october 2000 www.embedded.com volume 13, number 11

Embedded Systems Programming®

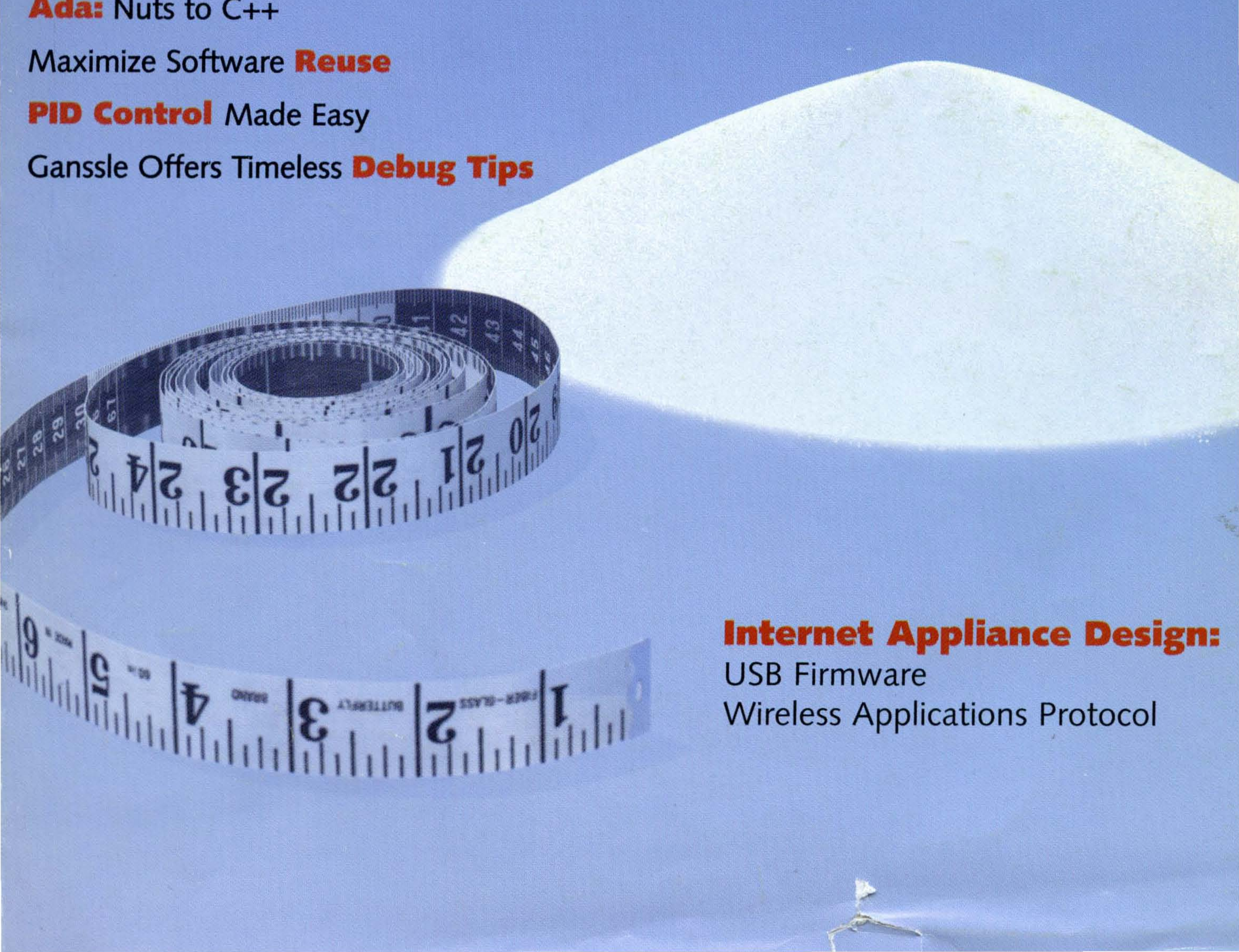
Avoid Mismeasurement
Use Unit Names

Ada: Nuts to C++

Maximize Software **Reuse**

PID Control Made Easy

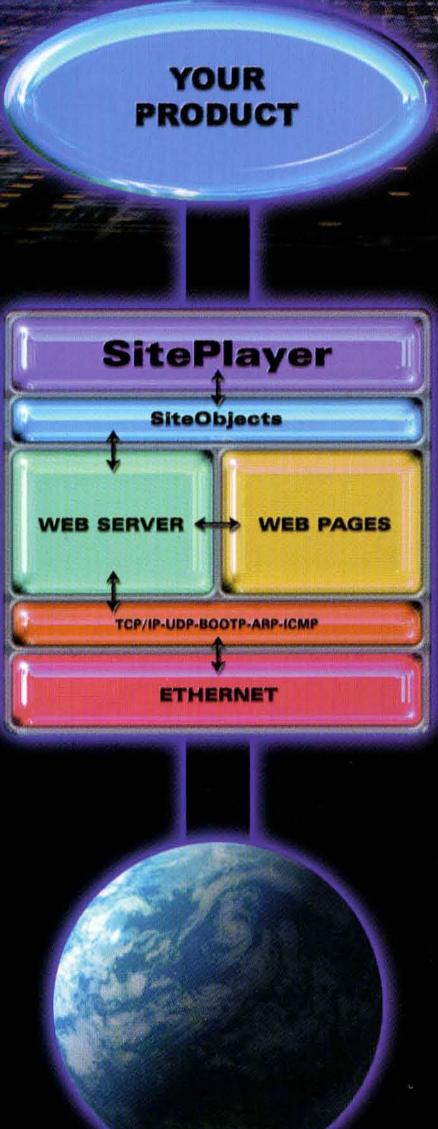
Ganssle Offers Timeless **Debug Tips**



Internet Appliance Design:
USB Firmware
Wireless Applications Protocol

SitePlayer™

EMBEDDED INTERNET SERVER



You've got a device. You want your device on the web. You don't have a lot of time. You don't want a lot of headaches. You don't want to spend a lot of money. You just want it to work.

You want it to be directly usable, with simple installation for your customers. No gateways, extra hardware or special wiring required. You just want Ethernet with installation as tough as plugging into a hub and browsing away.

Welcome to the world of SitePlayer. SitePlayer is a web server coprocessor module, complete with ARP, ICMP, BOOTP, UDP, TCP/IP and HTTP protocols, web page storage, 10baseT Ethernet interface, all in one square inch!

You create the look and feel of your device using standard web page tools. SitePlayer even gives you a library of graphical switches, knobs and LEDs to give your product a professional look.

During development, simply download your new pages to SitePlayer over Ethernet. Updates in the field are just as easy.

Need space for fancy graphics, MP3s, video clips, animations, embedded user's manuals, or multiple languages? SitePlayer supports optional flash memory cards for multi-megabyte web sites.

Communication from your device to SitePlayer is easy. A serial port is all you need. That's it! Minimal code to write and you are up and running with a world wide graphical user interface. Don't believe it? Just buy the SitePlayer developer kit for only $199.95 and see for yourself. SitePlayer chip sets are available for high volume applications.

Whether you need to make 10 or 10 Million devices, SitePlayer is the simplest, fastest, most cost effective way for you and your customers to get your device on the web.

WWW.SITEPLAYER.COM

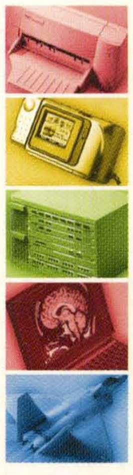
NETMEDIA INC. 10940 N STALLARD PLACE TUCSON ARIZONA 85737 t520.544.4567 f520.544.0800 SALES@NETMEDIA.COM

SitePlayer, the SitePlayer logo, and NetMedia are trademarks of NetMedia, Inc.

