-network configuration protocol, there is a static setup at the BOOTP server.

DHCP

The Dynamic Host Configuration Protocol (DHCP) is a superset of the BOOTP protocol and is defined by RFC 1541. DHCP extends the functionality of BOOTP in several areas; the vendor options area allows up to 312 bytes of configuration information and IP addresses can be "leased" from a DHCP server. The lease functionality of DHCP is the most significant addition and was designed to make address assignment to computers, especially laptops, more dynamic and automatic.

The DHCP request and response packets follow the same format as BOOTP with some DHCP-specific operation tags. We will limit the discussion here to the functional differences rather than going into detail on the message sequence and formats. The interested reader is encouraged to refer to the RFC for more detail.

The DHCP protocol allows for three different address assignment options. With *manual* assignment, a DHCP administrator can define an entry in a DHCP server configuration to respond to a specific client computer with a specific IP address configuration. This mode operates identically to BOOTP and, in fact, a DHCP server can serve BOOTP clients in this fashion.

The *automatic* address assignment mode differs in that the administrator need not preconfigure an address entry for a client. When an arbitrary client makes a request, an address is served to the client that it can use permanently from that point forward. In effect, this mode lies between a pre-configured, static assignment mode and the fully dynamic mode explained hereafter.

The *dynamic* address assignment mode introduces the concept of *leasing* IP addresses for some length of time to a DHCP client host. Specifically, a DHCP administrator configures a DHCP server to utilize one or more *IP address pools.* When a DHCP request is received, the configuration is checked to determine if the client is listed as either a static client or a previously known automatic client. If not, the server selects a "free" (currently unused) IP address from its pool. This address, along with other automatic configuration information such as subnet mask, gateways, and other utility servers, is returned to the client. Along with the address, however, is a *lease time.* The lease time tells the client how long it can use the address. Once the lease has expired, the client must make another DHCP request. Administrators can configure lease times from very short, per-

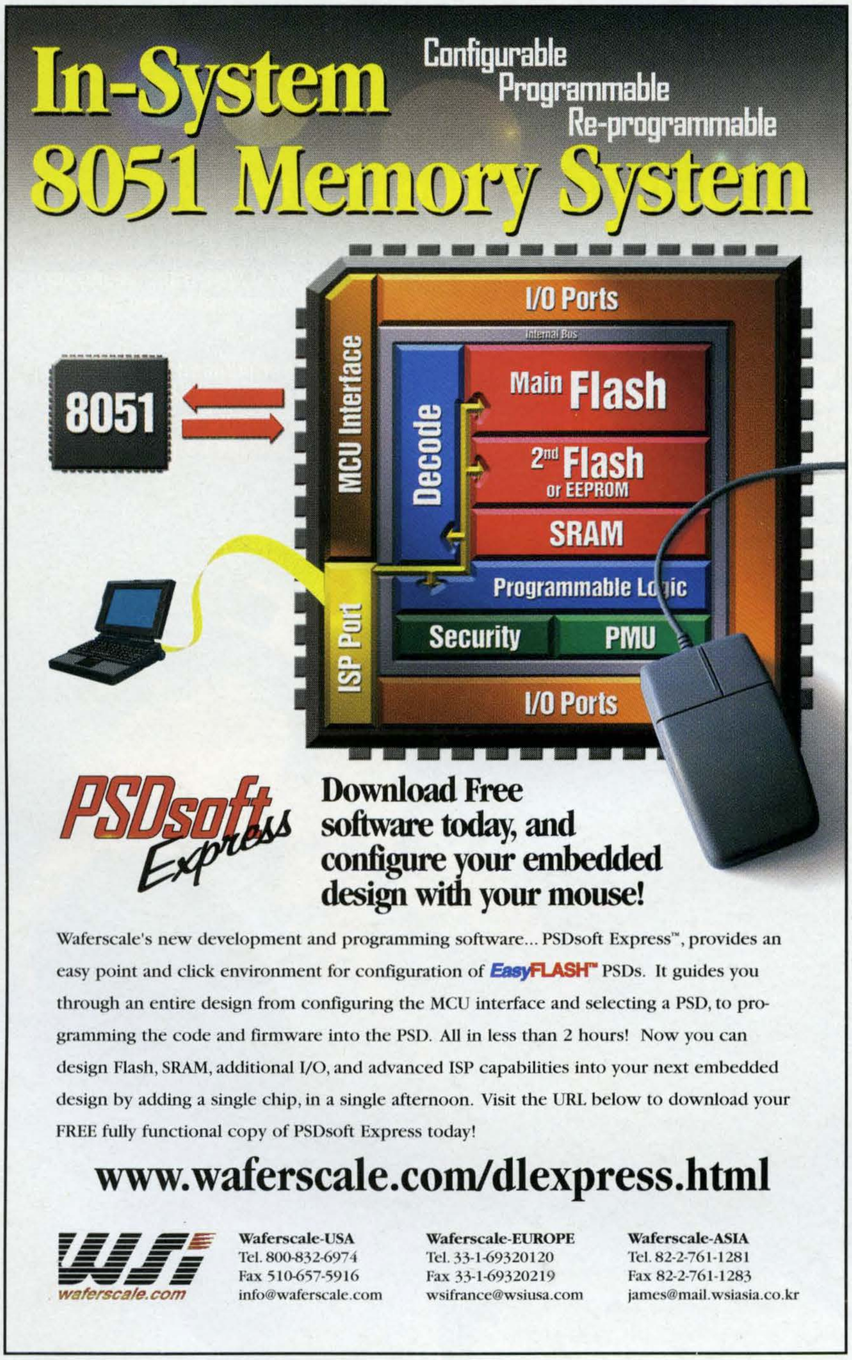
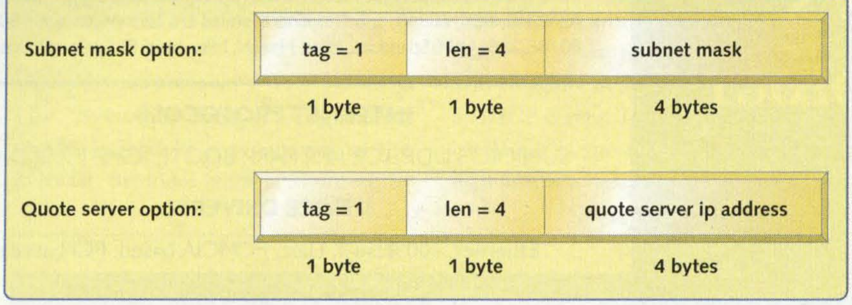


FIGURE 5 Vendor information field option formats



Subnet mask option:	tag = 1	len = 4	subnet mask
	1 byte	1 byte	4 bytes
Quote server option:	tag = 1	len = 4	quote server ip address
	1 byte	1 byte	4 bytes

haps an hour, to infinity, which is a permanent assignment.

The DHCP protocol eases computer configuration because new computers (or devices) can be added to a network and be configured and start communicating automatically. No client information needs to be set up in a configuration file.

Perhaps the most significant use of DHCP is in facilitating configuration of mobile laptop users. Without DHCP, a traveling laptop user requires reconfiguation each time the laptop moves from one network location to another. A laptop configured to use DHCP (typical for the Windows operating system) makes a DHCP request each time it boots on a new network. The local DHCP server then supplies the laptop with the exact configuration information it needs for the local network including an IP address. The lease time allows the address to be freed and go back into the available pool when the laptop user leaves.

The lease mechanism allows a small address pool of dynamic addresses to be leveraged across many potential mobile users. The pool need only be as large as the number of simultaneous DHCP users. Without the automatic DHCP scheme, administrators are forced to look for a free address, give it to a user, and configure the laptop. If a user should leave and return without manually "re-requesting" an address from the administrator, there may be situations where address duplications on the network occur. Obviously, the human administration is both time-consuming and error-prone.

Although DHCP is extremely helpful to laptop users and network administrators, its uses for address assignment in embedded network devices are more limited. Consider the difference between a laptop and an embedded network device such as the network management card in a UPS. A laptop will typically have a unique host name given to it by a user or administrator, perhaps ChristopherLaptop1. Should this laptop be installed on a Windows network, it could receive a dynamic IP address through DHCP and broadcast its host name using TCP/IP-NETBIOS (a Windows networking protocol). Since it has an IP address, TCP/IP communication is facilitated and the laptop name will appear in the computer list in Windows Explorer. While the laptop user is connected and happy, other interested parties could actually locate and use the laptop by name as well.

The UPS device has some key differences. First, when the unit is initially deployed and installed, it will not have anything configured in it except factory defaults. Secondly, as an embedded device, we assume that the device is more "server-centric," that is it provides some service to the network or its users, the use of which is obtained by connecting to the device through some TCP/IP protocol such as telnet or HTTP (for browser-based access). The first issue limits our ability to "identify" the unit uniquely out of the box. The second point emphasizes the fact that as a user, we are not operating *from* the unit as in the laptop case, but rather to the unit which requires us to know the IP address of our device. If the device booted up, requested a DHCP address, and received one from the pool, how would we know what the address is? It would be possible for the device to advertise itself using NETBIOS and show up on a Windows list. However, this solution is Windows-centric and does not address the issue of unique identification.

Returning to the example, if we had multiple UPS systems on a local network, how would we distinguish between them? Without pre-configuration (the task we wish to avoid with DHCP), the UPS might have a model number, a serial number, and a hardware address. The latter two are unique, but not easy to figure out from a distance. At a minimum we would have to read the serial number from the back of the UPS and note it and the location of the UPS for later correlation. This scheme also assumes another piece of infrastructure is in place, a DHCP-to-DNS (Domain Name System) connection. Such a connection populates a DNS server with hostname-IP address pairs for dynamically configured devices. While such systems exist they are not standard or typical.

In the end, assuming we did everything mentioned, we could communicate with the UPS by using *telnet ups1400VA_SN2366590*. The host name "ups1400VA_SN2366590" would resolve to the IP address assigned by DHCP through an automatic DNS entry population. Inelegant at best and not much work saved over BOOTP at worst. This example highlights why DHCP is rarely, if ever, used to configure network infrastructure or embedded devices. Of course, those devices with a Windows relationship, such as network printers, are a noteworthy exception. Printers tend to be more easily identified and can be more directly managed through Windows network names.

Different strokes

As should be evident, there is no one perfect or "best" configuration method. Different devices and different circumstances call for unique installation and configuration choices. As a network device provider, it is most important to understand how your customers are going to deploy your devices and provide them with the requisite configuration options. In most cases, devices should have both a static local configuration method and one of the flexible, over-the-network configuration methods such as BOOTP or RARP. The latter two enable remote configuration and reconfiguration.

Pre-assigned addresses, gleaning, and DHCP tend to be most useful in circumstances where devices are being installed locally and are unique on the local network. In these cases, it is desirable to use the quick, cable-free methods. Device identification is not a problem because many identical units are not being installed.

An additional, similar, scheme is to provide a custom program that a user can install and use on a network computer that mimics a BOOTP or RARP server. Such a program could "listen" for BOOTP or RARP requests from your devices only. When a request is received, the user can be prompted for IP configuration information. The program could then respond, again as if it were a BOOTP or RARP server, and configure the device. Several companies employ such methods today.

While some of these protocols are indeed standards covered by RFCs, their exact use is not. Combining methods such as gleaning and BOOTP require the developer to determine the relationship between the configuration methods both in terms of sequencing as well as priority. For example, if a device receives an IP address through gleaning while it is making BOOTP requests, presumably the address should be used and the BOOTP process stopped. Another scenario could be that a device boots and makes BOOTP requests for a while, but receives no response—the device had received a BOOTP response at an earlier date. Should the device "fall back" to that prior address or remain unconfigured and inaccessible?

Such questions are but a few of the many that need to be answered when implementing a robust and flexible configuration scheme for your network embedded systems. **esp**

Christopher Leidigh leads the embedded networking group at American Power Conversion, implementing SNMP and HTTP management platforms for power and environmental protection systems. He has been programming in various languages for over 14 years and has been doing embedded hardware and firmware design for eight years. He can be reached at CLeidigh@apcc.com.

Building Test Scaffolding

by Michael Aivazis

We continue our discussion of unit testing by describing the techniques necessary to build test scaffolding. Scaffolding is the extra code you must build around a class in order to test that class. Our first step in scaffolding is to write a class that contains the method test. We'll use test as the place from which we call different methods of the class we're testing.

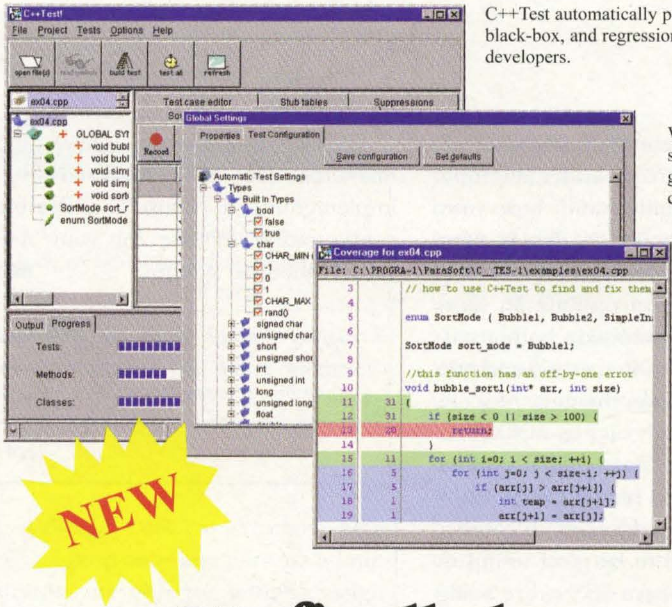
Suppose we are testing the `BankAccount` class from our previous column. This class manages a bank account; it contains methods for withdrawals and deposits, a method that reports the balance, and a constructor `BankAccount`. To see the code for the `BankAccount` class, please refer to the previous issue of this magazine.

We will call our scaffolding class `BankAccountTest`. As you can see below, we are testing to see if our class actually handles deposits and withdrawals correctly. We use our scaffolding class to make two deposits and one withdrawal, and then we check the balance:

```
#include "BankAccount.h"
#include <iostream>
using namespace std;

class BankAccountTest {
public:
     static void test(){
        BankAccount account;

        account.deposit(20);
        account.deposit(30);
        account.withdraw(7);

        if (account.balance() == 43) {
           cout << "Test passed" << endl;
        } else {
           cout << "Test failed" << endl;
        }
     }
};

int main()
{
     BankAccountTest::test();

     return 0;
}
```

After running this test, our balance comes up correctly as $43 and we read the message "Test passed." Here we have taken a standard approach to testing; we have tested the simplest functionality first and will now test how the class handles special cases, such as a balance of $0 or a negative balance. We will address these special cases in our next column.

Michael Aivazis, Ph.D., is Director of Technology at ParaSoft. You can reach him at mga@parasoft.com.



```
File: C:\PROGRA~1\ParaSoft\C__TES~1\examples\ex04.cpp
        // how to use C++Test to find and fix them

        enum SortMode ( Bubble1, Bubble2, SimpleIn

        SortMode sort_mode = Bubble1;

        //this function has an off-by-one error
        void bubble_sort1(int* arr, int size)
    31  {
    31      if (size < 0 || size > 100) (
    20          return;
            }
    11      for (int i=0; i < size; ++i) (
     5          for (int j=0; j < size-i; ++j) (
     5              if (arr[j] > arr[j+1]) (
     1                  int temp = arr[j+1];
     1                  arr[j+1] = arr[j];
```

ALEXANDER WOLFE

The Software Side of Crusoe

With considerable ballyhoo, Transmeta has finally introduced its new processor family. Here's a take on what it will mean to software developers.

After five long years in a highly secretive development cycle, Transmeta Corp. finally came out of the closet with its launch offerings this past January—two embedded processors dubbed the TM3120 and the TM5400. The chips are the first of a family of very-long-instruction-word (VLIW) processors called Crusoe.

For software developers, the salient fact is that Transmeta is positioning Crusoe as "the first family of software-based microprocessors." What does that mean in terms of writing code to run on Crusoe? Unfortunately, Transmeta has been somewhat hazy in providing answers to that question—especially since they are declining to release the Crusoe instruction set.

This article attempts to tackle—in rough form, anyway—the software development question. It's an article of faith that Transmeta will ultimately release more detailed information, but this should get the ball rolling.

Transmeta has been so closely watched because it was co-founded by former Sun Microsystems architect Dave Ditzel, who led the design of the first SPARC MPUs. In addition, one of its star employees is Linux creator Linus Torvalds. (Crusoe CPUs are optimized to run a mobile version of the Linux operating system, though they handle many other OSes as well.)

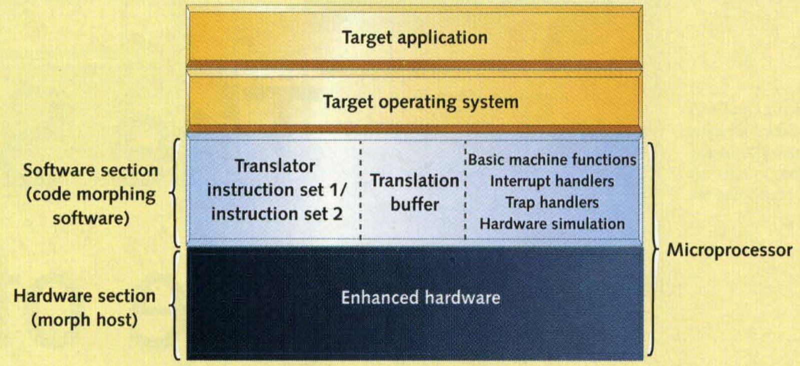
Still, it's not yet crystal clear whether the Crusoe family is as revolutionary in a "software-centric" way as Transmeta claims. Many people I've talked to in the industry say that the chips are impressive, but they make use of technology—such as VLIW computing—that already exists.

Under the hood

First off, let's summarize the Crusoe hardware itself.

The lower-end TM3120 will run at from 333MHz to 400MHz. It is equipped with a 96K L1 cache and is packaged in a 474-PIN ball grid array. IBM—Transmeta's fab partner—is already producing the device (however, only Transmeta will sell the part) in its 0.22-micron process technology on a 77mm die.

FIGURE 1 An early Transmeta patent highlights its "code morphing (i.e., translation) software"



The more powerful TM5400, which is currently sampling, runs at from 500MHz to 700MHz. It has a 128K L1 cache and a 256K L2 cache. The 73mm die is fabbed in a 0.18-micron process.

While the 3120 is aimed mainly at Internet appliances, Transmeta hopes the 5400 will find a home in a yet-to-be-developed generation of long-life subnotebook computers. Transmeta officials told me they had numerous potential customers, some of which will release products this year. However, they declined to name any of the firms.

Software translation

The linchpin of Transmeta's technology is its "code-morphing software" (CMS), which translates x86 applications into native Crusoe code that runs on the TM3120 or TM5400.

In this regard, the Crusoe chips are positioned as embedded options for the x86 applications world. Transmeta does not see developers writing native applications. However, as any embedded programmer would ask, "How are you going to optimize your code if you give it all over to a translator and lose the right to get in there and tweak it?"

The folks at Transmeta admit this is a valid question, but for now they're falling back on the answer that their CMS is so good that there's nothing to worry about. (More to the point, it's likely a trade secret that Transmeta wants to keep out of the prying hands of other companies' engineers.)

Equally important, shunting the complexity off to the software makes the Crusoe silicon itself much simpler (that is, more VLIW-like) and requires less power dissipation. (This stands in contrast to Intel's highly complex IA-64 architecture, which, it should be noted, is aimed at the high-end server market.)

Nuts and bolts

Nevertheless, concerned about what I perceived as a lack of flexibility for software developers, I was able to press someone close to Transmeta for some answers. I can't tell you the person's name, but I thought a brief summary of our conversation would shed more light on the issues at hand than anything I could paraphrase. (Besides, I'd like him or her to keep their job!)

ESP (me): If you can't throw in in-line assembly, how do you optimize? Do have to optimize your C code through the preprocessor you get from Transmeta?
Transmeta: No. Our code morphing software (CMS) does the optimization. We analyze code over a period of time and keep reanalyzing it. Then we ask, "Is there a way that we can optimize this to any greater extent?" and we do that.
ESP: That's done in Crusoe?
Transmeta: That's done in the CMS

itself, in real time [i.e., during execution].

ESP: So, in other words, while the chip is running this small block, let's say, it's ROMable?

Transmeta: It's initially ROMable but it gets extracted when the computer comes up.

ESP: It sounds like you're performing the analysis function while simultaneously taking instructions and doling them out to the [VLIW] functional units of the chip.

Transmeta: Yes, but let me be a little bit clearer. CMS lives in a ROM. It's compressed just like with a Huffman algorithm. From the time you power up, our TM3120 extracts the decompression algorithm and then decompresses the entire CMS into its own local memory. So at that point, once it decompresses itself into its entire own local memory, then it does the first x86 batch from the reset vector. So from the x86 point of view, it looks just like an x86.

ESP: So, in other words, it has to fetch the first instruction.

Transmeta: That's right. At that point it looks just like an x86 processor but with a major difference: when you start executing loops, we actually keep track of where code is executed inside real memory and inside user memory. We also keep statistics. For example, if we've been through an area 50 times, we figure we can optimize that and make it run faster. We also have different levels of optimization inside the code-morphing software.

ESP: How do you do that—with compiler switches?

Transmeta: No, this is all at run time. [Decisions are made] according to the software itself and according to what the instruction mix is, how many times it's been executed, whether it's running out of RAM or ROM, and whether it's doing any I/O cycles.

ESP: So how would I program for Crusoe?

Transmeta: It's just an x86. You would write your program on any x86 C compiler and just run it—there's nothing special.

We're expecting people to develop for our processor at the x86 level, not at its native instruction-set level. We would want them to embed it as if it were an x86 processor. We're hoping that they would embed it into a set-top box or something like that. It would have Web browser capability and because it's x86 compatible, they have access to all the plug-ins that are applicable to x86, whether they be Linux-based or Windows-based.

There's certainly much more to explore in getting ready for Crusoe. Paradoxically, perhaps some of the most valuable public documents—which explain advanced code optimization techniques—are those related to Intel's Itanium processor (see the Resources section).