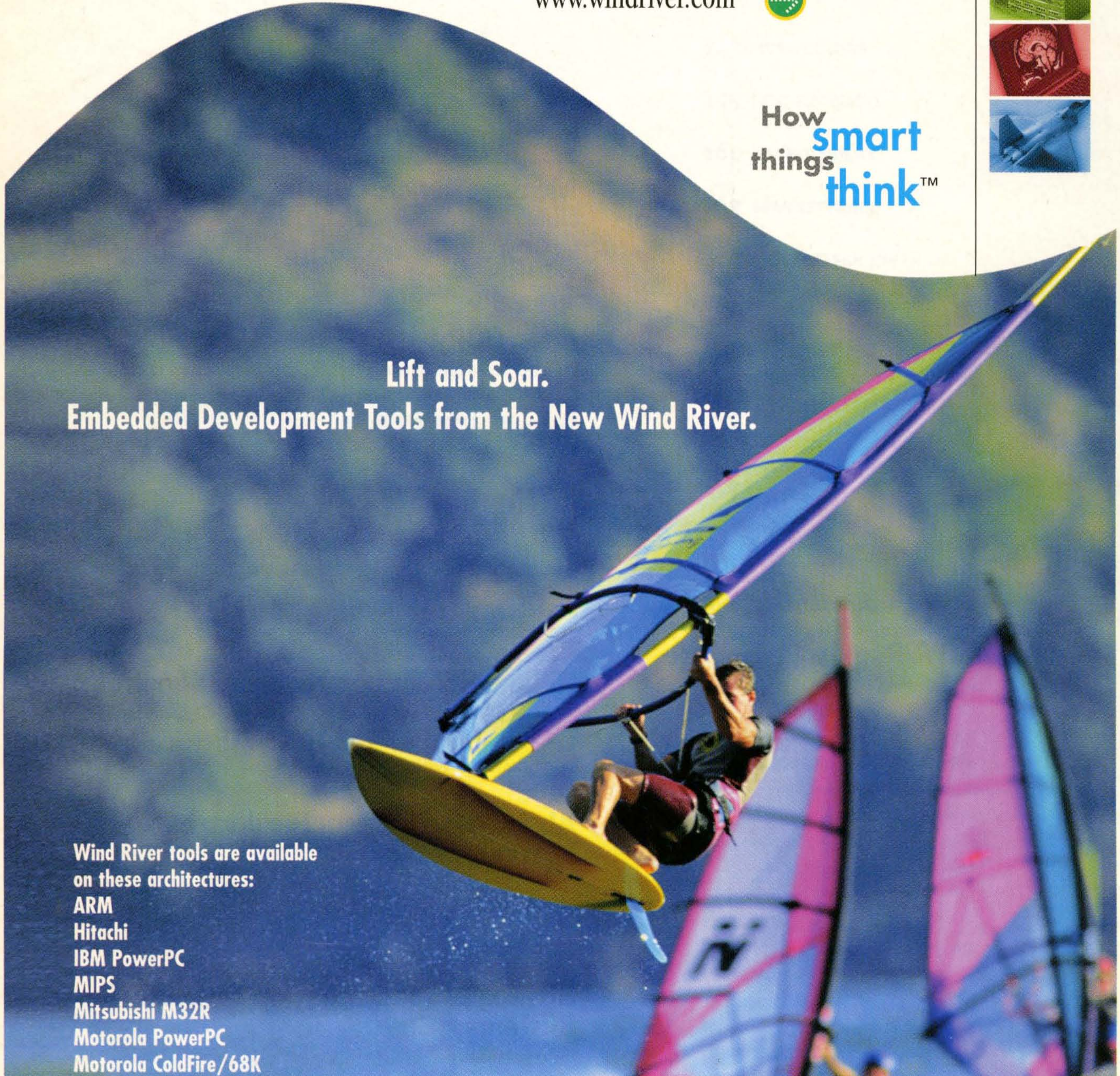
WindRiver®

Diab™C/C++/FastJ,™ RTA, visionICE, visionPROBE, SingleStep Solutions,™ SNiFF+,™ and Development Boards. Separately, they're Best in Class. Market Leaders. Powerful. Open. Scalable. Now unified under Wind River, these tools form complete solutions that lift your development project and make it soar, no matter what operating system you choose, all the while backed by Wind River's world class sales and support organization. To help your next embedded project take off, call 1-800-545-WIND or browse http://embeddedtools.windriver.com and register for a free tools evaluation.

www.windriver.com



How smart things think™



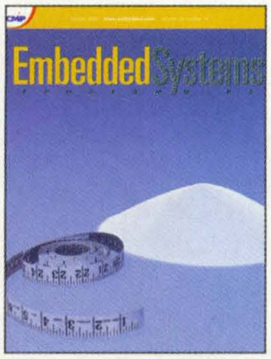
Lift and Soar.
Embedded Development Tools from the New Wind River.