By the way, the total CMS footprint is 16MB, but that includes the whole ball of wax. Your working footprint can, thankfully, be made substantially smaller. **esp**

Alexander Wolfe is managing editor for microprocessors and embedded at Electronic Engineering Times. *He holds a BE in electrical engineering from Cooper Union. He wrote assembly language code for embedded systems in the early '80s. He later co-authored* From Chips to Systems: An Introduction to Microcomputers, 2nd Edition *(Sybex, 1987). He can be reached at awolfe@cmp.com.*

Resources

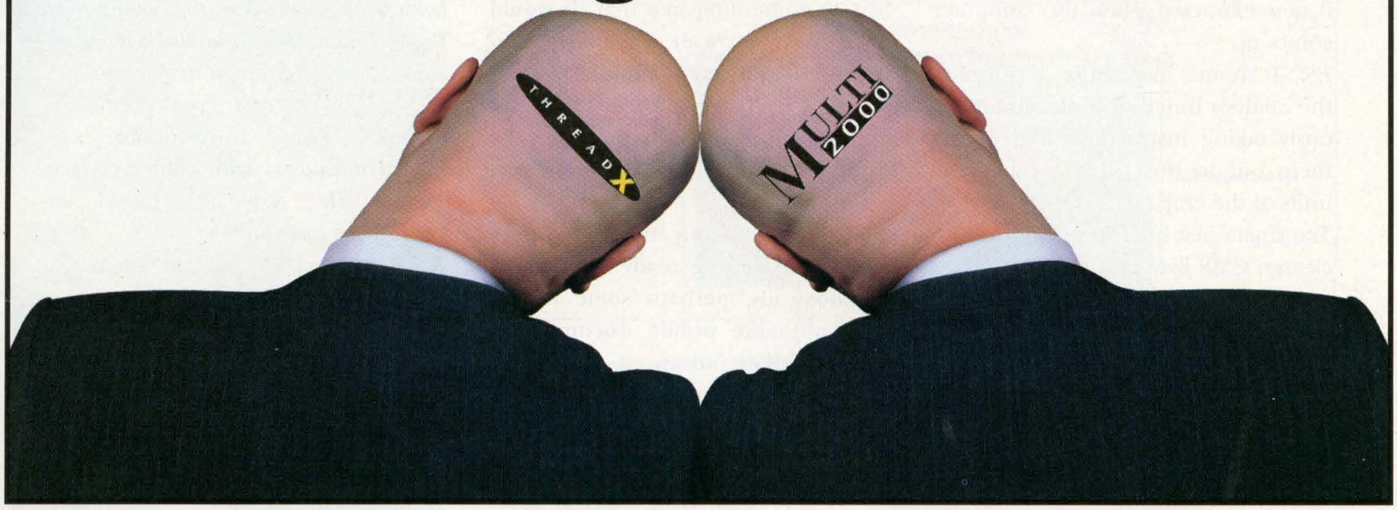
Booth, Rick. *Inner Loops: A Sourcebook for Fast 32-bit Software Development.* Reading, MA: Addison Wesley Longman, 1997.

www.transmeta.com/dev/

developer.intel.com/technology/itj/q41999/articles/art_1.htm

www.trimaran.org

We Put Our Heads Together



And Came Up With This Great ARM Solution.

ThreadX

- No Royalties
- Fastest RTOS for ARM
- Smallest memory requirements
- Complete ThreadX source code
- Preemption-threshold™
- Picokernel™ architecture
- Optimized ARM interrupt handling
- IRQ and FIQ interrupt support

MULTI 2000

- Best IDE for ARM Development
- Best ARM Debugger
- Best ARM Compiler
- Seamless ARM/Thumb interworking
- CodeBalance™ tuning
- ThreadX-aware Debugging
- Run-time Event Analyzer for ThreadX
- ThreadX C Library Integration
- Complete ARM simulation

What if the best RTOS in the market combined with the best Compiler, Debugger, and IDE? You'd get seamless integration, no hitches, no snags, and no surprises.

Well now it's happened. Express Logic's ThreadX® kernel and Green Hills Software's MULTI® 2000 IDE have come together to create the industry's most powerful development and run-time solution for deeply embedded applications.

With ThreadX you get a royalty-free, source-code RTOS, and state-of-the-art features like preemption-threshold and picokernel design. The advanced MULTI 2000 IDE gives you an Optimizing C/C++ compiler, ThreadX-aware debugging, complete ARM simulation, and the run-time EventAnalyzer for ThreadX. Together, ThreadX and MULTI 2000 are easily the best ARM solution available today.

For a free demo CD of the MULTI 2000/ThreadX solution, call Green Hills Software at 805.965.6044 or visit us on the web at www.ghs.com/armsolutions.

www.ghs.com/armsolutions

Best support for the ARM Processor

ARM6
ARM7
ARM7M
ARM7TDMI
ARM7500fe
ARM8
ARM9
ARM9E
StrongARM

Other Processors Supported:

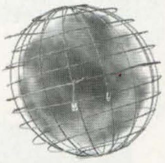
- Win32 • PowerPC • X86
- ColdFire/68K • Hitachi SH
- MIPS • NEC V8xx • SPARC
- M-Core • TriCore

From the brains of

Express Logic, Inc. and Green Hills Software, Inc.

Green Hills
SOFTWARE, INC.

ph. 805.965.6044 • sales@ghs.com • www.ghs.com/armsolutions



Embedded Internet Tools

Platform-independent application/Web server

iServer is a small, scaleable platform-independent application/Web server, written in Java. iServer serves static Web pages and generates data-driven Web pages using Java Servlets, Java Server Pages (JSP), iScript, Common Gateway Interface (CGI), and Server Side Includes (SSI). iServer also provides Servlet capabilities to any application supporting Apache Modules, ISAPI, NSAPI, CGI, or Java. iServer also features combined servlets, filters configuration files, allows Java Virtual Memory parameters to be specified in JAVA_EXE environmental variable, adds session support to the ClusterRedirector, allows configuration of additional server parameters, adds new template and session configuration variables, allows stopping/restarting server, and expands documented API. iServer is available now and licensed on a per-seat basis for $100.

Servertec
Kearny, NJ
(201) 998-1048
www.servertec.com

Software development kit

The Network MicroConverter Kit is based on the ADuC812 MicroConverter, a data acquisition system on a chip featuring an 8-channel, 12-bit ADC, two 12-bit DACs, and an integrated flash microcontroller core on a single 52-lead PQFP device. Additional hardware on the kit's PC board includes RS-232 and RS-485 line drivers/receivers that allow multi-drop serial communications to on-chip flash/EEPROM. The Network MicroConverter Kit works in conjunction with emWare's EMIT software. The development kit is available now for $99 though emWare's Web site.

Analog Devices Inc.
Norwood, MA
(800) 262-5643
www.analogdevices.com

emWare Inc.
Salt Lake City, UT
(801) 453-9300
www.emware.com

Middleware to support WAP-enable devices

Nettech Systems will deliver support for Wireless Application Protocol (WAP)-enabled devices. Nettech's middleware supports both HTML and wireless mark-up language (WML)-based access to Internet information. The middleware offers mobile services through its ExpressQ server which features store-and-forward messaging, push technology, guaranteed delivery, security, and intelligent agents which allow interfacing with Internet e-mail. Beta is available now.

Nettech Systems Inc.
Princeton, NJ
(609) 734-0300
www.nettechrf.com

Virtual machine for embedded Java apps on Linux

Embedded PERC v. 2.2 supports the Linux operating system and builds on the current suite of tools and run-time environment available with PERC v. 2.1 software, including the PERC Java-compliant virtual machine, compilers, tools, and libraries. The PERC platform also supports VxWorks, pSOSystem, Windows NT, ETS, and RTX. The Pentium and Celeron processors and PowerPC are supported as well. The Embedded PERC v. 2.2 product family also includes the ROMizer static-linking tool, the PERC native method integration tool, the QuickPERC just-in-time and flash compiler, the Embedded Toolkit component library, and PERC v. 2.2 libraries. The typical PERC virtual machine and libraries fit into 256K to 900K ROM and 64K to 128K RAM footprint. Embedded PERC v. 2.2 software is available now for licensing.

NewMonics Inc.
Lisle, IL
(630) 577-1590
www.newmonics.com

Web browser with desktop-equivalent browsing

Escape v. 4.0 Professional edition is an embedded device browser which delivers Internet browsing to the PC desktop in less than an 800K footprint. It's targeted for Internet-enabled consumer products such as digital TV set-top boxes, networked game consoles, automotive computing, Internet screenphones, and wireless handhelds. The Escape v. 4.0 Professional edition architecture provides support of HTML 4.0, forms, frames, tables, applets, images, JavaScript 1.4, SSL v.2.0/3.0, DHTML, DOM, and CSS. A free download of the desktop version of the browser is available for Windows, Unix, Linux, and Macintosh systems. The embedded version of Escape v. 4.0 Professional Edition for smart device OEMs is currently available.

Espial Group
Ottawa, Ontario, Canada
(972) 361 8080
www.espial.com

CodeWright® 6.0

The Programmer's Editing System™

Send us a WOW use of CodeWright and you and your company may be featured in one of our upcoming ads.

"In the war of editor interfaces, CodeWright is the great peacemaker."

Nicholas James Witchey, CodeWright convert since 1996
William Gustafson, CodeWright convert since 1997
Chris Greiveldinger, CodeWright convert since 1998
Steve Daily, CodeWright convert since 1997
Software Engineers, US Software
www.ussw.com

Your development environment.

CodeWright is the perfect environment for developing embedded systems. Use it as a stand-alone editor with a rich set of productivity enhancing tools or as an integration platform that enables your comand line driven or Windows based compilers and tools to be combined within a state of the of the art Windows based environment.

CodeWright is highly customizable: use your familiar keymaps, one or more of your preferred languages, customize toolbars and menus, automate your processes using macros, invoke your current tools and much more.

Armed with top quality products, superior knowledge and highly qualified engineers, US Software stays in step with the rapidly changing embedded systems field. Equipped with CodeWright, their engineers do battle with code each in their own way. Sometimes the tool makes the troop.

Download or call for an evaluation copy
www.premia.com/embedded

$299
Single User
Multi-User License available
Windows 2000/98/95/NT

Outside North America
Call (503) 641-6000
Fax (503) 641-6001
MC/Visa/Amex/P.O.'s
1-800-675-7793

premia®

the source code intelligence company™

Dan Saks

Passing Parameters by Value with const

Over the past decade, I have written numerous articles in this and other publications on various aspects of programming style, mostly as it applies to C and C++. Many of my lectures at the Embedded Systems Conferences touch upon style issues, as do the seminars I offer my clients.

One of the style issues that just keeps coming up is when to use, or not to use, `const`. Most C and C++ programmers know at least something about the const qualifier, if only because their compiler occasionally forces them to add `const` to a declaration here or there. Unfortunately, many programmers lack a clear sense for when they ought to use `const`, especially when nothing in the code compels them to do so.

Programming projects and organizations adopt coding guidelines often in the hope of helping programmers decide issues such as when to use `const`. Sensible guidelines enable programmers to work around their lack of understanding of the language and write code that is more reliable and maintainable than it would otherwise be.

Over the past few years, I've read and reviewed the coding guidelines for several different programming projects. More than once, the guideline regarding `const` echoed Scott Meyers' simple recommendation, namely, "use `const` whenever possible."[1] It's basically sound advice, but it's very broad and prone to misinterpretation.

Too many possibilities

The problem with using `const` "whenever possible" is that the set of possibilities is surprisingly large, and too many of those possibilities are meaningless.

Some programming style issues just never go away. Take, for example, when to use, or not to use, `const`.

For example, consider the declaration of the Standard C `bsearch` function:

```
void *bsearch
 (
 void const *k,
 void const *t,
   size_t n,
   size_t s,
   int (*f)
     (
     void const *fk,
     void const *fe
     )
 );
```

This function searches an array for an element that matches a key. Parameter `k` points to a copy of the key. Parameter `t` points to the initial array element in an array with `n` elements, each `s` bytes in size. Parameter `f` points to a comparison function that `bsearch` uses to compare the key against an array element. If `bsearch` finds a matching array element, it returns a pointer to that element; otherwise, it returns a null pointer.

(The declaration for `f` declares two parameters, `fk` and `fe`. They correspond to arguments that `bsearch` passes to `f`, not arguments passed to `bsearch`. Nonetheless, these parameters contribute to the type of `f`, and hence to the type of `bsearch`.)

The declaration for `bsearch` already mentions `const` in four places, all of which are meaningful. However, it doesn't use `const` at every possible opportunity. Not by a long shot.

Here's a declaration for `bsearch` that uses `const` everywhere possible:

```
void const *const bsearch
 (
 void const *const k,
 void const *const t,
  size_t const n,
  size_t const s,
  int const (* const f)
    (
    void const *const k,
    void const *const e
    )
  );
```

Yeow! That's ten more const qualifiers. Unfortunately, only one of the additional qualifiers adds any real meaning. (See if you can figure out which one before I tell you.)

Top-level cv-qualifiers, revisited

In my last column I explained that C++ ignores top-level cv-qualifiers (the keywords const and volatile) in parameter declarations when determining function signatures.[2] For example, in a function declared in C++ as:

```
void *f(void const *const p);
```

the rightmost const is a top-level cv-qualifier for parameter p. This qualifier does not contribute to the function's signature, so the function declared above has the same signature as a function declared as:

```
void *f(void const *p);
```

Let's see how this applies to bsearch.

Using const everywhere, the function's first parameter is:

```
void const *const k,
```

The rightmost const (in gray) is one I added. It is a top-level cv-qualifier, so it doesn't participate in the function's signature. The same can be said for the added qualifiers in all of the other parameter declarations:

```
void const *const t,
size_t const n,
size_t const s,
int const (* const f)
  (
  void const *const k,
  void const *const e
  )
```

Parameter f merits a little discussion. The original declaration for f was:

```
int (*f)
  (
  void const *k,
  void const *e
  )
```

so that f had type "pointer to function returning an int, with two parameters of type 'pointer to constant void'." Changing (*f) to (*const f) changes f's type to "constant pointer to function...." This const is a top-level cv-qualifier, and is not part of f's signature.

The const added to f's return type is not a top-level cv-qualifier for f itself, but rather a top-level cv-qualifier for f's return type. C++ doesn't ignore cv-qualifiers in function return types, but discards cv-qualifiers from function call expressions that return non-class types such as int. Moreover, a function's signature does not include its return type.[3] For all practical purposes, the effect of these rules is as if C++ ignored the cv-qualifier in bsearch's return type. (There are a bunch of warped, but interesting, language issues here. I expect to explore them in detail in a future column.)

The one that matters

Using const "whenever possible" adds two const qualifiers to bsearch's return type:

```
void const *const bsearch
  (
  // parameters ...
  );
```

The rightmost const is at the top-level of a non-class return type, so it's effectively ignores. On the other hand, the leftmost const is not at the top-level of anything, so here it is, the one added const that has some meaning.

Overloading with const, revisited

Let's look again at bsearch as declared in Standard C:

```
void *bsearch
  (
  void const *k,
  void const *t,
  size_t n,
  size_t s,
  int (*f)
    (
    void const *fk,
    void const *fe
    )
  );
```

Parameter t points to the initial element in the array to be searched. It has type "pointer to constant void." This means that, even though bsearch does not know the exact type of the elements in the array, it's promising to treat those elements as if they were constant.

On the other hand, bsearch returns a pointer to an element in that array as a "pointer to (non-constant) void." That is, bsearch locates an element in an array it regards as constant, but returns a pointer to an element as if the element were non-constant. This is an accident waiting to happen. It is exactly the problem I described when I explained overloading with const.[4]

The Standard C library has several other functions with this flawed behavior, including memchr, strchr, and strstr. For example, the C library declares strchr as:

```
char *strchr(char const *s, int c);
```

It returns a pointer to a (non-constant) character pointing to a character in a null-terminated string of constant characters. The Standard C++ library fixes this flaw by providing two strchr functions:

The RTOS for Internet Appliances, Systems-On-A-Chip, and all other Embedded Systems

INSPIRATION

FINAL CODE

EmProve
Virtual Application
Development and
Prototyping
(VAD-P™)
environment

RTXC™
Internet
Connected
Multitasking
Real-Time OS

BeaconSuite™
- Compilers
- Linkers & Locators
- Debuggers
- Simulators

Q.E.D.™
- JTAG Debugger
- 186 /386 ICE

The shortest distance between two points includes the ground we've already covered for you.

We have covered that ground with the most useful, flexible, scalable RTOS you can imagine for DSP, 32-, 16-, and 8-bit embedded applications.

It's what we do for a living - building it right so that you don't have to. You *could* write your own d.o.s. (dumb old scheduler), and *then* try your hand at integrating some file, graphics, and network components.

But why?

As a proven OS, **RTXC** has a pre-emptive scheduled multitasking real-time kernel with an extensive set of services and subsystems built in to assure top performance.

For example, **RTXCnet** includes a full array of networking components - from TCP/IP or PPP to web browser/server. All RTXC components are integrated to kernel resources and architected to fully utilize pre-emptive multitasking. We blow the competition's polling-based architectures away - and so will you! Just pick the networking components you need and RTXCnet arbitrates synchronization and interactions between your application, network stack and kernel.

RTXCfile32, our MS-DOS compatible file management subsystem can be used on any embedded system that uses RTXC - without regard to the target processor. We have drivers for CD ROM, floppy disks, hard disks, PCMCIA memory cards, and FLASH or RAM disks. Our drivers can be tailored to meet your most demanding file-oriented storage device needs.

Our **RTXCgraphics** subsystem endows your development effort with an existing portable graphic user interface library. Video device support is superlative, and fully supports the Rapid Application Development Standard.

Supported Processors

AMD: x86xx Family, ELAN 3/4/520C, K6
ARM: 6, 7/7T, 710, 9T, StrongArm
Hitachi: H8/300 Family
IBM: PPC 4/6
Infineon: 80C16x Family
Intel: 80x86/386/486 Real Mode,
80x86/386/486/Pentium Family Protected Mode
Mitsubishi: M16C
Motorola: DSP, StarCore, M.CORE, MPC500,
ColdFire 52xx/53xx, 68K Family, CPU 32 Family,
68HC16 Family, 68HC12 Family,
and 68HC11 Family
Philips: 80C51XA
ST Microelectronic: ST10
Texas Instruments: 320C3x/5x/20x/54x

www.embeddedpower.com

Embedded Power
CORPORATION

U.S. 281-561-9990
email: info@embeddedpower.com
Europe: +44 (0) 1256 474448
email: eurosales@embeddedpower.com

You should only use const when it has a meaningful impact. Meaningless occurrences of const are distracting clutter.

```
char const   *
   strchr(char const *s, int c);
char         *
   strchr(char       *s, int c);
```

It does the same with several other functions, but not `bsearch`.

It appears that `bsearch` should be overloaded as two functions:

```
void *bsearch
  (
  void const *k,
  void *t,
  // ... the rest ...
  );
```

and:

```
void const *bsearch
  (
  void const *k,
  void const *t,
  // ... the rest ...
  );
```

I suspect the C++ standard's failure to do this was an oversight.

Guidelines

I digress. Let's get back to my main point.

I'm pretty much of the opinion that, although you should look for opportunities to use `const` whenever possible, you should only use `const` when it has a meaningful impact. Meaningless occurrences of `const` are distracting clutter.

Top-level cv-qualifiers aren't very meaningful in function parameter types and in return types for functions returning non-class types. Therefore, my advice is:

Use `const` *whenever possible, but not at the top level in function parameter types nor at the top level in return types for functions returning non-class types.*

Some programmers object to this advice because they believe there are times that you really should declare parameters passed by value as `const`. For example, suppose you implement a `search` function with the same parameters as `bsearch`, but using a different algorithm. The `search` function's fourth parameter, `s`, is the size in bytes of each array element. That size should remain constant during any one invocation of `search`. It seems to make sense to declare that parameter as `const`, as in:

```
void *search
  (
  // ... stuff before ...
  size_t const s,
  // ... stuff after ...
  );
```

My first objection here is that this use of `const` is an implementation detail. Even without the `const`, `search` can't change the argument passed as `s` because it's passed by value. The `s` that `search` sees is just a copy of the original. Adding `const` doesn't change that.

My other objection is that, even if `const` appears as above in `search`'s declaration, such as in a header file, there's no assurance that `s` is really const inside `search`. If the function definition does not declare parameter `s` as `const`, then `search` can change `s` after all. In that case, using `const` in the declaration of `s` is the worst kind of clutter. It's an out and out lie.

If you, as the author of a function such as `bsearch`, want to protect yourself from clobbering your own copy of `s`, here's what I think you should do. You should *declare* `search` in a header with parameter `s` as non-`const`:

```
/* search.h */

void *search
  ( ... , size_t s, ... );
```

but define `search` in a source file with parameter `s` as `const`:

```
#include "search.h"

void *search
  ( ... , size_t const s, ... )
  {
  // ...
  }
```

Once again, the declaration and the definition have the same signature, so calls to the function declared in the header will invoke the function defined in the source file.

I think this style represents a pretty good compromise. Unfortunately for you C programmers out there, this technique works only in C++. C does not employ the concept of function signatures. It simply insists that every declaration for a given function be the same, cv-qualifiers and all, everywhere in the program. **esp**

Dan Saks is the president of Saks & Associates, a C/C++ training and consulting company. He is also a contributing editor for the C/C++ Users Journal. *He served for many years as secretary of the C++ standards committee and remains an active member. With Thomas Plum, he wrote* C++ Programming Guidelines. *You can write to him at dsaks@wittenberg.edu.*

References

1. Meyers, Scott. *Effective C++, 2nd. ed.* Reading, MA: Addison-Wesley, 1999.
2. Saks, Dan, "Top-Level cv-Qualifiers in Function Parameters," *Embedded Systems Programming*, February 2000, p. 63.
3. Saks, Dan, "Function Signatures and Name Mangling," *Embedded Systems Programming*, August 1999, page 79.
4. Saks, Dan, "Overloading with const," *Embedded Systems Programming*, December 1999, page 81.

AVR®
60
www.iar.com

JOSEPH LEMIEUX

OSEK/VDX Network Manager and Implementation Language

The OSEK specification covers three subsystems commonly found in automotive applications: operating systems, communications, and network management. This article completes a discussion of all three.

Last month I introduced the first two substandards of the OSEK/VDX standard, the operating system and the communication standards ("The OSEK/VDX Standard: Operating System and Communication," March 2000, p. 90). This month, I will provide an overview of the final substandard, the network management (NM) standard. I will also provide an overview of the OSEK Implementation Language (OIL), which is the specified way of defining an OSEK OS implementation.

Network management

The network management specification provides a definition of a system intended to manage the nodes on a network and provide information to the application on the state of health of each node. Figure 1 shows a complete implementation with the NM included. Network management is accomplished using one of two methods: direct or indirect. The NM component of the OSEK implementation communicates with the application,

The network management specification provides a definition of a system intended to manage the nodes on a network and provide information to the application on the state of health of each node.

the OS, and the COM through the standardized API interfaces described in each specification.

All nodes on the bus must incorporate an NM implementation using the same method of network management. This requires careful coordination of an entire system that includes electronic control units (ECUs) designed by multiple manufacturers located worldwide. Consequently, while the operating system and communication standards can be implemented on only a few ECUs in a system, the NM requires implementation on all ECUs simultaneously.

This article will provide an overview of the operation of the network management component. The actual specification contains many state charts and flow charts detailing the actual working of the NM and should be studied before implementing this system. See the resources section at the end of the article for details on obtaining the actual specification.

Direct network management

Direct network management is similar to token ring. The network is defined as a logical sequence of nodes, with each node having a logical number such that the successor of a node is the node with the next highest logical number. If no nodes exist with a higher logical number, then the lowest logical number is considered the successor.

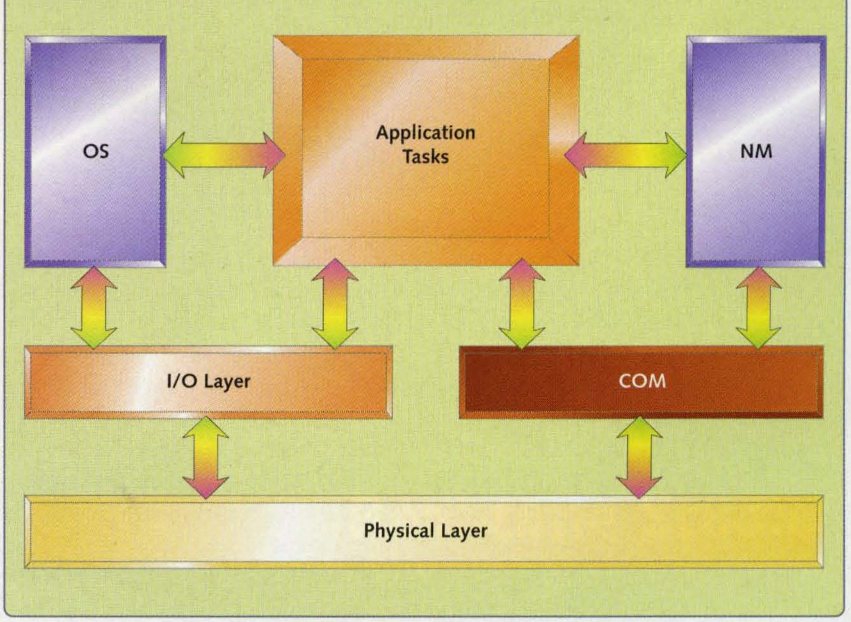
Figure 2 shows a simple network with four logical nodes that will be used extensively for illustration purposes. For simplicity, the nodes are numbered from 0 to 3. However, there is no requirement for the numbering to be sequential. In this figure, I purposefully did not number the ECUs in the same order as the physical layout. This illustrates that the physical layout of the network has no relation to the logical layout.

Configuration states. Each node determines the configuration of the network based upon the reception of *ring* and *alive* messages discussed later. The states of all nodes on the network, including its own, can be determined from these two messages. This current configuration is used to determine the logical successor and report the configuration to the application so necessary action can be taken if a node is absent.

Each external node can be either present or absent. A node is considered present if an *alive* message is received from it or if a *ring* message is sent by it. The node also determines its own state based on the ability to send messages. If a message is successfully sent, then the node considers itself not mute and participates in the ring. If it could not send out a message, then it considers itself mute. Success is determined by the data link layer of the communication portion of the OSEK system. In this state, the node periodically sends *limp home* messages in case other nodes can still receive the transmissions.

Message structure. Implementation of direct network management is performed using encoded messages broadcast over the network to all nodes. To implement the token passing algorithm, each message includes three distinct fields—the Source Logical Node ID, the Destination Logical Node ID, and an opcode. Eight different opcodes are defined—four ring opcodes, two alive opcodes, and two limp home opcodes. The possible opcodes are shown in Table 1. These fields must be encoded into a reserved field or

FIGURE 1 An application using a complete OSEK implementation



BEN FISHMAN

Ring messages are the primary means of communication and monitoring of the network.

fields of the bus message. In the case of extended controller area network (CAN), this can be easily encoded in the 29-bit header, or the information can be encoded partially in an 11-bit header and the rest in the first few data fields.

FIGURE 2 Simple network using direct network management

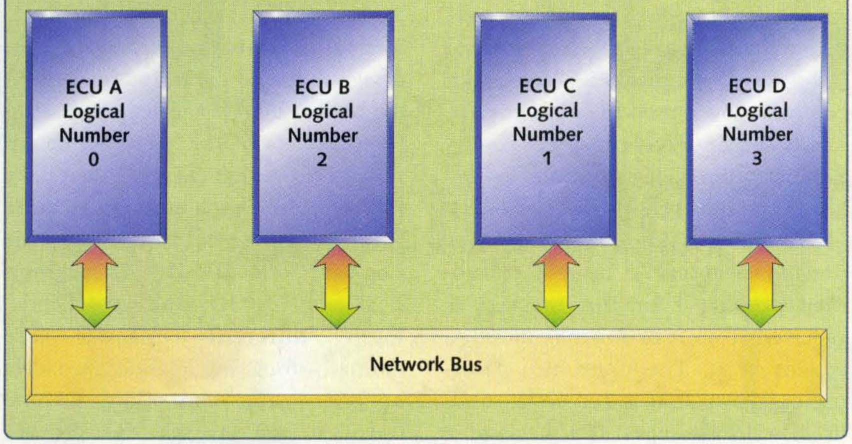


TABLE 1 Direct network management opcodes

Type	Opcode	Bus Sleep Request	Bus Sleep Acknowledge
Ring	0	No	No
Ring	1	Yes	No
Ring	2	Yes	Yes
Ring	3	No	Yes
Alive	4	Yes	Don't care
Alive	5	No	Don't care
Limp Home	6	Yes	Don't care
Limp Home	7	No	Don't care

The opcodes shown in Table 1 are encoded as 0 through 7. However, a specific requirement for encoding of the opcode is not specified. How they are encoded into the actual network message is dependent upon the network structure (for example, J1850, CAN, and so on). Opcode numbers 2 and 3 in the table are defined as separate opcodes. However, the Bus Sleep Request is ignored when the Bus Sleep Acknowledge is set. The usage of the Bus Sleep parameters is discussed later.

Alive messages are intended to register the node to the network. When an alive message is received, each node determines if the new node is its logical successor and uses this information for the next ring message. The new node monitors the current messages to determine its logical successor. When it receives a ring message, it passes the ring message to its logical successor. In general, the actual logical successor will be skipped the first time. Recovery from a skipped message is described later.

Ring messages are the primary means of communication and monitoring of the network. When a logical node receives a ring message, the NM passes the message to the application and starts an alarm. Notification of the application is performed through activation of an OS task or setting an OS event. The application can then process the message and change the data prior to transmission. When the alarm expires, the ring message is sent to the logical successor of the node.

Finally, limp home messages are sent by a node which has lost the ability to receive messages. Since the node doesn't know if its transmitted messages can be received by the other nodes on the network, a special limp home message is transmitted periodically so other nodes can use the information that it contains.

Normal operation. The operation of the network under direct network management is best illustrated with an example. Figure 3 shows a sequence of steps using the network in Figure 2, which is assumed to be in an automobile. Node 0 and Node 2 are connected to the battery and put themselves to sleep to save power. Node 3 is connected to the accessory position of the ignition key (that is, Radio). Node 1 is only connected to the Run and Start positions.

The sequence of operation is as follows:

- The battery is connected and alive messages are sent in step 1. These messages are received by all live nodes and registered
- Stable operation begins in 2 and 3 with ring messages being sent and received by all live nodes
- Node 3 comes alive in step 4 and sends the alive message. The other live nodes recognize the new node and update their successor nodes as necessary
- Node 3 recognizes Node 0 and Node 2 in steps 5 and 6 based upon the broadcast ring message. It now knows the complete configuration

Rational Rose for real-time?

The result could be painful.

More and more development teams agree Real-time Studio® is the better system and software modeling solution.

It makes perfect sense. If you're developing real-time embedded systems, you need a software modeling solution that can support your entire team and will actually fit your existing process. Make no mistake — reach for Real-time Studio. This is the one and only software modeling solution with an object-based repository enabling complete, up-to-date, shared views of your system. Finally, your entire development team can communicate and collaborate — from start to product delivery. Developed exclusively for real-time systems, Real-time Studio is non-intrusive, flexible and features automatic code generation for C, C++, and Java. Today, it's the proven solution for hundreds of developers and development teams. And the list keeps growing. Get started on the fastest path to the right product. Visit **www.artisansw.com/real** for more information and a demo of Real-time Studio, plus you can register for our upcoming Real-time UML seminar.

ARTiSAN SOFTWARE

Real-time Studio®
www.artisansw.com/real

No developer was actually harmed or risked injury in the making of this ad.
Rational Rose is a registered trademark of Rational Software Corporation.
Real-time Studio is a registered trademark of Artisan Software Tools, Inc.

At certain times in the operation of the system, it may become necessary for the bus to go to sleep.

FIGURE 3 Network messages during direct NM operation

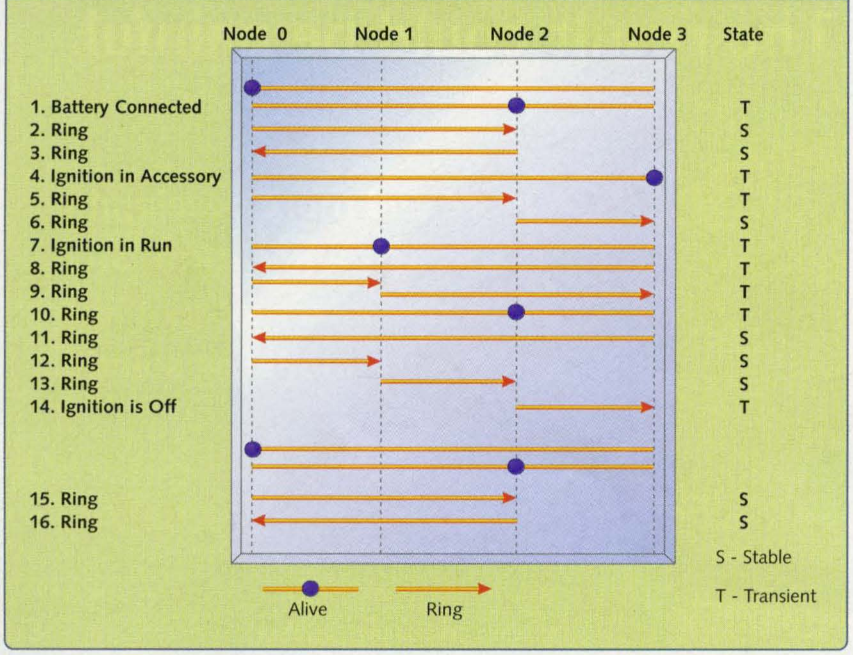
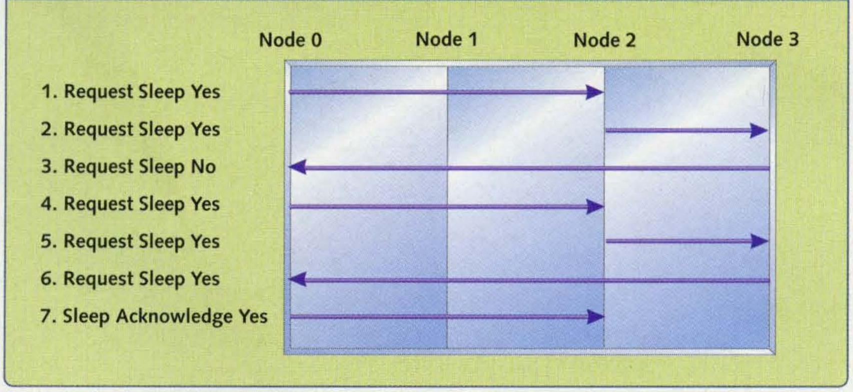


FIGURE4 Sequence of ring message in preparation for sleep



of the network and stable communication begins

- Node 1 comes alive in step 7 and sends an alive message. The other live nodes recognize the message
- The ring messages in steps 8 and 9 identify Nodes 3 and 0 to the new Node 1. Node 1 thinks its successor is Node 3
- Node 1 gets the ring message in step 9 and passes the ring message to Node 3 in step 10. Node 2 recognizes that it has been skipped and sends a "don't forget about me!" alive message
- Stable operation continues in steps 11 through 13
- In Step 14, the Ignition switch is switched to off and then the ring message is sent to Node 3. Node 3 is no longer alive and does not respond within a predefined period. Nodes 0 and 2 identify the failure and send alive messages to restart the communication
- Stable operation continues in steps 15 and 16

Limp home operation. When a node detects that it is mute—that is, it cannot determine if messages that are transmitted are received by the other nodes—it enters Limp Home Mode and begins sending periodic limp home messages. The intention is to provide information to other nodes if they can still receive the messages.

Bus sleep. At certain times in the operation of the system, it may become necessary for the bus to go to sleep. In the example in Normal Operation above, Nodes 0 and 2 will send messages forever when the Ignition Switch is Off. However, it is preferred that the nodes go to sleep to preserve power. In order to accomplish this, the Bus must first be put to sleep. This is accomplished by using the ring messages with Bus Sleep Request and Bus Sleep Acknowledge.

When a node desires to go to sleep, it sends a ring message with Bus Sleep Request Yes and Bus Sleep Acknowledge No. If all nodes send ring messages with Bus Sleep Request Yes and Bus Sleep Acknowledge No, then the first node will send its next ring message with Bus Sleep Acknowledge Yes. If any node responds with a ring message with Bus Sleep Request and Bus Sleep Acknowledge No, then the original node will send a Bus Sleep Request Yes and Bus Sleep Acknowledge No the next time it receives the ring message.

As illustration, take the example with the Ignition Switch in Accessory. After a long period of time (say 30 minutes), the system may want to shut down to conserve power. A sequence of ring messages as shown in Figure 4 may occur.

In this sequence, Node 3 is not

Aim to Deliver the Best Product on the Market?

So Do We. In Fact We Already Have.

The best development environment anywhere ASPEX is a fully integrated, completely open development environment offering you the tools to develop, integrate and optimize embedded software for today's complex products. If you've found other development environments slow, buggy and prone to crashes, you'll find ASPEX just the opposite.

Superior debugging capabilities All this multiprocessing and integration vastly increases the complexity of software debugging. But ASPEX can handle it. From a single event-driven debugger, ASPEX lets you simultaneously debug multiple target systems whether the processors are all the same or of different architecture.

A toolkit for today Mixed processors, multiple processors, systems-on-chips. Intensely increased integration. That's today's embedded software reality and ASPEX was designed from the inside out to seamlessly support it all. Make it smaller, make it cheaper, make it faster, make it better. ASPEX can make it happen.

Easy to learn, easy to use ASPEX provides a uniform, graphical interface that tightly integrates all tools and lets you manage projects graphically instead of learning cryptic command strings. Our unique Extended Target Visibility feature actually lets you see more target resources. Our Trace and Analysis feature helps you track down hard to find problems.

ASPEX supports ARM, Analog Devices, DSP Group, Intel Strong ARM and Texas Instruments. Let ASPEX give you the tools you need to meet the challenges you face. The tools to develop, integrate and optimize embedded software for today's increasingly complex products. We're experts in embedded development tools and it shows in every aspect of ASPEX. Test drive ASPEX for yourself and experience features our competition hasn't even dreamed of. Download your free, fully functional evaluation copy of ASPEX at www.allant.com.

ALLANT

Allant Software Corporation
1280 Civic Drive, Suite 206, Walnut Creek, CA 94596 • 925.944.9690 • fax 925.944.9612 • email info@allant.com • www.allant.com

[Indirect network management] operates by monitoring periodic messages from the nodes on the network and providing this information to the application.

ready to go to sleep the first time that it receives the Request Sleep Ring message. It replies with a normal ring message and Nodes 0 and 2 recognize this rejection of sleep. The next time the request arrives, Node 3 is ready. Node 0, since it is the first node to see that all nodes have requested sleep, will send the Acknowledge Sleep message. This is the last normal operation message to be sent. There may still be Limp Home messages being sent, but these are ignored in determining if sleep should occur.

TABLE 2 Direct Network Management timing

Time period	Description
T_{Typ}	Typical time for a node to wait from reception of the ring message to transmission to its sucessor.
T_{Max}	Maximum time to wait until a given node recognizes that the recipient of the last Ring message sent is absent.
T_{Error}	Time between transmissions of limp home messages when a node considers itself mute.
$T_{WaitBusSleep}$	Time to wait for bus traffic to settle prior to entering Bus Sleep mode. This period should be valid network-wide.

FIGURE 5 Simple Network using Indirect Network Management

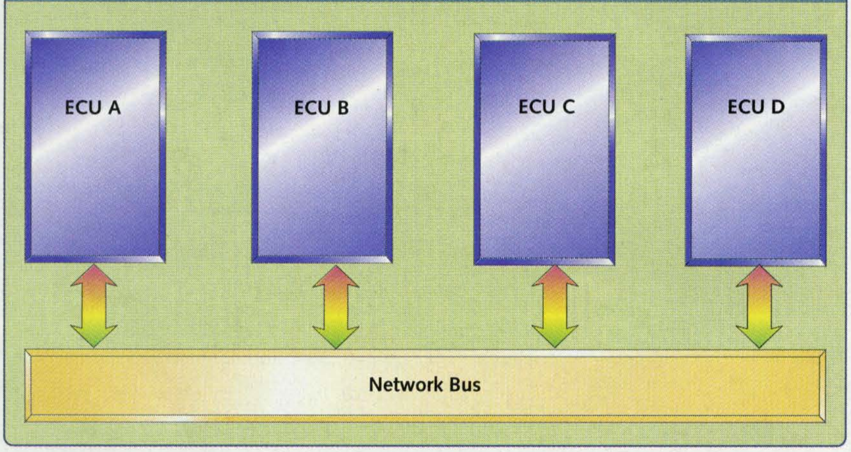


TABLE 3 States of nodes in Indirect Configuration Management

	Standard State		Extended State	
Type	Emitter	Receiver	Emitter	Receiver
Normal	Node is not mute	Node is present	Node is not mute statically	Node is present statically
Failure	Node is mute	Node is absent	Node is mute statically	Node is absent statically

After a Bus Sleep Acknowledge ring message is received, all nodes wait a predetermined time before they shut down operation. This is to allow messages that are presently queued by a node to be sent.

Timing. Direct network management defines four time periods as shown in Table 2. These time periods are implemented using OS alarms and are determined according to the following rules:

- The longest T_{Typ} in the network should be much less than the shortest T_{Max}. This should also take into account all tolerances
- T_{Error} does not have any specific requirement for duration. However, this should be relatively long compared to T_{Typ}. The standard suggests a useful value of 10 times T_{Typ}
- $T_{WaitBusSleep}$ must be greater than or equal to T_{Error}

Indirect network management

Indirect network management is much simpler to implement in each node. It operates by monitoring periodic messages from the nodes on the network and providing this information to the application. Figure 5 shows the network that will be used to discuss indirect network management.

In this network, the address or logical number of the individual ECU is not required. The periodic message that is transmitted is the only information that is needed. This message may or may not contain address information. A limitation of indirect network management is that nodes that do not transmit periodic messages can't be monitored. To remedy this, a "state of health" periodic message may be defined for monitoring only.

Configuration states. Indirect network management defines the current state of the network configuration based on the emitter and receiver capability of the node. In addition, extended states can be defined. This can best be illustrated in Table 3.