Wind River tools are available on these architectures:
ARM
Hitachi
IBM PowerPC
MIPS
Mitsubishi M32R
Motorola PowerPC
Motorola ColdFire/68K
Motorola MCORE/DSP

Embedded Systems
PROGRAMMING

contents

OCTOBER 2000

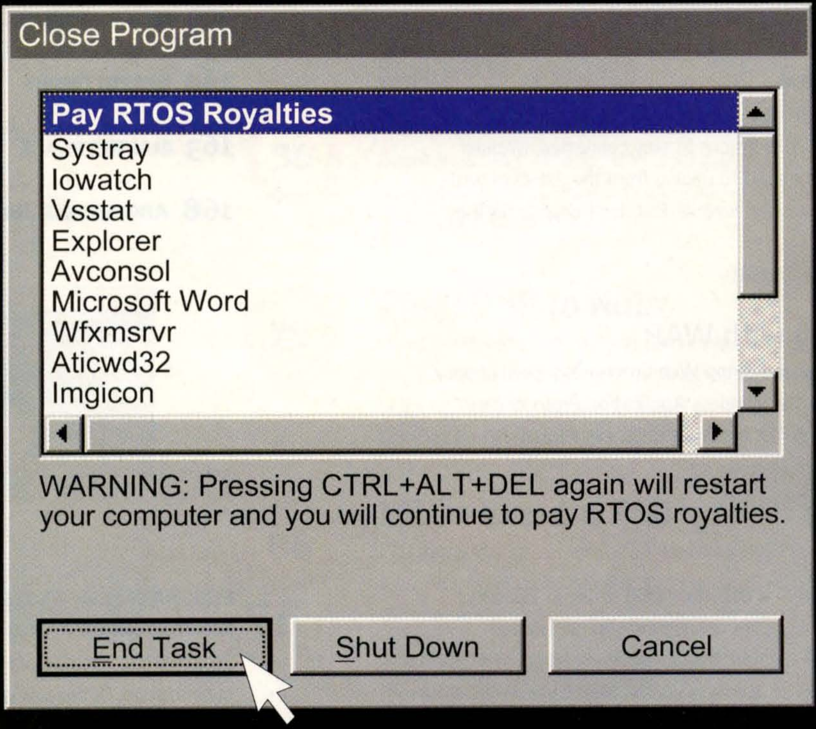


COVER

Do your neighbors look at you funny when you ask to borrow a foot of sugar? So would we. A naming convention might help you avoid mismeasurement and embarassment.

Cover illustration by Rupert Adley.

Wouldn't It Be Nice If It Were This Easy To Eliminate RTOS Royalties?



WARNING: Pressing CTRL+ALT+DEL again will restart your computer and you will continue to pay RTOS royalties.

It Is!

ThreadX® and INTEGRITY™ Royalty-free RTOS Solutions
EEMBC - Best Optimizing Compilers
MULTI® 2000 World-Class IDE

Green Hills
• SOFTWARE, INC. •

805.965.6044 • sales@ghs.com • www.ghs.com

EmbeddedSystems
PROGRAMMING

EDITORIAL DIRECTOR
Lindsey Vereen, lvereen@cmp.com

EDITOR IN CHIEF
Michael Barr, mbarr@cmp.com

MANAGING EDITOR
Felisa Yang, fyang@cmp.com

EDITOR AT LARGE
Alex Wolfe, awolfe@cmp.com

EDITORIAL ASSISTANT
Timothy Sullivan, tsullivan@cmp.com

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

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

PUBLISHER
Eric Berg, (415) 278-5220

NATIONAL SALES MANAGER
Stephen Corrick, (323) 512-4100

EASTERN REGIONAL SALES MANAGER
Damon Graff, (781) 839-1285

EASTERN SALES REPRESENTATIVE
Jared Grimm, (781) 839-1286

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

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

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

PRODUCTION COORDINATOR
Matthew Powell

CIRCULATION MANAGER
Jennifer Schuler

CIRCULATION ASSISTANT
Michael Falabella, mfalabel@cmp.com

CIRCULATION DIRECTOR
Susan Harper

SUBSCRIPTION CUSTOMER SERVICE
Toll free: (877) 676-9745
(847) 559-7307
Fax: (847) 291-4816
Omeda Communications
610 Academy Drive
Northbrook, IL 60062
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
Stella Valdez, (916) 983-6971

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

VICE PRESIDENT/ELECTRONICS
Donna J. Esposito

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

PRESIDENT/ELECTRONICS
Steve Weitzner

Consumer Audit BPA

CMP

Visit our Web site at
www.embedded.com

Michael Barr

Producer/Consumer Update

A well-known synchronization problem describes a single data buffer shared by two or more threads of execution. One or more producer threads write new data into the buffer, in parallel with one or more consumer threads that read data from it. Depending on the specifics, one or more semaphores may be needed to code the producer/consumer solution successfully.