The difference between extended and standard states is basically the number of times that the failure condition occurs. In standard states, as soon as the conditions exist to move

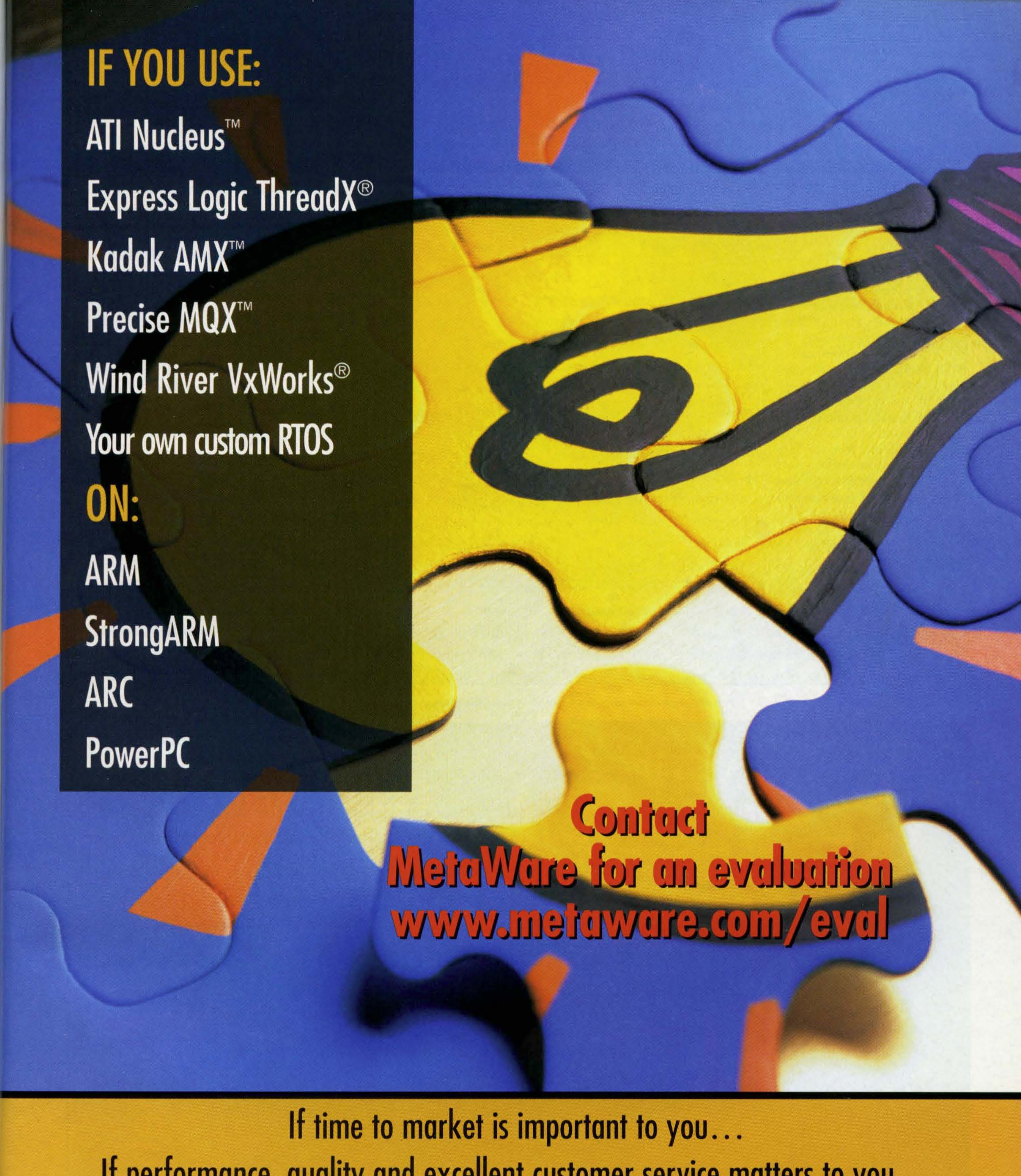
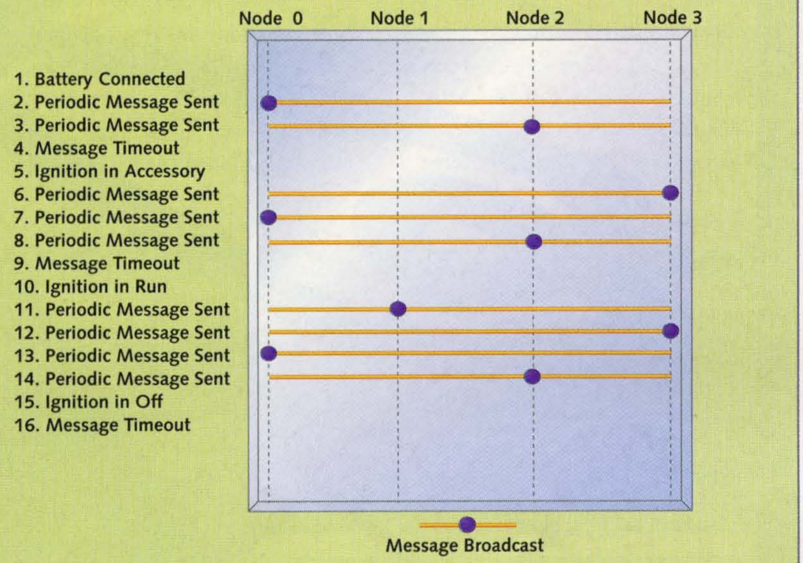
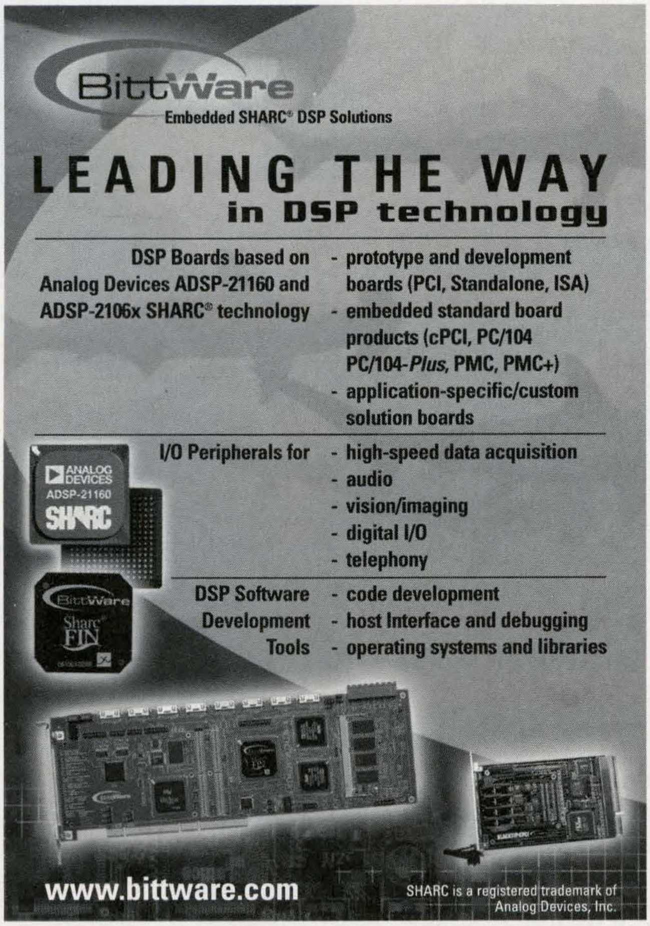


FIGURE 6 Network Messages during Indirect NM



from Normal to Failure or vice versa, the move occurs. In extended states, a counter is incremented whenever a failure condition occurs. When the counter reaches a threshold, the state is changed to failure. The counter is decremented whenever the conditions for normal state occur. The counter is limited to not exceed the threshold, thereby moving to the normal state from the failure state as soon as a valid normal condition occurs.

Normal operation. Monitoring of nodes is performed using three mechanisms—transmission, reception, and status signal. Transmission monitoring is performed using acknowledgements to a transmitted message by the emitter. If the message is not acknowledged, the emitter is considered mute. Reception monitoring may use the timeout capabilities of



the COM. If a periodic received message is not received within a time period and the alarm expires, then the node is considered absent by the receiver. Finally, status signal monitoring uses the data link layer status to indicate a problem with the bus, such as bus off.

In Limp Home Mode, the network management continues to monitor the bus for received messages.

Figure 6 illustrates the same system described under direct network management, but operating under indirect network management. The sequence of messages used to determine standard states is as follows. For extended states, counters need to be incremented and decremented before a node is considered absent.

- The Battery is connected in step 1 and Nodes 0 and 2 come alive. By default, these nodes believe that all four nodes are present. They commence sending their periodic messages
- In step 4, the monitoring alarm expires and Nodes 0 and 2 recognize that Nodes 1 and 3 are absent
- In step 5, the ignition switch is moved to Accessory and Node 3 comes alive. Node 3 assumes that all four nodes are present. When it transmits its first message, Nodes 0 and 2 recognize that it is present. Steps 6 through 8 show normal operation
- In step 9, Node 3's alarm expires and it recognizes that Node 1 is absent
- In step 10, the ignition switch is moved to Run and Node 1 comes alive. It assumes that all nodes are present. When Node 1 transmits its periodic message in step 11, Nodes 0, 2, and 3 recognize that it is present. Normal periodic transmission then occurs
- In step 15, the ignition switch is moved to off
- In step 16, the message timeout expires in Nodes 0 and 2. At this time, these nodes recognize that Nodes 1 and 3 are absent

Limp home operation. In Limp Home Mode, the network management continues to monitor the bus for received messages. It's possible that the ECU may be unable to transmit messages, but may still be able to receive messages. Consequently, NM continues to monitor the configuration, but assumes that it is mute. Since messages do not flow through the NM as they do in direct network

> **The OSEK Implementation Language (OIL) specification defines a standard language to describe the objects defined in the OSEK Operating System specification.**

management, the application may continue to attempt to transmit messages. If a message is successfully transmitted, then NM returns to normal operation.

Bus sleep. Unlike direct network management, indirect network management does not define a specific method of putting the bus to sleep. When the conditions exist for entering sleep, it is up to each node to enter the waiting for bus sleep condition. When a timer expires, the NM enters the Bus Sleep state.

The specification does allow a master-slave type network in which the master determines when the bus sleep condition exists. It then commands the slaves to go to sleep via a bit in the message or an explicit sleep command message.

Timing. Only two timers are defined in indirect network management—the message timeout and the wait to sleep timeout. However, there are two types of message timeouts allowed—global and per message timeouts.

The global timeout checks periodically if any messages have been successfully sent and if all monitored messages have been successfully received at least once since the last timeout. This defines a series of windows in which to check the messages.

The per message timeout uses the COM deadline monitoring services where each monitored message is defined with an alarm. If any alarm expires, the NM is immediately notified and action is taken to address the alarm.

OIL

The OSEK Implementation Language (OIL) specification defines a standard language to describe the objects defined in the OSEK Operating System specification. The goal is to enable software to be highly portable between processors. Unfortunately, the Communication specification (COM) and Network Management (NM) specification are not included in this specification. Consequently, switching an application from one OSEK implementation to a different implementation on another processor still requires some custom configuration. If the networks differ, then this is understandable. However, if both implementations use a CAN network, then there should be some way to standardize the definition of the COM and NM systems. Since these

TABLE 4 Object and attributes defined in OIL

Object	Attribute	Description
CPU	All objects	Wrapper defined for each CPU in an application. Multiple CPUs are allowed in a definition. The attributes of the CPU are all the objects defined for that CPU.
OS	STATUS	Status returned from interface calls. Either STANDARD or EXTENDED.
	STARTUPHOOK	Existence of hook routine, either TRUE or FALSE.
	ERRORHOOK	Existence of hook routine, either TRUE or FALSE.
	SHUTDOWNHOOK	Existence of hook routine, either TRUE or FALSE.
	PRETASKHOOK	Existence of hook routine, either TRUE or FALSE.
	POSTTASKHOOK	Existence of hook routine, either TRUE or FALSE.
APPMODE	None	Available application mode. At least one must be defined.
TASK	PRIORITY	32-bit value representing the priority of a task. Priority 0 is lowest.
	SCHEDULE	Preemptiveness of a task—either NON or FULL.
	ACTIVATION	32-bit number representing the maximum number of queued requests to activate the task.
	AUTOSTART	Defines whether the task is automatically started upon system startup. Either TRUE or FALSE.
	RESOURCE	Optional attribute that defines which resources are accessed by the task. Multiple resources may be defined.
	EVENT	Optional attribute that defines which events the task may react upon. Multiple events may be defined. If omitted, the task is a basic task.
COUNTER	MAXALLOWEDVALUE	32-bit value that represents the maximum allowed value of the counter.
	TICKSPERBASE	32-bit value representing the number of ticks required for the counter to reach a specific unit. The interpretation of this attribute is specific to the implementation.
	MINCYCLE	The minimum number of counter ticks allowed for a cyclic alarm using this counter.
ALARM	COUNTER	The counter defined above that is assigned to the alarm.
	ACTION	The action to be taken when the alarm expires, either ACTIVATETASK or SETEVENT. ACTIVATETASK requires the name of the task, and SETEVENT requires the name of the task and the event.
RESOURCE	None	Defines a resource that is accessed by a task in the application.
EVENT	MASK	Defines an event and a 64-bit mask with one bit set that corresponds to the event. If the mask is AUTO, the bit is chosen automatically by the OS.
ISR	CATEGORY	Defines the ISR and the category from 1 to 3. The name of the ISR must be the same name as the function in the application code.

Introducing...

The OSEK Implementation Language is a powerful language to allow applications to be easily ported to different microcontrollers and to OSEK systems from different vendors.

specifications are undergoing continuous improvement, this may be possible in future versions. OIL specification 2.1 was just released in June 1999. OIL v. 2.0 was a recommended practice, whereas v. 2.1 is required for OSEK compliance.

The OIL defines 12 objects. However, only nine are detailed in the specification. These nine are CPU, OS, APPMODE, ISR, RESOURCE, TASK, COUNTER, EVENT, and ALARM. The other three are MESSAGE, COM, and NM, which are out of the scope of the OS specification and are not defined in OIL. However, individual implementations can, and should, define these objects in any format desired.

Usage of OIL entails creating an application definition file that contains the list of objects defined for the application. This can be manually generated or automatically generated by a user interface. The format of the OIL file contains two sections—the implementation-specific and the application-specific sections. The implementation specific section is usually defined by the provider of the OS. The OIL specification contains a good example of a complete file and is available on the Web site referred to in the Resources section.

Each object has the basic format:

```
OBJECT <name> {attributes};
```

The definition is similar to the C programming conventions of a structure typedef, where the list of attributes are separated by semicolons and can appear on multiple lines. The list of attributes may not be used for some objects. Table 4 summarizes the objects and attributes defined in OIL.

NM/OIL summary

Direct network management defines a powerful system to control messages on a bus. The ability to understand the configuration of the system rapidly and to identify a failed node is useful. The token ring type system minimizes the possibility of crashes on the bus and may be useful on a bus type that is intolerant to these crashes.

The drawback of this type of network management is the large overhead in each node on the network and the slow response to an update of data in one node. The more nodes that exist, the greater the latency between the change in data and the actual transmission of that data on the network. In addition, if a node has more data to transmit than allowed in a network ring message (for example, eight bytes in CAN), then the latency doubles since the node has to split the data and wait until the next ring message is received. The specification is not clear on the handling of data that is transmitted in the ring message—is the data only consumed by the recipient of the ring message or can all nodes use this data? For efficiency purposes, it would be expected that since all nodes receive the ring message, they can use the data.

Direct network management should be reserved for applications in which only a few nodes exist with minimal data to transmit and where the nodes can tolerate long latencies between data updates. If rapid response is required, indirect network management should be used. My experience in the American automotive industry indicates that most, if not all, applications today could not tolerate the long latencies that Direct Network Management creates. However, with the rapid integration of the Internet into vehicles, this may be changing.

The OSEK Implementation Language is a powerful language to allow applications to be easily ported to different microcontrollers and to OSEK systems from different vendors. The major limitation at this time is the lack of a standard definition for the Communication and Network Management portions of the specification. I understand that this is a monumental task considering the conflicting network definitions that exist today. However, I encourage the OSEK committees to continue working toward a level of definition that will grow to encompass the major network definitions (for example, CAN, J1850, J1939, and so on). Keep watching the OSEK Web site for updates. **esp**

Joseph Lemieux is an senior applied specialist with EDS Embedded Solutions in Troy, MI. He holds an MSEE from the University of Michigan and has been writing software for embedded systems in the automotive industry for over 17 years. During this time, Joe has used both homegrown and commercial operating systems. He can be reached by e-mail at joe.lemieux@eds.com.

Resources

The OSEK/VDX committee has published five documents that provide greater detail of the standard. They are available at the OSEK/VDX Web site for free downloading: *www-iiit.etec.uni-karlsruhe.de/~osek/*

OSEK/VDX: Operating System, Version 2.0, revision 1, October 15, 1997

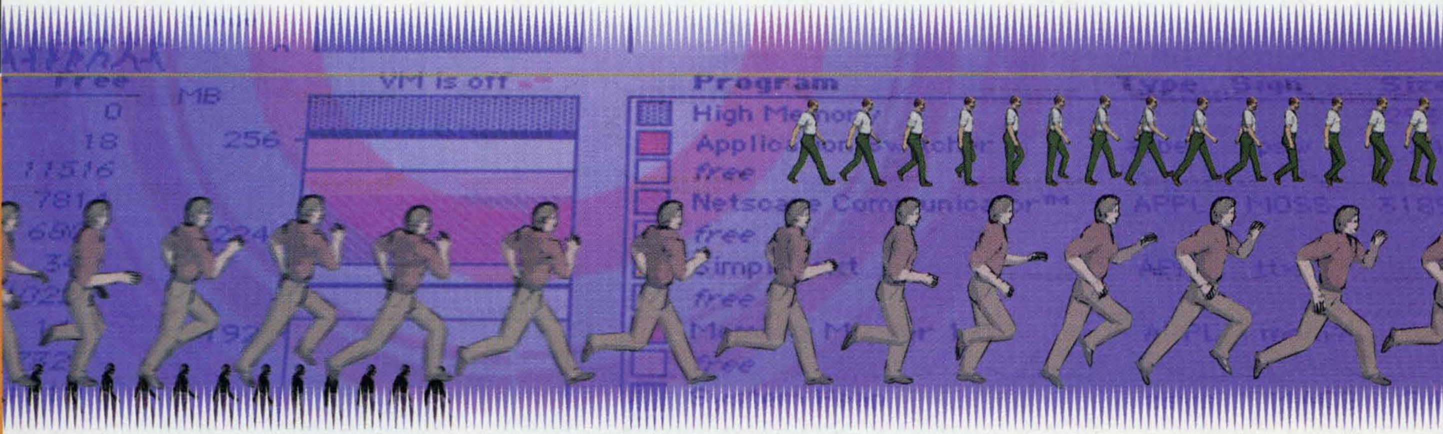
OSEK/VDX: Communication, Version 2.1, revision 1, June 17, 1998

OSEK/VDX: Network Management—Concept and Application Programming Interface, Version 2.50, May 31, 1998

OSEK/VDX: System Generation—OIL: OSEK Implementation Language, Version 2.1, June 30, 1999

OSEK/VDX: Overall Glossary, Version 2.0, September 23, 1997

NIALL MURPHY



Safe Memory Utilization

Whether you're using only static memory, a simple stack, or dynamic allocation on a heap, you have to proceed cautiously. Embedded programmers cannot afford to ignore the risks inherent in memory utilization.

Every program uses random access memory (RAM), but the ways in which that memory is divided among the needy parts of the system varies widely. This article surveys the options available in hopes that the reader will be better equipped to choose an approach for a given project.

The mechanisms include statically allocating all memory, using one or more stacks, and using a heap. We will examine how the heap implementation can impact fragmentation and real-time performance.

Static allocation

If all memory is allocated statically, then exactly how each byte of RAM will be used during the running of the program can be established at compile time. The advantage of this in embedded systems is that the whole issue of memory-related bugs—due to leaks, failures, and dangling pointers—simply does not exist. Many compilers for 8-bit processors such as the 8051 or PIC are designed to perform static allocation. All data is either global, file static or function static, or local to a function. The global and static data is allocated in a fixed location, since it must remain valid for the life of the program.

The local data is stored in a block set aside for each function. This means that if a function has a local variable *x*, then *x* is stored in the same place for every invocation of that function. When the function is not running, that location is usually not used. This approach is used in C compilers when the hardware is not capable of providing suitable support for a stack. Figure 1 shows the memory organization with no heap and no stack, just globals and one static block per function

This approach prohibits the use of recursion or any other mechanism that requires re-entrant code. For example, an interrupt routine can't call a function that may also be called by the main flow of execution. In return for this loss of flexibility, the programmer is guaranteed no run-time memory allocation issues. It might be useful if all compilers gave the programmer the option of not using the stack. By statically defining all of the space, the programmer sacrifices some flexibility and efficiency, in exchange for extra robustness.

For large systems, completely static allocation is not feasible since an enormous amount of RAM would eventually be required to satisfy every possible execution path of the program.

Some clever compilers may establish that two particular functions can't be simultaneously active and, so, allow the memory blocks associated with those two functions to overlap. This approach puts an extra restriction on the code that function pointers can't be used.

To benefit from the inherent memory safety of a completely static environment, it's important that the programmer avoid introducing dangers by trying to implement dynamic memory (such as reusing global data for different purposes) on top of the static environment.

For large systems, completely static allocation is not feasible since an enormous amount of RAM would eventually be required to satisfy every possible execution path of the program.

Stack-based management

The next step up in complexity is to add a stack. Now a block of memory is required for every call of a function, and not just a single block for each function in existence. The blocks are stored on a stack, and are usually called *stack frames.*

The stack grows and shrinks as the program executes, and for many programs, it isn't possible to predict, at compile time, what the worst case stack size will be. A multitasking system will have one stack per task (plus possibly an extra one for interrupts). Some judgement must be exercised to make sure that each stack is big enough for all of its activities. It's an awful shame to suffer from an untimely stack overflow especially if one of the other stacks has a reserve of space that it never uses. Unfortunately, most embedded systems do not support any kind of virtual memory management that would allow the tasks to draw from a common pool as the need arises.

One rule of thumb is to make each stack 50% bigger than the worst case seen during testing. In order to apply this rule, the programmer must know how big the stack, or stacks, became during testing. One simple technique is to "paint" the stack space with a simple pattern. As the stack grows and shrinks it will overwrite the area with its data. At a later time, a simple loop can run through the stack's predefined area to detect the furthest extent of the stack. Figure 2 shows an example of the life of a simple stack.

The simple pattern written to the stack should be non-zero, since it is quite common to have data on the stack which has been assigned to zero. It would be difficult to distinguish this data from unused stack space.

Many RTOSes offer a stack size tracing feature. If yours does not, or if you are not using an RTOS, it's not difficult to implement it yourself, though it is likely to be non-portable. The technique can be used during the testing phase to refine the stack sizes, and it can also be used on a production system to give early warning of a stack that exceeds a watermark that the designers did not expect to be reached. In this case, the watermark level on the stack is checked to see if the pattern has been overwritten. An expensive measurement of the exact extent of the stack is unnecessary. Checking the watermark on every write to the stack would be difficult and expensive, but it can be checked easily on a timed basis. This may not catch a stack overrun due to infinite recursion, which would overflow the stack very quickly, but it would catch a case where the stack grew a small amount bigger than the designers expected.

The previously described technique fails in one scenario. Consider a large local array which extends beyond the top of the stack. If the program does not write any data to the array, the pattern will not get overwritten. The highest legal piece of stack space will contain the pattern, and so it will look as if the stack did not overflow. Data pushed onto the stack will overwrite some other area of memory, but checking the stack will indicate no problem. If you guess that this is what has happened then the easiest way to check is to make the stack size much bigger, and check the size again. Now that the array is within the bounds of the bigger stack, the true worst case stack size will be found.

Heap-based management

Many objects, structures, or buffers require a lifetime that does not match the invocation of any one function. This is particularly true in event-driven programs, which is typical of many embedded systems. One event may cause an item to be created, and that item will remain in use until some other event leads to its demise.

In C programs heap management is carried out by the `malloc()` and `free()` functions. The `malloc()` function allows the programmer to acquire a pointer to an available block of memory of a specified size. The `free()` function allows the programmer to return a piece of memory to the heap when the application has finished with it.

While stack management is handled by your compiler, heap management requires care by the programmer. A number of particularly devious bugs can creep into your program by way of the heap.

At a certain point in the code you may be unsure if a particular block is no longer needed. If you `free()` this

RUPERT ADLEY

FIGURE 1 Memory organization with no heap and no stack

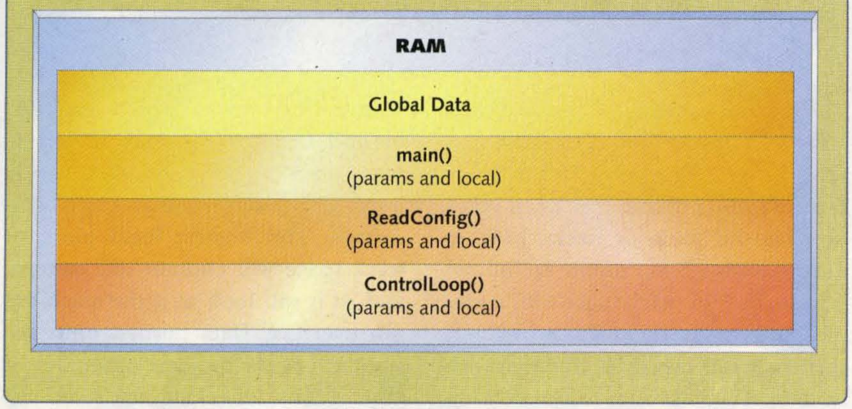
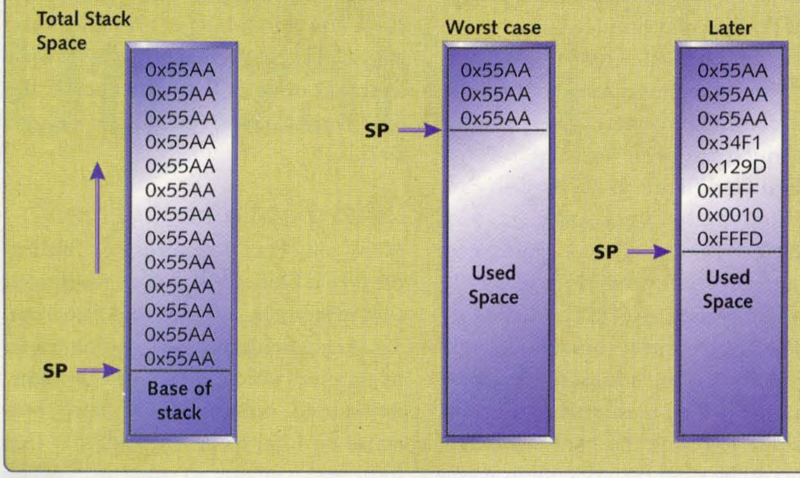


FIGURE 2 The life of a simple stack



piece of memory, but continue to access it (probably via a second pointer to the same memory), your program may function perfectly until that particular piece of memory is reallocated to another part of the program. Then two different parts of the program will proceed to write over each other's data. If you decide to not free the memory, on the grounds that it may still be in use, then you may not get another opportunity to free it (since all pointers to the block may have gone out of scope or been reassigned to point elsewhere). In this case the program logic will not be affected. But if the piece of code that leaks memory is visited on a regular basis, the leak will tend towards infinity, as the execution time of the program increases.

Ultimately, the amount of physical memory will decide how long the program can execute. On many desktop applications a small leak is acceptable, say a compiler which leaks 100 bytes for every 1,000 lines compiled. Such a program can still happily compile a 100,000-line file on a modern PC, since on exit of the program all allocated memory will be recovered. However, on many embedded systems no upper limit on the life of the program is acceptable. Any memory leak is a bug, and should be rectified by correcting the logic of the application program.

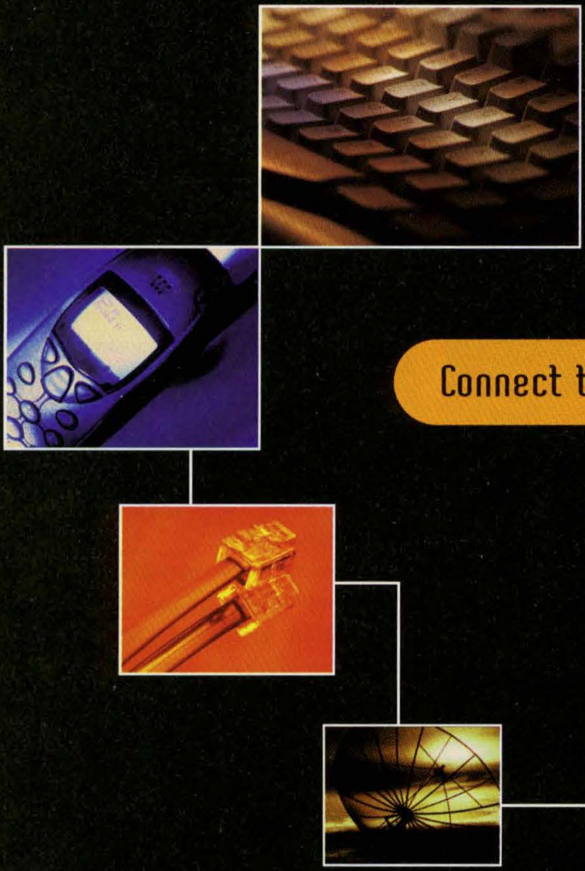
In addition to leaks there is another problem called fragmentation, which can't be corrected at the application level. This problem is inherent in most implementations of `malloc()`. It is caused by the blocks of memory available being broken down into smaller pieces as many allocations and frees are performed.

Does this mean that `malloc()` and `free()` cannot be used in embedded systems? No, but there are so many restrictions that in many cases programmers choose against it or they write their own restricted versions of `malloc()` and `free()`.

In order to better understand where the limitations lie, we will now examine how `malloc()` works. The following description is of a typical implementation, but the standard C specification does not demand that it be implemented this way.

The heap is a large block of memory that is made up of smaller blocks of memory allocated to the application and blocks that are free. Each block, allocated or freed, contains a header. Figure 3 shows a heap in its initial state and the result of a single allocation of 10 bytes. The Free List pointer always points to the first available block. When an allocation is requested, this list is iterated, searching for a block to return. Ideally, a block of exactly the right size is available. If not, some larger block is broken into two. In this way, an initial heap of one large block can become a heap containing a linked list of many small blocks that are free, interspersed with many blocks that have been allocated to the application.

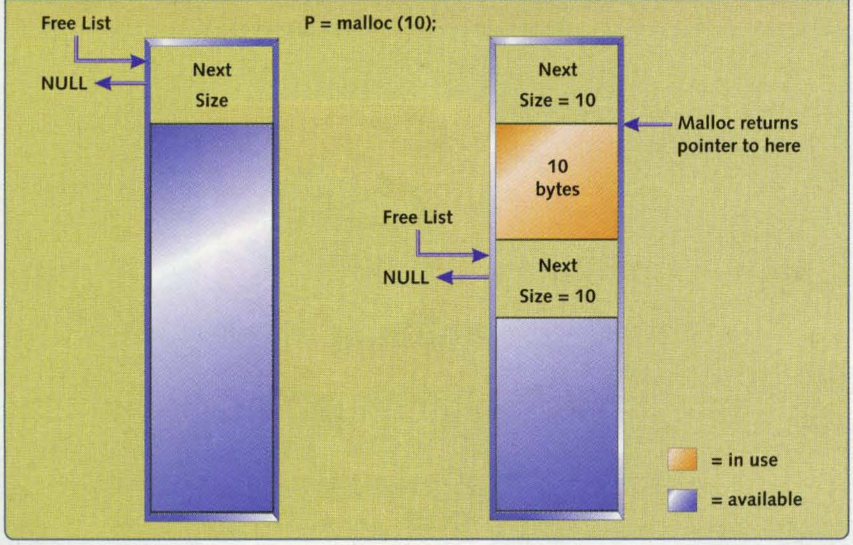
Figure 4 shows the heap after a number of allocations. On the left-hand side, the free list still only contains a single element. Next, one of the blocks is freed and the right-hand side shows a free list with a second element. The available bock is of size 15 bytes. If an allocation of 10 bytes took place, the block of 15 may be broken down into a block of 10 and a block containing the remainder. The remainder block may be so small that no request is ever made that it can satisfy. While free blocks such as this may



be merged later with adjacent free blocks, there is a danger that some will be lost forever.

The danger of fragmentation has been overestimated by academic experiments that focused on randomly sized allocations. In practice, allocations tend to come in a limited number of sizes. In a survey of a number of Unix applications, it was found that 90% of allocations were covered by six sizes, and 99.9% of allocations were covered by 141 sizes.[1] This means that the probability of finding a block that exactly matches the size of any given request is far higher than would be estimated given a random distribution of allocation sizes.

FIGURE 3 A heap in its initial state and the result of a single allocation of 10 bytes



I believe that in embedded systems the variety of sizes allocated in any one application is even smaller. File and string handling are rarer in embedded applications, and those are areas where allocation sizes tend to vary the most. Allocation of space for data structures will be more restricted, since their size does not vary at run time.

While the request pattern may reduce fragmentation, we still want our `malloc()` and `free()` code to keep it to a minimum. Fragmentation can also be reduced by using the appropriate policy when allocating and freeing blocks. Possible allocation policies include:

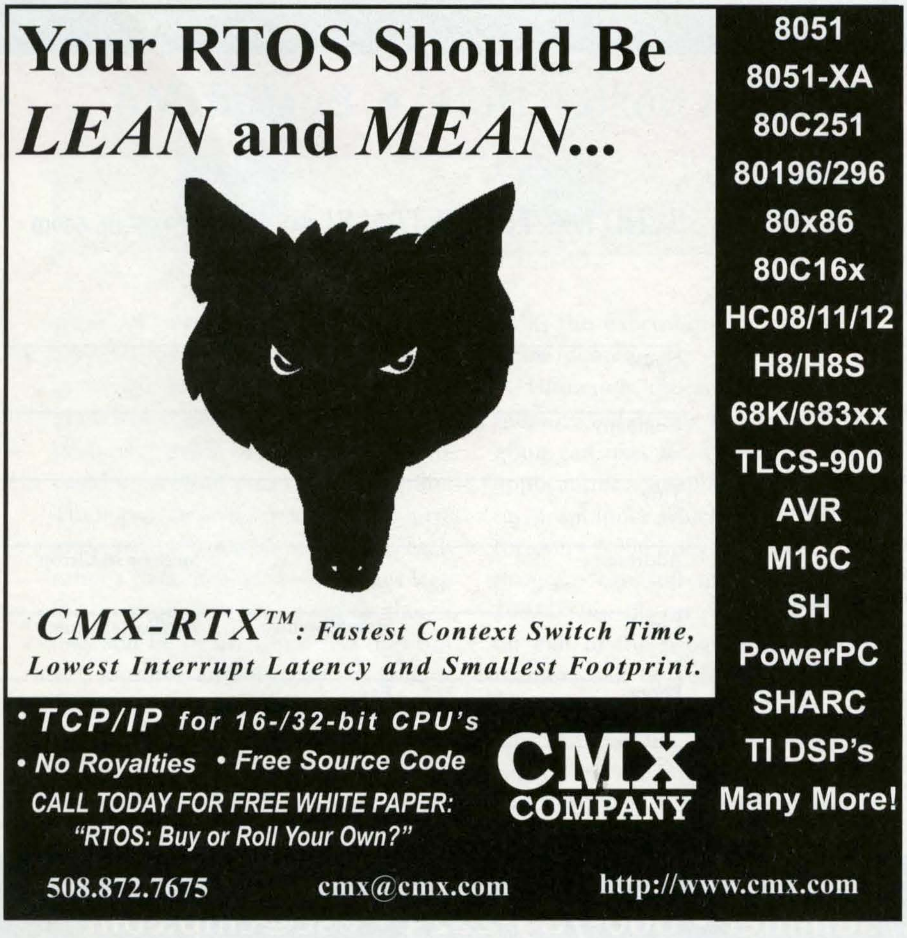
- First fit: allocate (and possibly split) the first block found that is large enough to fulfill the request
- Best fit: allocate the best fit after an exhaustive search

Possible free list management policies include:

- Sort the free list in order of address, to simplify merging of adjacent free blocks
- Maintain the list in most recently used order, to match patterns of use where similar sizes are allocated and freed in bursts

Unfortunately, the policies that lead to least fragmentation (Best Fit and address order lists) take the most time to allocate and free blocks. So the choice of algorithm is going to involve trade-offs.

Careful design of the heap mecha-



LISTING 1 Simple allocator code

```
#define SALLOC_BUFFER_SIZE 90000
static unsigned char
GS_sallocBuffer[SALLOC_BUFFER_SIZE];

static Boolean FS_enabled = TRUE;
int GS_sallocFree = 0;

void *salloc(int size)
{
    void *nextBlock;
    assert(FS_enabled);
    if(GS_sallocFree + size >
            SALLOC_BUFFER_SIZE)
        assert(FALSE);

    nextBlock =
&GS_sallocBuffer[GS_sallocFree];
    GS_sallocFree += size;
    return nextBlock;
}

void sallocDisable(void)
{
    FS_enabled = FALSE;
}
```

nism can lead to systems that suffer fragmentation losses of only 1% in Unix applications.[1] This is a small amount if it is constant, but it's difficult to establish that a program will not make a pattern of requests that increases that amount at some later point in its execution. The conclusion is that heap use does involve an element of risk, which the programmer may choose to accept in return for a more flexible, RAM-efficient system.

Static with allocation

Projects that either do not need the complexity of a full heap or can't afford the risk of fragmentation, can use a technique that allows allocation, but not freeing. This means that after a program has completed its initialization code, the main loop of the program (or the loop of each of its tasks) will not allocate any further memory. This technique can be implemented with the normal `malloc()` routine, but I've always found it useful to write a custom version. My custom version has the following advantages over using the normal `malloc()` routine:

- The overhead of the headers on each block is avoided
- The routine can be disabled once initialization is complete

This technique also has the following advantages over declaring all memory globally.

- Different start up sequences can allocate memory to different purposes, without the programmer having to explicitly consider which items can be active simultaneously
- The namespace does not get polluted as much. In many cases a pointer to an item may exist, but there is no need for a global or file static to exist for the item itself. Creating the global or file static allows access to the item from inappropriate parts of the code
- It is easy to transition to using a `free()` function later

Listing 1 contains the simple allocator code. One thing that might need to be added for a multitasking system is a locking mechanism to prevent simultaneous access from a number of tasks. This allocator does not handle memory alignment issues that you may need to address, depending on the alignment restrictions of your platform.

While this approach is memory-safe in comparison to heap usage, it consumes far more RAM than a design that also uses `free()`. However that amount of RAM can be determined by a single run of the system, and will not vary after the `sallocDisable()` function has been called at the end of the start-up sequence.

Pools

We now return to schemes that allow the application to free memory. Pools, or partitions, of fixed-size memory blocks can be used to completely eliminate the potential for fragmentation. They are a compromise between static allocation and a general purpose heap, since this heap can be tuned at design time for the size of the requests that will be made. While the standard implementations of `malloc()` and `free()` have to be general purpose, many embedded systems consist of a single program, and your heap can be tuned so that it works brilliantly for this one program even though it might fail miserably for others.

Each pool contains an array of blocks. Unused blocks can be linked together in a list. The pools themselves are declared as arrays. This mechanism avoids the overhead of a header for each block, since size information is fixed for each pool. Figure 5 shows the way in which requests are directed to the pool which is equal to the request, or the next larger block, if no exact match is available. This system must be tuned by deciding which size blocks to make available and how many blocks to provide in each pool. Defining pools at sizes which are powers of two (that is, 2, 4, 8, 16, 32, 64 ...) is a good starting point to use if size measurements have not yet been taken for your application.

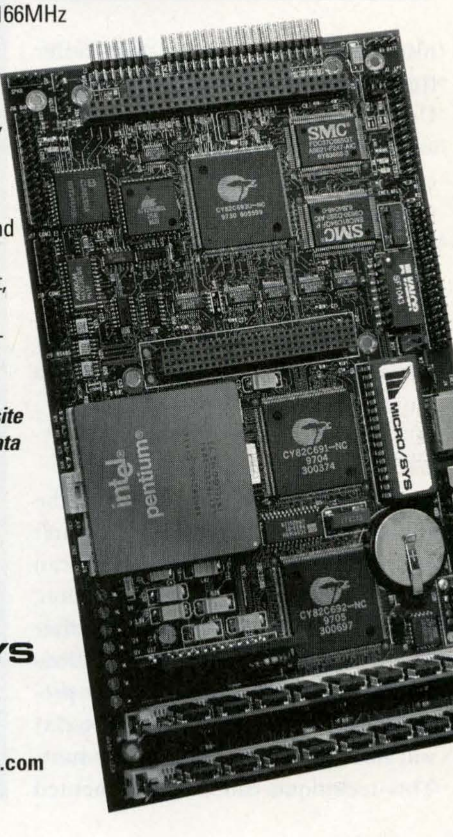
One of the major motivations for using pools to implement a heap is that a careful implementation can have fixed execution times for allocations and for freeing of blocks. More general heap implementations always involve iterating through lists which can vary in size.

By monitoring the size of each pool, and confirming that the number of blocks in use ceases to grow after extended use, the designer can be confident that leaks have been eliminated. While it is wise to size the pools larger than the worst case seen in test, designers should be aware that allowing too much "padding" leads to wasted memory.

Some implementations of `malloc()` use pools for small requests and a general purpose heap for large requests.[2]

RTOS partitions

Many RTOSes provide a memory pool mechanism, usually called partitions. Partitions are useful for implementing the pool-based heap described above. If you use a number of partitions to implement your heap, then avoid



using the RTOS calls directly. You may have calls such as those in Listing 2.

This format is typical of many RTOSes. Clearly, the onus is on the application programmer to be certain that a block is returned to the partition from which it was allocated. When implementing your pools, hide the calls to the partition code. Pass the size required to the allocation function and allow it to decide on the best partition to use. Only pass the pointer to the block to the `free()` function. It should be able to derive the partition from the address of the block. So the code snippet above will change to something much more maintainable like:

```
block1Ptr =  myAlloc(1000);
block2Ptr =  myAlloc(200);

myFree(block1Ptr);
myFree(block2Ptr);
```

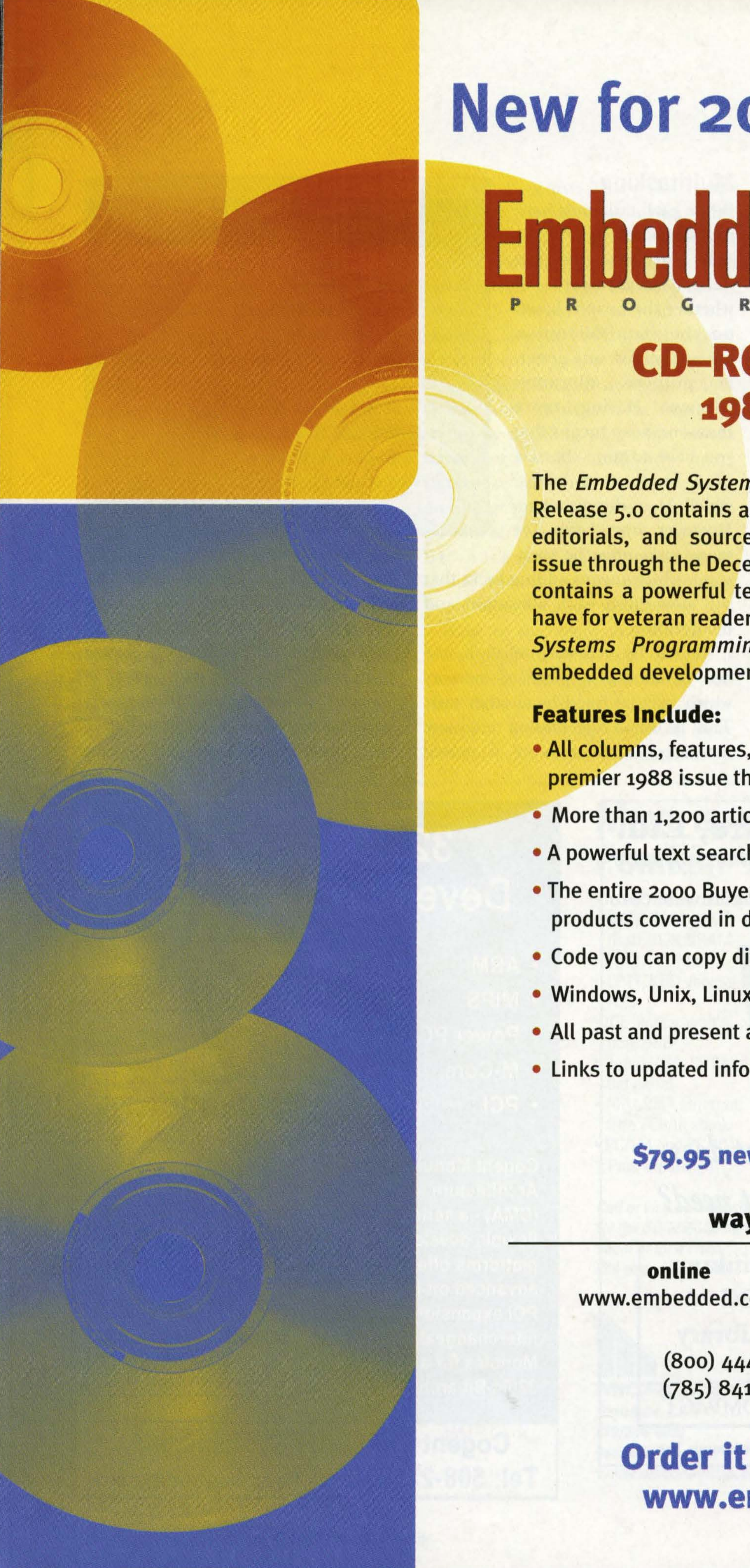
Multitasking

While each task must have its own stack, it may or may not have its own heap, regardless of whether the heap is based on the static allocation scheme, pools, or a general purpose allocation scheme. Having more than one heap means that you have to tune the size of a number of heaps, which is a disadvantage. However one heap for many tasks must be reentrant, which means adding locks that will slow down each allocation and deallocation.

It may be necessary to allow one task to allocate a piece of memory which may be freed by another task. This is useful for passing inter-task messages. When memory is passed between tasks in this way, make sure that it is always well defined who owns the memory at each point. It is obviously important that two tasks do not both believe that they own a piece of memory simultaneously. If this happened, it could lead to two calls to free the same memory block.