Now well into its second decade, an essential element of *ESP*'s success has been the quality of its producers. In our case, the producers are columnists and feature authors. What generally makes our pieces so informative is that they're written by practicing embedded software developers just like you. The vast majority of the content we publish is written by full-time software developers and consultants who desire to share knowledge acquired through first-hand experience with their peers.

The results of our most recent editorial survey indicate that the average *ESP* reader has been programming for more than 17 years, developing embedded software for a decade, and reading the magazine for about seven years. And, yet, less than 1% of you have ever written even one article for us! If you think *ESP*'s been a pretty good read so far, just imagine how much more valuable of a community resource it could be if we could tap the rest of that potential.

The connection between readers and writers is an important one: readers of our magazine make the best *ESP* writers. That's because you are the ones building actual embedded/real-time systems that must be completed on time, meet their target price points, work reliably, and survive in field conditions. So you know how to do things right the first time, minimize system costs, and work around the little annoyances of the real world. In short, you've learned certain "tricks of the trade" that benefit you on a day-to-day basis—and could benefit your peers as well.

Sharing your knowledge with your peers—by writing an article or a book or speaking at a technical conference—is an excellent way to enhance your professional reputation, meet new friends, and learn about nuances of your own ideas that you hadn't considered before. In my particular case, sharing knowledge developed through my experiences as an embedded programmer and a consultant was how I ultimately found a career as a magazine editor. And, of course, writing an article is also a good way to earn some extra cash.

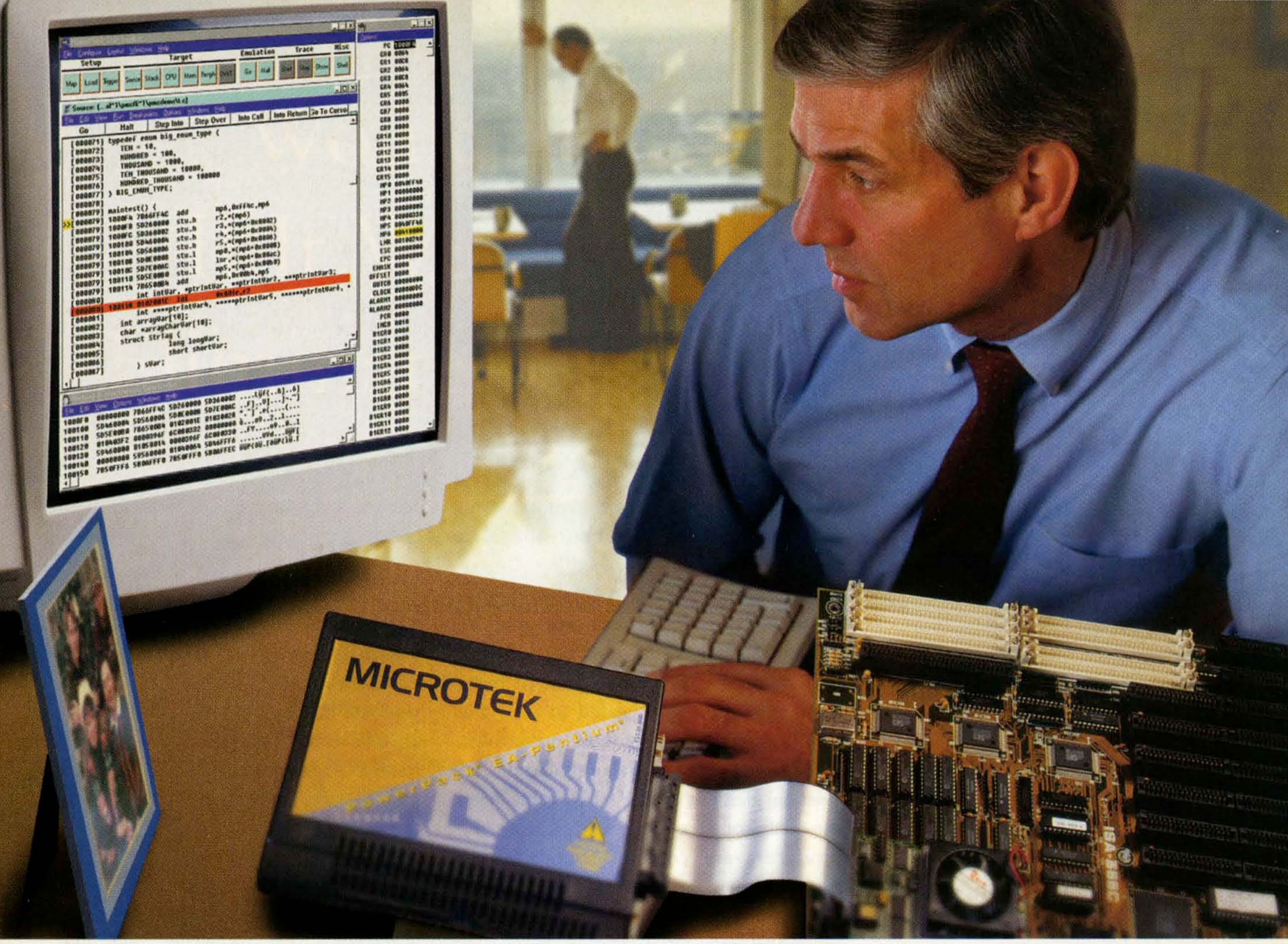
In addition to articles on any number of topics concerning the embedded software developer, we're also looking for a new columnist. An ideal columnist would be someone who likes to write, has lots of (at least sketchy) ideas for column topics rolling around in the back of their brain, and can find the time to write something new every month. Ideally, I'd like to find someone who could take up the "Real-Time" mantra abandoned by former *ESP* columnist and editor-in-chief Tyler Sperry.