LISTING 2 The wrong way to allocate from partitions

```
block1Ptr =
partitionGetBlock(partitionOfBlocksSized1000);
Block2Ptr =
partitionGetBlock(partitionOfBlocksSized200);

partitionFreeBlock(block1Ptr,
partitionOfBlocksSized1000);
partitionFreeBlock(block2Ptr,
partitionOfBlocksSized200);
```



New for 2000...

EmbeddedSystems
PROGRAMMING

CD–ROM Library 1988–1999

The *Embedded Systems Programming* CD-ROM Library Release 5.0 contains all columns, features, news items, editorials, and source code from the 1988 premiere issue through the December 1999 issue. This time-saver contains a powerful text search engine and is a must-have for veteran readers and for those new to *Embedded Systems Programming*, the preeminent source of embedded development for more than 11 years.

Features Include:

- All columns, features, and source code from the premier 1988 issue through the December 1999 issue
- More than 1,200 articles—all columns and features
- A powerful text search engine
- The entire 2000 Buyer's Guide—more than 1,200 products covered in detail
- Code you can copy directly into your designs
- Windows, Unix, Linux and Mac compatibility
- All past and present author biographies
- Links to updated information on www.embedded.com

$79.95 new $29.95 upgrade

ways to order:

online
www.embedded.com

e-mail
orders@mfi.com

phone
(800) 444-4881 U.S./Canada
(785) 841-1631 other countries

Order it Online today!
www.embedded.com

FIGURE 4 The heap after a number of allocations

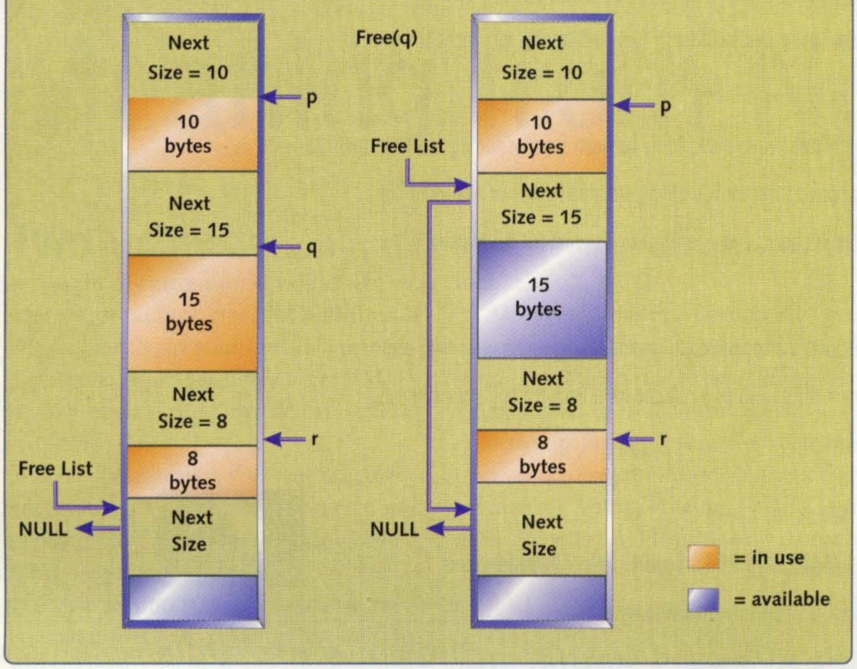
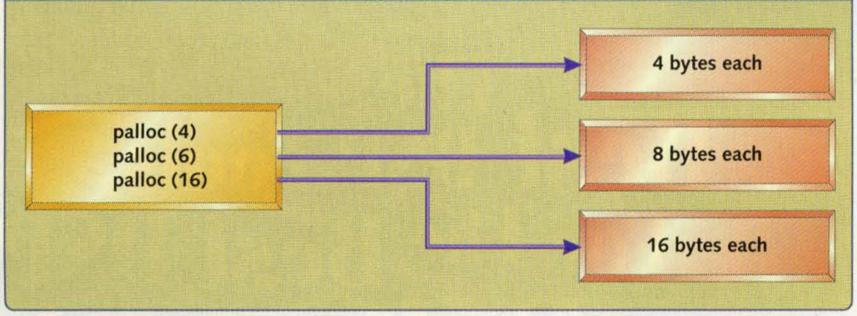


FIGURE 5 Using the appropriate pool



Libraries

Libraries, whether written in-house or purchased from a third party, can cause many difficulties in memory management. The author of the library does not have full knowledge of how the library is going to be used. A library may allow the application code to create an object, or allocate memory in some way, but the library may not be able to free that item because the library does not know when the application has finished with it.

Consider a library that concatenates two strings and returns the result in a newly allocated block. The library can't tidy up the string later because the library doesn't know when the application has finished with it. One possibility is that the library has a `destroyString()` routine that the application calls when it has finished with the item. This has the disadvantage that the onus is on the application to remember to call this function. Another approach is that the library always uses a static space so that the string returned is valid until the next time that function is called, at which time that space will be overwritten with the next result. This latter idea is not suitable for reentrant code, which is so essential to many embedded systems.

Many libraries, especially object-oriented libraries, will allocate storage at some time that the application will have to free. In such cases the rules must be very explicit and clear, and the author of the library must ensure that these rules are communicated to the application writer.

Some libraries will allow the application to specify which `malloc()` and `free()` functions should be used for its heap management. This allows the application to manage its own memory separately from the libraries. By using debug versions of `malloc()` and `free()`, the designer can distinguish between a leak in the application and one contained within the library.

Final decisions

This overview of memory management should have given the reader some ideas about what approach is right for their project. The final design decisions will be based on a combination of how much RAM is available, the complexity of the application, and whether third-party software is involved. **esp**

Niall Murphy has been writing software for user interfaces and medical systems for ten years. He is the author of Front Panel: Designing Software for Embedded User Interfaces. *Murphy's training and consulting business is based in Galway, Ireland. He welcomes feedback and can be reached at nmurphy@panelsoft.com.*

References

1. Wilson, Paul and Mark Johnstone. "The Memory Fragmentation Problem: Solved?" International Symposium on Memory Management, Vancouver, Canada, October 1998. Available at *www.cs.utexas.edu/users/oops/papers.html*
2. Lethaby, Nick and Ken Black. "Memory Management Strategies for C++," *Embedded Systems Programming*, July 1993, p. 28.

The
Best DSP Tools
For The Best DSPs
Delivered.

Don Morgan

Meaning from Noise

Two messages from readers recently caught my attention. I would like to present them both here. In addition, I would like to say that I really appreciate the feedback; it helps a great deal in shaping the direction these columns take.

The first letter was an acknowledgement for the code in some of my columns and a request for more of the same—especially in the form of pseudo-code so that it might be more easily transported to micros as well as other DSPs.

I do like to present working code when I can, on as many platforms as possible. When I do, I will also include pseudo-code. I realize the assembly language code is extremely processor-specific, but I like to present working code so that the reader can view the reality of the coding and test it. It's always nice to have something to compare your code to. Real code also reveals some difficulties that may not be evident in pseudocode. For instance, it is a simple matter to say that cotangent = x/y, but coding that in fixed point presents some problems that depend upon word size and normalization that are not immediately apparent in a high-level abstraction like pseudocode.

In a related vein, I realize that the math associated with some functions can sometimes seem overwhelming. It might seem that simply having the code is enough, but the truth is that understanding how something works simplifies debugging and integrating code as well as developing it. This understanding also allows the engineer to create his own special variations much more easily. It becomes a tool.

Another reader contacted me about my column in the February issue ("A Paean to Noise," February 2000, p. 95). In that column, I had created the impression that the distribution of a random noise sequence is related to its spectrum. This is not true. To prove it, I'll show how to compute the spectrum of a random sequence in this issue. This will give me an opportunity to explore some fundamentals of statistical processing and describe how some of its tools relate to digital signal processing.

The author brings in statistical tools such as correlation and show how they relate to digital signal processing.

Here are two questions: we have said that white noise does not correlate, but what does that mean? If it doesn't correlate, what can we know about this stuff?

How can something random be stationary?

Before we begin, I would like to draw a distinction between two sorts of random processes. To understand this, it helps to think of a random process as an infinite collection of sample functions all occurring simultaneously. A process is stationary if its sample averages are independent of absolute time. We have been and will be talking about stationary processes in this column. How do you know if something is stationary? If a process has a beginning or ending, it probably isn't stationary. It may, however, be considered stationary (at least over most of its lifetime) if it lasts for a long time compared to the period of its lowest frequency spectral components.

A random process that is independent of actual time is much easier to work with and analyze, though it can be argued that there are really no *true* stationary processes, only those that are stationary for all practical matters. A random number generator that generates a very short sequence can hardly be described as stationary, but one fulfilling the considerations in the previous paragraph will suit.

Zero mean

White noise is a *zero mean* random signal—its average, or mean value, is zero. Using continuous mathematics, we express this with an integral:

$$(E[x])T = \int_0^T x(t)dt \qquad (1)$$

Equation 1 is just a way of saying that we can find the area under a curve we call $x(t)$ during the interval 0 to T by taking the average of the sum

If you do not have a closed form for predicting the behavior of a function—as you do with a sinusoide, for instance—you can always determine the mean and use that as a guide to future values.

of the individual pieces we call $x(t)$. The solution of this integral is a limit approached by the sum of the averages (or average of the sum). In this formula, we also introduce the symbol $E[x]$, which means "the statistical expectation of," or the *mean.*

The concept of the mean is applied ubiquitously; it describes a characteristic of a signal or sequence. If you do not have a closed form for predicting the behavior of a function—as you do with a sinusoide, for instance—you can always determine the mean and use that as a guide to future values. It helps us here to bring meaning to signals that are not deterministic, such as a random series.

But if the mean is all you have, you will quickly find that it does not provide enough information for most applications. Many and widely varying numbers can produce the same average, including numbers that vary widely in value from one another, as well as sequences comprising very small excursions interspersed with very wide ones.

Mathematicians are frequently concerned with how closely an approximation follows an actual or known signal. Many series, such as the Taylor series, will cling rather closely to a function, only to diverge widely at some point or points. Therefore, to really characterize an approximation, you would not only want the mean of the function but you would also like to know just how far the sequence diverges when it does. If you have a zero mean signal, its expected value is going to be zero; its instantaneous value might be anything.

This brings us to an extremely important value known as the standard deviation. In determining how closely an approximation of any sort might be to an actual signal, we can use the standard deviation as a figure of merit. The standard deviation, or variance, σ^2, is given by:

$$\sigma^2 = E\left[\left(x - E[x]\right)^2\right]$$

This equation reads as "the mean of the square of the deviation of the value and its expected value"—the average of how far the value strays from the ideal. Using this we can determine how closely a signal or sequence tracks an ideal.

Let's look at this idea another way.

Correlation

There is another wonderful, but perhaps not immediately obvious, aspect of stationary random sequences such as white noise. If we multiply two such data sequences by one another (sample by sample), summing the products after each multiply, our final sum will be zero or very close, depending on the length of the sequence.

To illustrate how this works, let's take first a sine wave at an arbitrary frequency, say 10Hz. We then sample it at 1kHz, producing 100 samples of each complete sinusoid. This means that we will have 50 samples of the positive half and 50 samples of the negative half. This sequence will start at zero and continue until $N = 999$. Let us note this:

$$x(n),\ n = 0,1,2\ldots N$$

Now, we take that same sequence (same 1,000 samples, 100 samples per sinusoid), and we offset it from the origin ($n = 0$) slightly, say five samples and call this k, so that $k = 5$.

If we multiply these two sequences, the original starting at zero and the new one starting at $k = 5$, by one another, sample by sample, summing these products as we go, then dividing by the length of the sequence, we will find that we get a positive result—actually, a large positive result. This is because, if you follow this in your imagination, you will be multiplying positive halves of the sine waves by that same positive half offset by only five samples for a positive value and you will be multiplying negative halves by negative halves (with the same offset) for a positive value. The only numbers that subtract occur when the offset is a positive half while the original sequence is in the negative and the reverse.

Of course, the result of a correlation is not always positive. If the offset in our example was $k = 50$ instead of 5, we would be multiplying the samples in the positive half of each sine wave by the samples in the negative half then summing, resulting in a large negative sum. The larger the result, in a normalized case, the more closely it approaches one (1 or -1), the more the sample streams correlate.

Since I already presented the integral form in last month's column, allow me to introduce the statistical (discrete) representation:

$$R_{xx}(k) = E\left[x(n+k)x(n)\right] \qquad (2)$$

This is the *autocorrelation function.* Note that *xx* denotes autocorrelation; there is another function called a *cross correlation* taken between two different sequences that will have *xy* next to the operator. You will also note that the result of the function is dependent upon the offset (k).

Taking this same principle and applying to a data stream whose constituent samples are random will result in a comparatively small non-zero sum, as you probably already see. In that case, you will see the bias if any.

This function finds untold uses in all forms of digital signal processing. In video, for instance, say you needed to find a feature or produce a map of a video field. If you took a correlation line by line, both vertically and horizontally, you could find lines and fea-

CMP MEDIA PRODUCT CATALOGS

Complete Embedded Product Information

Available Now!

ARM Development Guide
AMD Fusion E86 Catalog
CAN Solutions Directory
DSP Source Guide
Embedded Internet Source Guide
8051 Product Directory
Flash Memory and Components Source Guide
Hitachi Development Tools Catalog
IP Catalog for System-on-a-Chip Design
MIPS RISC Resource Catalog
Philips 80C51 + XA Development Tools
PowerPC Resource Guide
Siemens Microcontroller and DSP Development Tools Guide
USB, PCI and CompactPCI Connectivity Solutions and PC/104—PC/104-*Plus* Catalog
VantageISI Partners Catalog
X86 Catalog: including 586 & 686 Derivatives
Windows® CE Product Catalog
Windows® Embedded Catalog

ARM DEVELOPMENT GUIDE 1999-2000
1999 CAN SOLUTIONS
1999 DSP SOURCE GUIDE
Embedded Internet Source Guide 1999
8051 PRODUCT DIRECTORY 1999 EDITION
1999 FLASH MEMORY AND COMPONENTS SOURCE GUIDE
80C51+ & XA DEVELOPMENT TOOLS CATALOG
PHILIPS
1999 IP Catalog for System-on-a-Chip Design
MIPS RISC RESOURCE CATALOG
HITACHI Microcontroller Development Tools Catalog
Integrated Systems, Inc. Vantage ISI Partners Catalog
PowerPC RESOURCE GUIDE
USB, PCI and Compact PCI Connectivity Solutions Catalog PC/104 and PC/104-Plus
WindowsNT Embedded Product Catalog
WINDOWS CE PRODUCT CATALOG 1999
X86 CATALOG

3 easy ways to get your FREE product catalogs:

1. pick them up the Embedded Systems Conferences
2. order online through www.embedded.com
3. call 1-800-500-6815 in the U.S., 1-785-841-1631 outside the U.S.

Check out all the product catalogs at www.embedded.com

Additional charge applied to multiple orders

CMP
OEM Group

There isn't much magic to writing code for a correlation. In fact, it involves exactly the same arithmetic as a FIR filter except the time index is reversed.

tures in the histograms that result. In audio, this function allows us to find the power in a signal and eliminate noise, that is, those signals that do not correlate.

Power spectrum

Now, if you are to take the DTFT (Discrete Time Fourier Transform) of the autocorrelation function, you will find yourself with a spectrum representing the frequency content of the correlation. This is called the *power spectrum.* The power spectrum is given by:

$$S_{xx}(\omega) = \sum_{k=-\infty}^{\infty} R_{xx}(k)e^{-j\omega k}$$

This formula is a simple statement of the DTFT taken of $R_{xx}(k)$. We can use this technique to determine how the signal's power is distributed in the spectrum.

Now, if $k = 0$, we are taking the DTFT of the sum of the squares of the input sample sequence. The result is the average power or (again) the *variance* of the sequence, which is probably clear from Equation 2. If we set $k = 0$, we get:

$$\begin{aligned} R_{xx}(k) &= E\left[x(n)x(n)\right] \\ &= E\left[x(n)^2\right] = \sigma^2 \end{aligned}$$

To repeat the definition given previously, the variance is the "the mean of the square of the deviation of the value and its expected value." In this case, we have a zero mean signal, thus the variance is given as shown.

Since, for white noise, the power per frequency is equal to only the variance, we can say that its spectrum is flat. And we can say that white noise had equal energy per frequency.

Correlation algorithm

There isn't much magic to writing code for a correlation. In fact, it involves exactly the same arithmetic as a FIR filter except the time index is reversed. A versatile routine would make it possible to do both autocorrelations and cross correlations with the same code.

To do this, we need two strings (x,y), their dimension (N), a maximum value for k(M), and a result array (R) to hold the M dimensions of k.

1. Construct a `for` loop that will run from $k = 0$ to M
2. Nest another `for` loop that will calculate the value of R for each k as $R[k] \mathrel{+}= x[n+k]*y[n]/N$

The correlation and power spectrum can be useful tools that find use not only in statistical processing but also in all areas of signal processing. And they are based on such simple concepts.

Next month, we will look at how to use noise to make music. **esp**

Don Morgan is senior engineer at Ultra Stereo Labs and a consultant with 25 years experience in signal processing, embedded systems, hardware, and software. Morgan recently completed a book about numerical methods, featuring multi-rate signal processing and wavelets, called Numerical Methods for DSP Systems in C. *He is also the author of* Practical DSP Modeling, Techniques, and Programming in C, *published by John Wiley & Sons, and* Numerical Methods for Embedded Systems *from M&T.*

Free Pass to the Product Exhibits

a$35 value!

DSP World Spring Design Conference 2000

New Feature Added!

Free Introductory DSP Training

Attendees will have the opportunity to attend one of two introductory training sessions offered by Chris Bore, an internationally recognized expert in DSP training. The classes are available either on Tuesday, April 11, from 10:00am – 12:00pm or on Wednesday, April 12, from 9:00am – 11:00am.

Exhibit Hours

Tuesday, April 11
12:00pm – 7:30pm

Wednesday, April 12
11:00am – 5:00pm

DSP WORLD SPRING DESIGN CONFERENCE

DSP World Spring
Design Conference 2000

April 10-12, 2000
McEnery Convention Center
San Jose, CA, USA

www.dspworld.com

ESP4

Bring this ad for FREE admission to:

- Two-day admission to the DSP World product exhibits, April 11-12, 2000
- Keynote address, *Fast Forward: How Digital Signal Processing is Driving the Broadband Communications Era* Presented by John Scarisbrick, Texas Instruments, Inc. 5:15pm - 6:00pm, April 11
- Opening Night Welcome Reception on the Show Floor 6:00pm - 7:30pm, April 11

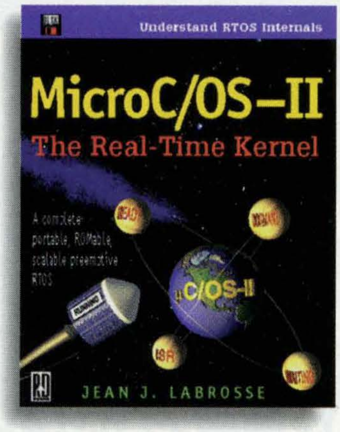
...and many other exciting plenary events. See www.dspworld.com for details!

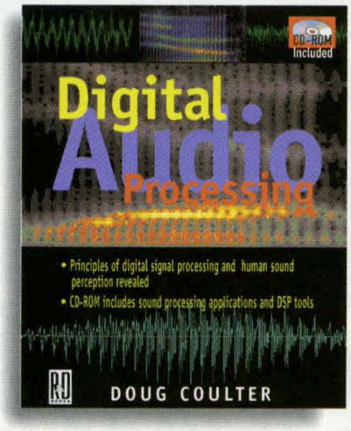
CMP EETIMES

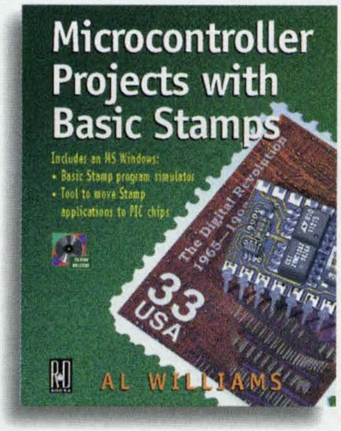
CMP Communication Systems design

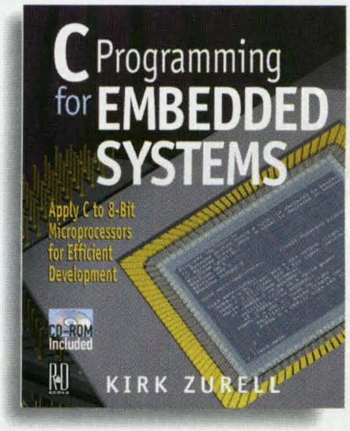
CMP EmbeddedSystems PROGRAMMING

CMP


Understand RTOS Internals
MicroC/OS–II
The Real-Time Kernel
A complete portable, ROMable, scalable preemptive RTOS
µC/OS-II
JEAN J. LABROSSE


CD-ROM Included
Digital Audio Processing
• Principles of digital signal processing and human sound perception revealed
• CD-ROM includes sound processing applications and DSP tools
DOUG COULTER


Microcontroller Projects with Basic Stamps
Includes an MS Windows:
• Basic Stamp program simulator
• Tool to move Stamp applications to PIC chips
33 USA
AL WILLIAMS


C Programming for EMBEDDED SYSTEMS
Apply C to 8-Bit Microprocessors for Efficient Development
CD-ROM Included
KIRK ZURELL

Tools for Embedded Developers

Software

RTOS simulation environment

ISIM is an advanced RTOS simulation environment for the INTEGRITY real-time operating system. Initially available for PowerPC processors, ISIM enables designers to develop and test INTEGRITY-based embedded applications on their PCs or workstations without requiring access to actual target hardware. It's a PowerPC instruction set simulator that provides support for PowerPC systems running INTEGRITY, including virtual memory facilities, exception handling, and debugging across a LAN. ISIM runs on Windows 95/98/NT and Solaris 2.x host platforms. ISIM is integrated with the MULTI debugger and supports both freeze-mode and run-mode debugging. ISIM is purported to simulate INTEGRITY-based PowerPC applications on a Windows or Unix-based host system at a speed of nearly 2 MIPS on a 550MHz Pentium III. ISIM is available now for use with INTEGRITY for $2,500 per user. The INTEGRITY RTOS and libraries are bundled with the MULTI IDE, a C/C++/EC++ compiler, a board support package, and a royalty-free run-time target project license. The complete package costs $7,900 per user on a PC and $10,900 per user on a Unix system.

Green Hills Software
Santa Barbara, CA
(805) 965-6044
www.ghs.com

Design tool suite

COTSware is a software designer suite that consists of the latest version of SwiftNet networking and host-to-target communications software, code generation tools, debuggers, real-time operating systems, utilities, and DSP libraries. The COTSware tool suite is available in several packages tailored for DSPs, data acquisition, and software radio applications. ReadyFlow, a new software product that's part of the COTSware suite, consists of a library of board-specific parameter definitions, header files, and C-callable functions for each product. SwiftNet v. 3.5 supplies a unified GUI control panel. This version of SwiftNet also includes SwiftNet Auto-Start, DSP-to-host control windows, and an improvement of J-TAG performance. COTSware DSP packages for 'C6x, 'C4x, and SHARC are available now starting at less than $4,000.

Pentek
Upper Saddle River, NJ
(201) 818-5900
www.pentek.com

Co-simulation environment for fixed-point DSPs

The Motorola DSP Developer's Kit simulates and test algorithms for Motorola's 56300 and 56600 families of fixed-point DSPs. It integrates the DigitalDNA development environment with Simulink and MATLAB DSP design tools. Normally, developers have to manually recode algorithms in DSP assembly language, simulating and debugging the algorithms using a separate set of tools. The DSP Developer's Kit provides co-simulation between software tools and the Suite56 simulators, integrated debuggin and analysis, and a foundation library of DSP algorithms packaged as MATLAB functions and Simulink blocks. The kit is available now on PCs running NT, starting at $995 for an individual license.

Motorola
Austin, TX
(512) 895-3131
sps.motorola.com

The MathWorks
Natick, MA
(508) 647-7000
www.mathworks.com

Software development kit for DSP family

SDK v. 2.0 for the ManArray DSP product family provides the tools for developing high-performance, scaleable, and reusable DSP cores for high-volume system-on-a-chip applications. It allows customers to convert C, C++, and MATLAB files into ManArray assembly code for DSP applications. SDK v. 2.0 offers an integrated development environment, including a system simulator; instruction set simulator; compiler for MATLAB and vector library; GNU-C compiler, assembler, linker, and loader; VLIW packer; register allocator; DSP libraries; and ManArray technical information and coding examples. SDK v. 2.0 is available now. Pricing depends on the options selected.

BOPS
Palo Alto, CA
(650) 330-8410
www.bops.com

Hardware

Pattern generator module

The TLA7PG2 is a pattern generator module for the TLA 700 Series logic analyzer. The TLA7PG2 is a multichannel, programmable pattern generator module with sequential control that plugs into the TLA 700. It generates digital stimuli to simulate infrequently encountered test conditions in hardware design and software program testing. It stimulates the prototype with data from the simulator for extended analysis. The TLA7PG2 features up to 64 channels per module and supports a 268MHz clock rate for data output. The product is be available now, starting at $12,995.

Tektronix
Portland, OR
(800) 425-2200
www.tektronix.com

The PIC16F87X flash family from Microchip.

Chips

PIC flash microcontrollers

The newest members of the PIC16F87X family of flash devices are the 28-pin PIC16F870 and 40-/44-pin PCI16F871 flash microcontrollers. With 2K x 14 bits of flash memory and 64 bytes of EEPROM data memory, the PIC16F870 and PIC 16F871 8-bit MCUs feature brownout detection, up to 5 MIPS performance at 20MHz, USART communications capability for peripheral expansion, two 8-bit timers, one 16-bit timer, and an operating voltage of 2.0V to 5.5V. The PIC16F870 has a five-channel 10-bit A/D converter. The PIC16F871 features an eight-channel 10-bit A/D converter and a parallel slave port. Available in SDIP, SOIC, and SSOP packages, the PIC16F870 (28-pin SDIP industrial temperature version) is $3.34 each in 1,000-unit quantities and the PIC16F871 (40-pin SDIP industrial temperature version) is $3.67 each. Volume production is scheduled to start in the summer.

Microchip Technology
Chandler, AZ
(480) 786-7200
www.microchip.com

32-bit DSP core

The TMS320C28x DSP is the next-generation core within the TMS320C2000 DSP platform. Based on a 32-bit architecture with controller instructions, 400 MIPS processing capability, and 32- and 64-bit math operations, the new C28x DSP core is claimed to provide 20 times the processing performance of existing solutions. The C28x DSP core is compatible with both the C24x motor control DSPs and the C27x mass storage DSPs. The C28x instruction set includes on-cycle read-modify-write operations, plus special I/O and branching operations. Extended addressing supports up to 8MB of program memory and up to 8GB of data. The TMS320C28x DSP products based on the C28x DSP core are scheduled to be implemented using 0.13-micron L-effective CMOS process technology. The first silicon samples are planned for the fourth quarter.

Texas Instruments
Dallas, TX
(972) 995-6611
www.ti.com

OEM

Board for ISP and thin server markets

The SPARCengine Ultra AXe board targets the ISP and thin server markets. It allows OEMs for service providers to create an integrated platform running on the 64-bit UltraSPARC processor. With a riser card, it fits into a 1µ form factor for thin servers. The SPARCengine Ultra AXe-300 board is now available for less than $1,000.

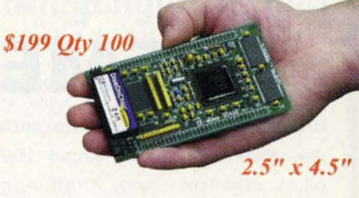
Sun Microsystems
Palo Alto, CA
(650) 960-1300
www.sun.com

Pentium III SBC

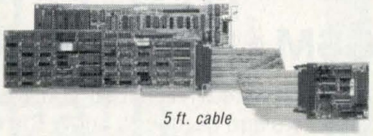
The VMIVME-7697 is a single-board computer that incorporates the Pentium III processor with MMX to the VMEbus platform, offering processor speeds up to 800MHz. It has real-time OS features such as 96MB flash, battery-backed SRAM, three 32-bit timers, and a software-programmable watchdog timer. For Windows NT and RTX applications, the VMIVME-7697 provides an S3 Trio3D AGP graphics adapter with 4MB of SGRAM. A fast/wide SCSI-2 and 10/100 Mbit Ethernet channel are also provided. The VMIVME-7697 is available now starting at $5,116.

VMIC
Huntsville, AL
(256) 880-0444
www.vmic.com

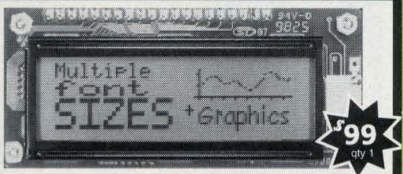
FREE
2 WEEK
TRIAL

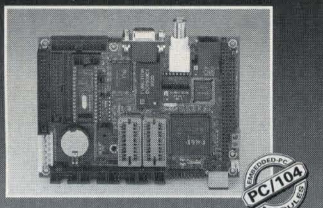


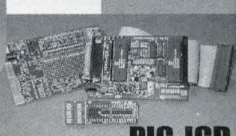
Microsoft Windows CE



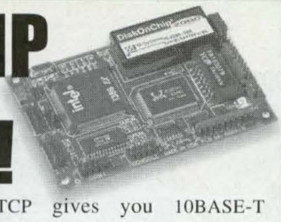
EMP-30
EMP-20
EMP-10


Multiple font SIZES +Graphics
$99
qty 1


EMBEDDED-PC
PC/104
MODULES



JUMP on the NET!



The µFlashTCP gives you 10BASE-T Ethernet connectivity, a full-function TCP/IP stack and 2 serial ports in a package 30% smaller than PC/104 solutions.

Field-proven TCP/IP stack, DOS and PC-compatible BIOS make development quick and easy.

Prices start at $169 qty 100.
Development kits are available.

Call 530-297-6073 Fax 530-297-6074
Check our web site for the latest updates
www.jkmicro.com/uflash

JK *microsystems*

From the Author of WATTCP

eRTOS

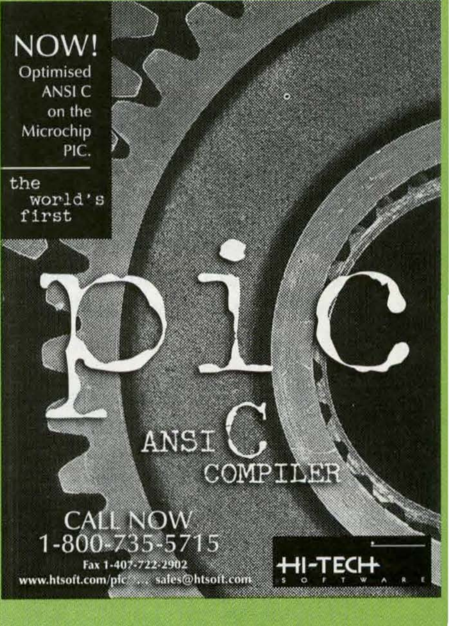
DOS Realtime Kernel with TCP/IP Support

- *Preemptive & Cooperative threads*
- *Critical Section Protection*
- *Interthread Messaging*
- *Complete Re-entrant TCP/IP*
- *Web, CGI, FTP, Email, Telnet*
- *Web Graphics*
- *Interrupt-driven Serial Support*

www.**ertos**.com

Call 530-297-6073 Fax 530-297-6074

JK *microsystems*



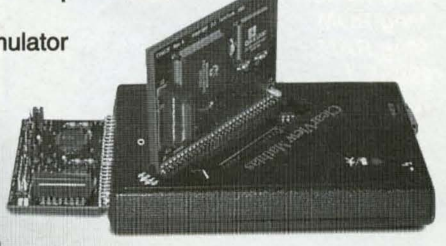
TechTools™ Embedded Systems Development Tools

Complete PICmicro® Development System

Get the TOTAL Package for only

$699

- PICmicro® In-Circuit Emulator
- PICmicro® Assembler
- PICmicro® Debugger
- Windows® IDE



ClearView™ Mathias is a full-featured In-Circuit Emulator with a highly productive Development and Debugging environment for the PICmicro.

Fully emulates the selected PICmicro, including program memory, register memory, EEPROM, I/O activity, SLEEP mode and all peripherals.

Intuitive, Easy to Learn full-featured environment with integrated ClearView Debugger.

TDE provides integrated source-level debugging for ALL popular PIC Compilers and Assemblers.

See why Engineers choose TechTools
Visit our website and request your FREE CDROM!

www.tech-tools.com

CONTACTS: (972) 272-9392 Fax: (972) 494-5814 Email: sales@tech-tools.com

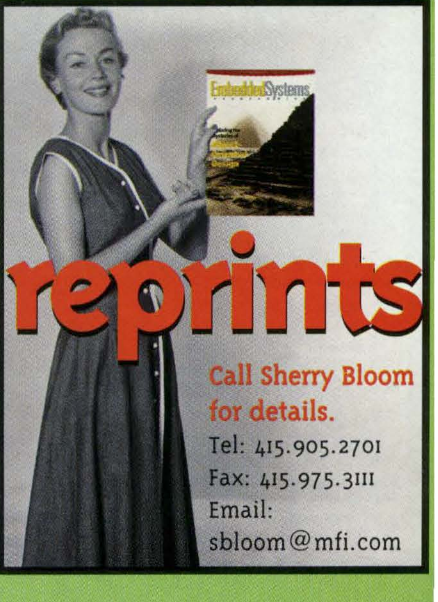
Copyright © 1999, TechTools, P.O. Box 462101, Garland, Texas 75046-2101 • ClearView, CVASM16, PICwriter, PICstation, the "Wizard" symbol and TechTools are trademarks of TechTools, P.O. Box 462101 Garland, Texas 75046. • PICmicro is a registered trademark of Microchip Technology, Inc. • All other trademarks are trademarks or registered trademarks of their respective company.

PromICE with Trace

The Leader in Memory Emulation

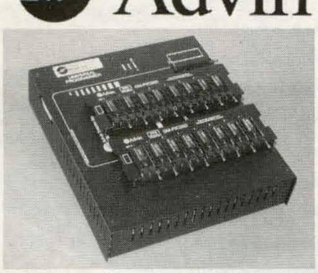
- Trace to pinpoint startup problems and isolate real-time bugs.
- Code Coverage to verify execution and speed up QA.
- Ultra-fast downloads via Ethernet, parallel and serial ports for Unix, Windows 95/NT and DOS.

Grammar Engine Inc.
Call Toll Free:
1-800-776-6423
www.gei.com

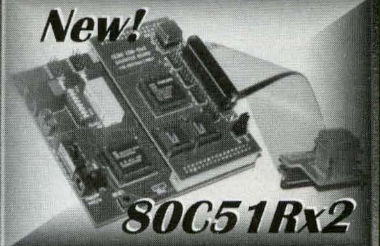


Call Sherry Bloom for details.
Tel: 415.905.2701
Fax: 415.975.3111
Email:
sbloom@mfi.com

CONITEC DATENSYSTEME
GALEP-III
Multi-Programmer
CHARGE
RDY/BSY



SB-56K


New!
80C51Rx2

ADVERTISER INDEX

Advertiser	URL	Page
Accelerated Technology	www.atinucleus.com	25
Advanced Transdata Corp.	www.adv-transdata.com	133
Advantech Technologies Inc.	www.advantec.com/epc	124
Advin Systems Inc.	www.advin.com	135
Agilent Technologies	www.agilent.com	33
Agilent Technologies	www.agilent.com	43
Allant Software Corp.	www.allant.com	101
American Arium	www.arium.com	C3
Analog Devices	www.analog.com	21
Analog Devices	www.analog.com	45
Archimedes Software	www.archimedessoftware.com	116
ARTiSAN Software Tools	www.artisansw.com	99
Avocet Systems Inc.	www.avocetsystems.com	C4
Avocet Systems Inc.	www.hmi.com	47
Bittware Inc.	www.bittware.com	104
Blackhawk	www.blackhawk-dsp.com	107
Blunk Microsystems	www.blunkmicro.com	135
CAD-UL	www.cadul.com	74
CAN-in Automation GmbH	headquarters@can-cia.de	128
Ceibo	www.ceibo.com	135
CMX Systems Inc.	www.cmx.com	114
Cogent Computer Systems Inc.	www.cogcomp.com	117
CONITEC Datasystems Inc.	www.conitec.com	135
COSMIC Software	www.cosmic-software.com	24
Dinkumware, Ltd.	www.dinkumware.com	117
Domain Technologies	www.domaintec.com	135
EBS	www.ebsnetinc.com	104
EBSnet Inc.	www.ebsnetinc.com	81
Electronic Data Systems	www.eds.com/careers	130
Electronic Engineering Tools	www.eetools.com	133
EMAC Inc.	www.emacinc.com	87
Embedded Power Corp.	www.embeddedpower.com	29
Embedded Power Corp.	www.embeddedpower.com	93
Embedded Systems Academy	www.esacademy.com	115
Embeddedtechnology.com	www.embeddedtechnology.com	109
emWARE	www.emware.com	63
Enea OSE Systems	www.enea.com	54
esmertec, inc.	www.jbed.com	65
EST Corp.	www.estc.com	1
Express Logic	www.expresslogic.com	51
Express Logic	www.ghs.com/armsolutions	88
General Software	www.gensw.com	40
Grammar Engine Inc.	www.gei.com	134
Green Hills Software Inc.	www.ghs.com/armsolutions	88
Green Hills Software Inc.	www.ghs.com	6
HI-TECH Software	www.htsoft.com	39
HI-TECH Software	www.htsoft.com/pic	134
Hitex Development Tools	www.hitex.com	135
Hitex Development Tools	www.hitex.com	35
Hiware	www.hiware.com	105
IAR Systems	www.iar.com	95
Infineon Technologies	www.infineon.com	10
Infineon Technologies	www.infineon.com	48
Insignia Solutions	www.insignia.com	67
InterNiche Technologies Inc.	www.iniche.com	69
Introl Corp.	www.introl.com	135
iSYSTEM USA	www.isystem.com	132
JK Microsystems	www.jkmicro.com/uflash	134
JK Microsystems	www.jkmicro.com	134
KADAK Products Ltd.	www.kadak.com	61
Keil Software Inc.	www.keil.com	20
Lynx Real-Time Systems	www.lynx.com	59
MetaLink Corp.	www.metaice.com	133
MetaWare, Inc.	www.metaware.com	103
Micro Digital Inc.	www.smxinfo.com	46
Micro Digital Inc.	www.smxinfo.com/ppces.htm	133
Micro/sys Inc.	www.embeddedsys.com	116
Microsoft Inc.	www.microsoft.com/embedded/	57
Microtek International Inc.	www.microtekintl.com	4
Motorola	www.digitaldna.motorola.com	8,9
National Engineering Search	www.nesnet.com	130
National Semiconductor	www.national.com	41
Needham's Electronics	www.needhams.com	132
Nohau Corp.	www.nohau.com	C2
Nohau Corp.	www.nohau.com	132
Noral Micrologics	www.noral.com	135
Nucleus Elecronics Corp.	www.nucleus1.com	133
Pacific Softworks	www.pacificsw.com	70
Paradigm Systems	www.devtools.com	83
ParaSoft Corp.	www.parasoft.com	84
Phar Lap Software Inc.	www.pharlap.com	44
Premia	www.premia.com	90
QNX	www.qnx.com	53
Rabbit Semiconductor	www.rabbitsemiconductor.com	77
RadiSys	www.radisys.com/SS7	19
RLC Enterprises	www.rlc.com	132
Scientific Microsystems Inc.	www.scimisys.com	132
Scientific Placement Inc.	www.scientific.com	130
Scott Edwards Electronics Inc.	www.seetron.com	133
Siemens AG HL	www.job4u-siemens.de	131
Signum Systems	www.signum.com	132
Swell Software	www.swellsoftware.com	38
TASKING	www.tasking.com/products	36
TechTools	www.tech-tools.com	134
Tektronix	www.tektronix.com/TLA600	23
Telelogic	www.telelogic.com	86
Telelogic	www.telelogic.com	138
Telelogic	www.telelogic.com	140
Tern Inc.	www.tern.com	132
Texas Instruments	www.ti.com/sc/tps701xx	13
The Math Works, Inc.	www.mathworks.com/espc	37
Treck Inc.	www.treck.com	79
U.S. Software Corp.	www.ussw.com	73
UC Berkeley	ptolemy.eecs.berkeley.edu/jobs	130
Vesta Technology	www.sbc2000.com	133
WebPRN.com	www.webPRN.com	17ab
White Mountain DSP	www.wmdsp.com	120
Wind River Systems	www.wrs.com	14,15
Working Engine Inc.	www.workingEngines.com	130
Waferscale	www.waferscale.com	80
XILINX, Inc.	www.xilinx.com	31
Z World	www.zworld.com	132

Jack G. Ganssle

In Praise of Kids These Days

Some of my recent columns ("A Look Back," December 1999, p. 121; "Cores, Cards and Tubes," January 2000, p. 97; and "Reality Bites," February 2000, p. 113) brought an enormous response from readers. More than a few e-mails expressed concern that today's young engineers work at such high levels of abstraction that they don't understand a system's low-level details. Reasoning such as "when you sling a million transistors at a time, you never learn the art of electronics" pervaded the responses. My immediate, geriatric response was to agree, wholeheartedly, with passion.

But maybe I'm wrong.

Sometimes the hardest part of getting older is learning to adapt to new ways. Even tougher is admitting that the new ways might indeed be better than those with which we're so comfortable.

For example, the U.S. markets have grown at an amazing rate for almost a decade. Some pundits gleefully predict that new business models have recession-proofed the economy. After weathering a series of boom and bust cycles over my career, I find myself disagreeing. But maybe my view of this is too much tempered by personal experience. Is it possible that things have indeed changed?

Take the .com companies—firms that annoy every business sense I have. Don't these folks have to make a profit sooner or later? But maybe my model of the business world is flawed. Fact is, Netscape built a business by giving products away for free. Perhaps a sea-change has truly shifted the way business works. When investors give a $20 billion company (Microsoft) a half-trillion dollar valuation, surely this is a sign that old business conventions are, at least, under attack.

Young engineers may look at vacuum tubes the way we look at button hooks, but they have enough on their plate contending with complex designs to pay attention to historical detritus.

What about the open-source movement? Here's a freebie approach that even half-trillion dollar outfits seem to fear and leaves me dazed and confused.

And so here's a sort of a parable, a story from the olden days, with an attempt to draw some meaning in terms of how things have changed.

It was tough being a teenaged geek. While the rest of my school chums pursued football and fast cars, dreams of electronics filled my brain and occupied too many waking hours. Girls, sure, they were important—so long as they didn't interfere with the important stuff.

(Now, of course, there are indeed girl-geeks as well as boy-geeks. The younger engineers I meet, though, generally lead much more balanced lives.)

Why were computers so fascinating in those nascently pubescent years? What compelled me, and many of my likewise geek-friends, to learn everything we could about these inanimate beasts? Dreams, surely, since none of us had even the slightest exposure to real computers, other than to watch their whirling tape drives spin in science-fiction movies. It wasn't until junior year in high school that I first actually used a computer. A friend's school had a timesharing account on a Honeywell mainframe, a machine we never saw, located miles away and fed by dozens of phone lines. We used an ASR-33 teletype in the classroom, connected by a 110-baud acoustic modem to the remote machine.

For $4 an hour we could program to our hearts content. Of course, at the time I was making $1.60 an hour as the lowest of engineering technicians in a space company doing Apollo subcontracts. The company's computer needs were completely serviced by an IBM 1132 rented by the hour, located off-premises, to which only one VP

But for me the ideas flowed faster than construction speed. I trashed each project as the next new concept improved on its predecessor.

had access. Once, noticing my interest, he took me along to see the machine, a near-religious experience for me.

(Thank God the computer priesthood disappeared; computers are now an equalizing force, one that has even been partially credited with the collapse of the Soviet empire.)

The $4 an hour charge was so much money that I spent an enormous number of hours off-line, checking and rechecking my code before running the paper tape into the machine. But that teletype introduced me to the cryptic world of command line compilers, to one of what eventually became many odd dialects of Fortran, and above all to the discipline of logical thinking. I learned what literal creatures these computers were, demanding above all completeness and correctness. Back then the "do what I mean, not what I wrote" compiler just did not exist.

(For 30 years we've heard predictions of meta-languages that will eliminate programming. It seems that a program is the smallest complete expression of what an application should do; programming is, in fact, an exercise in specifying the system, and as such will probably never go away.)

I wanted more computer time, but couldn't afford to keep pouring $4 an hour into the school's coffers. Clearly I needed my own machine. So started a succession of designs. Never having heard of assembly language I supposed the machine somehow ran Fortran directly, so I created a very simple subset that my crude logic circuits could execute. As soon as one design was done I'd have a better idea and start on the next version. One was bit-serial, with few parallel paths, to minimize hardware. Others were more conventional.

I actually started to build one or two of these designs. Before becoming a tech I had been a janitor at the same company. Those fat Apollo years led to a rule that we janitors (all unpaid young sons of company engineers) could keep all of the components found on the lab's floor, so my stockpile overflowed with resistors and RTL (resistor-transistor-logic, a prelude to TTL) ICs. The engineers liked to build 3D circuit prototypes, parts sprouting from a vectorboard up towards the sky. They'd junk these creations after working out the bugs; we'd snap them up and unsolder the components, sort them into our own personal bins at home, and then use them as the basis for our own experiments, science fair projects, and eventually computers.

But for me the ideas flowed faster than construction speed. I trashed each project as the next new concept improved on its predecessor.

(Then as now, building things out of hardware was difficult. But with IP cores, VHDL, and huge programmable logic arrays, much of hardware construction parallels writing code as we edit and compile abstract symbols rather than solder gates together.)

With college came accounts on the university's various machines and an obsession that permitted no time for class or other extraneous events. I discovered assembly language and operating systems. With as much computer time as I needed at the college (my accounts being, ah, "enhanced" by gaining a rather deeper knowledge of the OS's flaws) personal computer lust made little sense. But each of the seven deadly sins all defy logic, so my quest to obtain a machine continued to rage.

An article in *Popular Electronics* (as I recall) showed how an engineer obtained a junked IBM 7090. He rebuilt the monster, somehow figuring out how to rewire massive cable bundles that had been cut with an axe as the previous owner removed the machine. His biggest expense was air conditioning. Fortuitously, just then the university announced it was getting rid of its 7094, which covered several thousand square feet of floor space. With dreams of mainframe ownership I submitted a bid (zero dollars but a good home), though, since I lived in a dorm room, I had no idea where the thing could go. They wisely donated it to a smaller college. Drats, foiled again!

With dreams of mainframe ownership I submitted a bid (zero dollars but a good home), though, since I lived in a dorm room, I had no idea where the thing could go. They wisely donated it to a smaller college. Drats, foiled again!

Around this time I met an engineer who had managed to purchase, for an astronomical amount of money, a PDP-8 minicomputer with a single tape drive. DEC made thousands of these machines in many different configurations. My friend had dreams of building a small timesharing business around the computer, but the dreams turned to dust as within a few years timesharing became as obsolete as vacuum tubes. Again, though, I saw that personal computer ownership was a possible dream.

No one seemed ready to donate a computer, and the costs of buying one were simply beyond consideration, particularly when working many hours a week just to pay tuition. The obvious solution was to settle on a design and actually build a machine—any machine.

Over the years my parts collection grew past the RTL stage to quite a wide range of TTL devices. So design number eight, the last of these efforts, was based on a TTL CPU. God, I wanted a 16-bit machine, but that required too many parts, too much money, and far too much construction time. Eight bits seemed rather primitive (funny, since now almost 30 years later eight bits is such a huge part of the market), so I settled on a 12-bit architecture.

Now understanding the concept of machine language, program counters, and instruction registers, the design proceeded quickly and reasonably elegantly. My overriding concern was to keep costs down, leading to a rather simple instruction set.

For those of you who remember the TTL days, I used three 74181 ALU chips (now obsolete, though still available in their LS guise) as the central computational elements. These devices took in two 4-bit arguments and did one of 16 operations (add, subtract, AND, OR, and so on) on them, yielding a 4-bit result with carry. Four bits of my instruction word went directly into the 74181's control inputs, letting the computer do any math operation supported by these parts.

By now it was the very early '70s and Intel had started to make its mark in the memory world with the invention of the DRAM and with manufacturing various SRAMs. A surplus shop sold out-of-tolerance SRAMs cheaply. Thirty-six chips gave me 768 words of memory.

The girlfriend had unsurprisingly dumped me by now, leaving plenty of time for wiring. Circuit boards were too expensive, as were sockets for the ICs, leading to an implementation of several hundred TTL parts placed on about an acre of vectorboards, with wires (telephone wire from a source one didn't ask too many questions of) soldered to each pin.

The front panel had an array of toggle switches and LEDs that monitored and controlled the instruction register and a few other critical components. At first I had no real I/O device, so I would enter and run programs in binary on the switches.

By now I was sharing a corner of an apartment with some others, but the computer work required so much of my space allotment that for the next two years I moved onto the porch.

Debugging the completed machine took months of part-time work. Not owning an oscilloscope, my main tool was a voltmeter. Not a problem, since the machine was fully static. Running the clock at under 1Hz, I could track circuit operation quite easily without fancy gear.

At some point the machine worked, but was useless without a terminal. Months of scrounging turned up an old Model 15 Teletype, a "weather machine" that incorporated lots of strange weather map symbols as well as normal letters and numbers. These beasts spoke Baudot, a 5-bit code, instead of ASCII.

The Teletype weighed several hundred pounds and made a thunderous racket as its half-horse motor spun, sequencing a baffling array of levers and cams. But how could the computer communicate with the Teletype's serial data stream? I eventually wrote a program that "bit banged," transmitting and receiving data by precisely sequencing streams of bits between machines.

All of this code was entered on the switches, a word at a time. Happily, Intel invented the 1702 EPROM about this time. The lab where I worked allowed me to borrow two EPROMs, giving me 256 words of nonvolatile storage, enough to hold a very primitive command line OS that accepted programs from the Teletype's paper tape reader, and that interacted with me, the programmer.

So, what did I do with this finally completed computer? Nothing. Nada. Zip. In an old Star Trek episode, Spock, finally defeated in his quest for a Vulcan bride, tells the winner that "Wanting is often much better than having. It's not logical but it's very often true."

The 8008 also appeared about this time, more or less making my creation a dinosaur. Life got busier as work demands increased until the computer became a dust collector. Its final home was a dumpster.

So looking back on this frenzy of

My hat is off to those who master the very latest, the most abstract forms of engineering. Therein lies the future.

design and construction, I wonder how it benefited me. Sure, there's no better way to learn everything about the internals of a computer. The process helped me master every nuance of decoding and executing instructions. But so what?

Listen to the technologies in the story of my homemade computer: Teletypes (obsolete), bit-banging code (if not obsolete, it should be), 1702s and 74181s (both the parts and the PMOS/straight-TTL technologies long gone), tape drives (history except in much more sophisticated guises), timesharing (gone), Fortran (dead except in legacy code), and Apollo (sadly, gone with all of many hopes and dreams). Expertise acquired in these dead arenas is today useless unless translated to today's technologies.

Very few people design computers now. Even embedded folks buy pre-fabbed CPUs and hang pre-fabbed peripherals around them. Instruction register? Who cares how that works?!

Engineering is the art of problem solving. We use toolkits appropriate to the technology and the times. No one builds processors out of discrete ICs simply because that approach is inferior to off-the-shelf LSI parts. Those of us with the older skill sets may have great insights into many problems. But too many of us lack the skills of today, like building systems with IP cores, C++, and even Java.

In the late '60s, K&E, the slide rule people, predicted domed cities and other amazing changes in just a few decades to come. They missed the fact that in just a few years their business would disappear, overtaken by the cheap pocket calculator.

We think of symphonic music as something that undergoes no change, that's trapped in the 18th century. But the fact is that Mozart practically invented the piano concerto, Beethoven shocked the world by writing the first choral symphony, and Charles Ives pioneered atonality. Even classical music evolves. Beethoven would have been an unknown if he composed in the style of Bach.

The one pattern I've observed about embedded systems is the constant drive towards higher levels of abstraction. Moore's Law is merely a subset of this observation; higher transistor counts let us build ever more complex, abstract systems. But in parallel, tools evolve and application needs increase. Every technological improvement distances us from low-level details, a good thing really, since this is the only way to build bigger and more capable systems.

An obvious example is C++, which brings with it large overhead and ever more distance from the hardware as we abstract more of the system into encapsulated objects that only the original designers understand. Like software ICs these objects simply perform a function; we don't need to look under the hood to use them.

And so, when we middle-aged old farts complain that younger engineers have missed so much by not knowing how an archaic bit of electronics or code works, we're attempting, in a way, to preserve a status quo that was never quite so static. The engineers of tomorrow will be tremendously adept at making very complex systems work using tools we cannot now imagine, and that we'll probably never really master.

Nothing is static, stasis is death. My hat is off to those who master the very latest, the most abstract forms of engineering. Therein lies the future. **esp**

Jack G. Ganssle is a lecturer and consultant on embedded development issues. He conducts seminars on embedded systems and helps companies with their embedded challenges. He founded two companies specializing in embedded systems. Contact him at jack@ganssle.com.

P.J. Plauger

Great Reckoning, Little Room

From William Shakespeare's *As You Like It*, act 3, scene 3:

TOUCHSTONE: When a man's verses cannot be understood, nor a man's good wit seconded with the forward child Understanding, it strikes a man more dead than a great reckoning in a little room.

Curious phrase, that. Many scholars believe it to be a not-so-veiled reference to the death of Christopher Marlowe, a rival poet/playright to Shakespeare. Marlowe died well before his prime, at the tender age of 29, ostensibly as a result of a knife fight in an inn. According to the only two men present at his death, he had spent the day with them, talking, dining, and drinking. Marlowe quarreled with them over "the reckoning" presented by the landlady and, in self defense, one of them stabbed him in the eye. Absent any other witnesses, the inquest accepted this bald account.

Was Shakespeare taking a postmortem jab, as it were, at his past competitor? Maybe. But if you like conspiracy theories, you'll find plenty of grist for your mill. There's Marlowe the Secret Service operative, facing the British equivalent of the Spanish Inquisition, yet taking time off to spend a day with a couple of low lifes from the same trade. There's the remarkable similarity in writing patterns between Marlowe and early Shakespeare—including plays by the latter apparently written after Marlowe's putative death. There's the general cloud of uncertainty about practically everything associated with the writings attributed to one Wm. Shake-speare (yes, spelled with a hyphen), perhaps the same guy as the actor, second-generation usurist, and accused plagiarist William Shakspere from Stratford Upon Avon. Or perhaps not.

From smoke-filled rooms to teapots hosting tempests, small enclosures often produce results with more impact than you might at first imagine.

Maybe Marlowe was taking a postmortem, as it were, jab at the guy who was usurping his fame. After four centuries, we'll probably never know. In any event, that phrase keeps ringing in my ear.

I have spent a lot of time, over the past 20 years, arguing about software standards in "little rooms" all around the world. Okay, some of them were hotel ballrooms that could host an affair for a few hundred people, with dancing. Replace the cocktails and dancers with Cokes and fellow techies, however, and the rooms *look* pretty small. Moreover, far too many of them were truly small, windowless, and airless. Traveling to Copenhagen and Tokyo on a regular basis may sound glamorous—and it does have its moments—but most of those exotic weeks abroad were spent haggling over esoteric details in a series of little rooms.

Nor did the reckoning seem to be about any great matter, at least in the beginning. Way back in 1980 nobody much cared about what we did. I joined the original Posix effort, in its infancy, mostly as a matter of self defense. AT&T kept churning out new versions of Unix, and those of us trying to compete with compatible offerings were continually off balance. A handful of tiny to smallish companies saw a standard as a bulwark against arbitrary changes from AT&T. For their part, AT&T was willing to go along. But it wasn't a very big market in those days. AT&T was reasonably cordial in its customary role as the 800-pound gorilla, and we all just got along well enough.

The C standardization effort came along a few years later, with slightly

By the mid 1990s, it was clear to all players that developing an international standard for a language like Java would involve a great deal of reckoning. And it would add up to a great reckoning in its own right.

different dynamics. Now there were a lot more players, some of them hefty companies indeed, such as IBM and Digital (remember DEC?). AT&T was once again an active player and a willing participant, but this time they carried much less economic clout. The C compiler business was spread among dozens of competing companies, not concentrated like Unix in the hands of a principal supplier and one or two tiny competitors. We on ANSI committee X3J11 had a clear sense that the reckoning was greater. The rooms were also not quite so little as in the early days of what became the Posix standardization effort. But X3J11 valued even-handedness and cordiality very highly—thanks in large measure by the example set by Jim Brodie, the original Chair. We got things done with a minimum of threat and bluster.

Then came the C++ standard. It started out in 1989 as ANSI committee X3J16, in many ways a carbon copy of the C effort. AT&T was the purveyor of cfront, the de facto standard for what constituted C++ at any given moment. And a small contingent of AT&T employees, Bjarne Stroustrup not the least among them, immediately became the de facto steering committee for X3J16. But things change rapidly these days. By the time the C++ Standard was finally approved in 1998, cfront was no longer a supported product. C++ is too complex a language to encourage dozens of distinct implementations like C, but it has spawned quite a subindustry of implementors who carry on the torch in place of AT&T.

Along the way, ISO committee WG21 took control from X3J16. (Aptly abbreviated to just J16, the ANSI committee is now just one technical contingent among half a dozen from different nations, with just one vote on each formal motion.) At the peak effort, in the mid 1990s, it was not unusual to see 60 to 80 people attend each of the three annual weeklong meetings held in different venues around the world. Not such small rooms. And a distinctly greater reckoning was at stake, given the exploding demand for software developers that continues to this day.

Moreover, the international community had wised up in the process. They remembered with little fondness the days when an IBM or AT&T called the tune. They were still struggling with the economic clout of the American software industry, speaking through ANSI and IEEE most often, effectively dictating the specifications for important products and software-development tools. They became ever more determined to have a say in the reckoning that came out of those not-so-little rooms.

Dead reckoning

And that's how the software standards biz was happily evolving, at least until Java came along. By the mid 1990s, it was clear to all players that developing an international standard for a language like Java would involve a great deal of reckoning. And it would add up to a great reckoning in its own right. Standards are now recognized as a powerful leveling force, between large corporations as well as between nations. From the outset, Sun Microsystems retained tighter control over the Java specification, and even the Java name, than AT&T had ever done with Unix. It was soon clear to all players that Sun would not lightly cede technical control over Java to some committee of techies, even as they paid lip service to the notion that they would tolerate, even welcome, independent implementations of Java.

Sun cleverly preempted the usual process of standardization as early as 1996. They invoked a process intended for use by noncommercial standards organizations, such as IEEE, as a vehicle for controlling development of an ISO standard for Java wholly within Sun. I'll skip some of the gory details here, since I recently wrote on this topic for our sister publication, the *C/C++ Users Journal* (see "Standard C/C++: Java Standard Time," *CUJ*, January 2000). In brief, Sun won permission from ISO to spend two years developing the Java standard themselves. No meetings in little rooms, at least not outside Sun. At the end of that period, they requested and got an extension. Months later, Sun announced that they had changed their minds. They were not going to submit a Java standard to ISO after all.

Next thing you knew, Sun had approached ECMA, an industry consortium based in Europe that produces computing standards, to sponsor the development of a Java standard. Sun once again proposed to deliver the documents pretty much in their final state. And once again, after months of uncertainty and delay, Sun changed course. Just when they were supposed to deliver up the goods, they balked at the terms laid down by ECMA. Despite their repeated claim of support for an independent Java standard, Sun could not bring themselves to give up the control that such a document demands. Back to square one.

This time even the apologists for Sun joined in the rumblings of discontent. The upshot was that ECMA called a meeting of TC41, their Java technical committee, last January in Raleigh-Durham, NC. Its purpose was to explore ways to develop a Java standard even without the cooperation of Sun. The optimists were talking about producing a standard in as little as a year, presumably using materials found around the home. The pessimists wanted at least to begin the process.

I am pleased to report that, in two decades of standardization effort, I have yet to witness a single knife fight. Pete told me that there wasn't even a snowball thrown in anger during the TC41 meeting.

But the weather didn't cooperate. The attendees ended up snowbound for days by the worst storm to hit the Carolinas in many a year. My company sent Pete Becker. He was carrying a copy of our Jcore Library Reference, as our potential contribution to the effort. Luckily, he was also carrying his laptop. He spent the first couple of days holed up in his hotel room doing work he could have done back at the ranch. The meeting ended up lasting just one day instead of the scheduled three and a half. And the President of TC41 never did show up. His airline got him as far as Nashville before giving up.

The abbreviated TC41 meeting had an equally abbreviated outcome. All the committee agreed to do was explore whether they could produce a kind of commentary on existing Sun documents describing Java. Sun did send a representative, a good friend of mine from the ISO SC22 days. John Hill is an old pro at standards work. He was careful to deliver the message that Sun is "no less committed to standardizing Java" now than in the past. For my part, I have to characterize the end result as a small reckoning in a smaller room.

How dead is dead?

Here I should observe that my running Shakespearean analogy is not perfect. I am pleased to report that, in two decades of standardization effort, I have yet to witness a single knife fight. Pete told me that there wasn't even a snowball thrown in anger during the TC41 meeting. (I confess, however, that I have been present at meetings where I was grateful that some overwrought attendees did not have ready access to automatic weapons.) Murder and intimidation can achieve consensus in the short run, when dealing with individuals. But when every side can send reinforcements to future meetings, it's best to work toward compromises that are enduring. See how much progress we've made in international relations since the beginning of the twentieth century?

And yet, that pronouncement by Touchstone still resonates with me. It has that oxymoronic phrase "more dead," so typical of Shakespeare. (Grammarians can preach the illogic of an augmented superlative 'til the cows come home, but we all get Mark Antony's drift when he accuses Brutus of inflicting on Julius Caesar "the most unkindest cut of all.") How can you be more dead than dead? Well, maybe if you weren't really struck dead while arguing about that great reckoning in a little room. If your apparent murder was just a contrivance to get the Star Chamber off your case, you could nip off to Italy until things cooled down. Later, you could come back to England in disguise and pick up again your career as a playwright—under an assumed name, of course.

Under those circumstances, death takes on a more symbolic meaning. You suffer the lesser death of your identity, to be sure. You're reduced to being just a ghost writer for someone else, someone you can't help but perceive as a lesser talent. You can get even by inserting a few double entendres and in-jokes into your writing. It is probably significant that Touchstone's curious pronouncement is part of a general campaign to woo the simple country girl Audrey away from her dolt of a boyfriend. His name is Will. Evidence exists that a certain eponymous actor from Stratford Upon Avon played this role in an early production.

If you buy all this, then Touchstone's preamble makes a kind of sense. The worst death left to a ghost writer is to be unread. Lucky for us all, *As You Like It* has survived for four centuries and is still widely enjoyed. The next worst death is to be misunderstood. What if future audiences don't get the message? Lucky for the author of the play, many of us get the insights into human nature so neatly woven into what passes for an evening's light entertainment. And if you believe the appropriate subset of conspiracy theorists, we may even get Touchstone's veiled joke after all these years.

But whoever wrote that little speech, it rings a bell with me. As a writer or as an attendee at standards meetings, I want nothing more than to get my viewpoint across. You don't have to buy it outright, you don't even have to agree with it. But I want you to get the message before you make any decisions that affect me. I want my "verses" to be understood, whatever form they take. And I want that forward child Understanding to second whatever good wit I may from time to time exhibit. You can't build an informed consensus without getting the information across.

It was thus with some sadness that I failed to make myself understood recently. The incident occurred in conjunction with some of the commentary that followed the foreshortened TC41 meeting. What triggered it was a posting by Rex Jaeschke on one of the Java reflectors, summarizing the outcome of the meeting. Rex is another good friend from past standards work, primarily the C Standard in this case. He attended the meeting on behalf of Microsoft, who is paying him to represent them. It was Rex, in fact, who evidently championed the notion of producing a commentary rather than a full standard, in the interest of

8051

68HC11

Z80

Z180

64180

683xx

68K

68HC16

6809

6801/03

6502/01

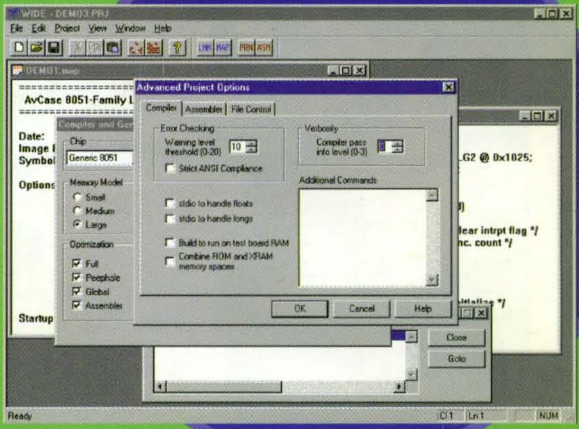
65816

And Many More

Not Just Any C Compiler The C Compiler

Avocet C Compiler, Macro Assembler, Simulator Package

Use Avocet tools and relax. With over 20 years of development expertise behind our software we are the wise choice for <u>your</u> embedded projects. One phone call to Avocet and you'll find the customer service you want, the tools you need, and the technical support you expect. Our C Compiler, Macro Assembler, Simulator package is the tool set you should not be without.



Chip: Generic 8051

Memory Model: ○ Small ○ Medium ● Large

Optimization: ☑ Full ☑ Peephole ☑ Global ☑ Assembler

Error Checking: Warning level threshold (0-20) 10

□ Strict ANSI Compliance

Verbosity: Compiler pass info level (0-3) 0

Additional Commands

□ stdio to handle floats

□ stdio to handle longs

□ Build to run on test board RAM

□ Combine ROM and XRAM memory spaces

LG2 @ 0x1025;

lear intrpt flag */

nc. count */

itialize */

- Includes full-featured Windows IDE, C Compiler, Macro Assembler, Linker, Simulator
- Accepts any combination of C, assembly, or relocatable object code files as input
- Generates highly optimized PROMable code
- Compiles all the way to hex code with a single command
- Supports bank switching
- Full source code for C libraries including floating point available
- Accepts single lines or blocks of in-line assembly code

Avocet. The Complete Tool Pack.

- Over 20 Years of Experience with Embedded Development Tools
- Compilers, Assemblers, Simulators, RTOS, In-Circuit Emulators for Hundreds of Processors
- Expert Technical Support Staff
- We will get you any tool you need

(800) 448-8500

www.avocetsystems.com

sales@avocetsystems.com

**Support for Hundreds of Processors.
Hundreds of Products.
One Phone Call.
Avocet.**

AVOCET SYSTEMS, INC