You already know you're a nerd. So why not make it official? Write for *ESP*. If you're interested, check out our writer's guidelines at *www.embedded.com/wriguide.htm* and send your ideas directly to me. While helping make a name for yourself, you'll also be helping us make a dent in our own kind of producer/consumer problem.

[illegible]

mbarr@cmp.com

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



"Now, I have the tools to keep my project on time."

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

Readers to the Rescue

An astute reader, Aaron Berken, pointed out an error in the `PRE(x)` macro in my article, "Design by Contract for C Programmers" (July 2000, p. 100). The assertion test was inadvertently left out of the macro definition on page 101. Unfortunately, this omission rendered the macro useless. What I meant to write was:

```
#define PRE(x) { if (testPre &&\
  !(x)) reFailed(VALUE(__LINE__)\
  , __FILE__, #x); }
```

This change applies to both the macro definition on page 101 as well as the macro expansion on page 102. My apologies for the inaccuracy.

Steve Kapp
Embedded Real-Time
skapp@emrt.com

From reader to reader

I have some comments in response to Charlie Carothers' letter regarding the May article "A 'C' Test," (p. 119).

The expression `(60 * 60 * 24 * 365)UL` is not valid because `UL` is not a postfix operator that can be applied to any integer expression. There is no `UL` operator in the C language. Instead, `U` and/or `L` can be used as part of the token that specifies a literal integer constant. These suffixes must immediately follow the digits of the integer constant and are part of the constant itself, rather than a modifier that is applied to the constant. For example, `60UL` is a constant of type `unsigned long.` (See section 6.1.3 of the ANSI C standard.) I believe the C++ standard requires `malloc(0)` to return a non-zero pointer, but the ANSI C standard states in section 7.10.3: "If the size of the space requested is zero, the behavior is implementation-defined; the value returned shall be either a null pointer or a unique pointer." Therefore, you cannot count on either behavior if you want your C program to be portable. With implementations that do return a valid pointer, I would guess that the library actually does allocate some memory. (`malloc` typically allocates some extra overhead memory for internal bookkeeping uses.) However, the result is likely to be undefined behavior if the programmer actually tries to store something in this "zero-sized" memory block.

Rod Spade

Addressing the problem

There are a couple of bugs in the code that accompanies Michael Barr's article "Software-Based Memory Testing" (July 2000, p. 28).

1. The documentation in the code (`memtest.c`) suggests that if you want to use the algorithm to test memory that has a data bus wider than an 8 bits, you should change the `typedef` for `datum` to a type that is as wide as the data bus.

For a 16-bit memory device, you would change the `typedef` for `datum` to `typedef unsigned short datum;`. However, this presents a problem for the loops in the address line test because the initializer in many of the loops is `offset = sizeof(datum);`. In this case `offset` is 2.

The first access to memory should be at the address corresponding to the least significant non-zero address line. For 8-bit memory, this address is 1, but for 16-bit memory, this address is 2.

Since the variable `baseAddress` is of type `volatile datum *`, the expression `baseAddress[offset]` becomes `baseAddress[2]`, and since `datum` is an `unsigned short`, this corresponds to the datum at memory address 4, instead of the desired address of 2.

To fix this problem, the variable `offset` (or `testOffset`) should always start at 1.

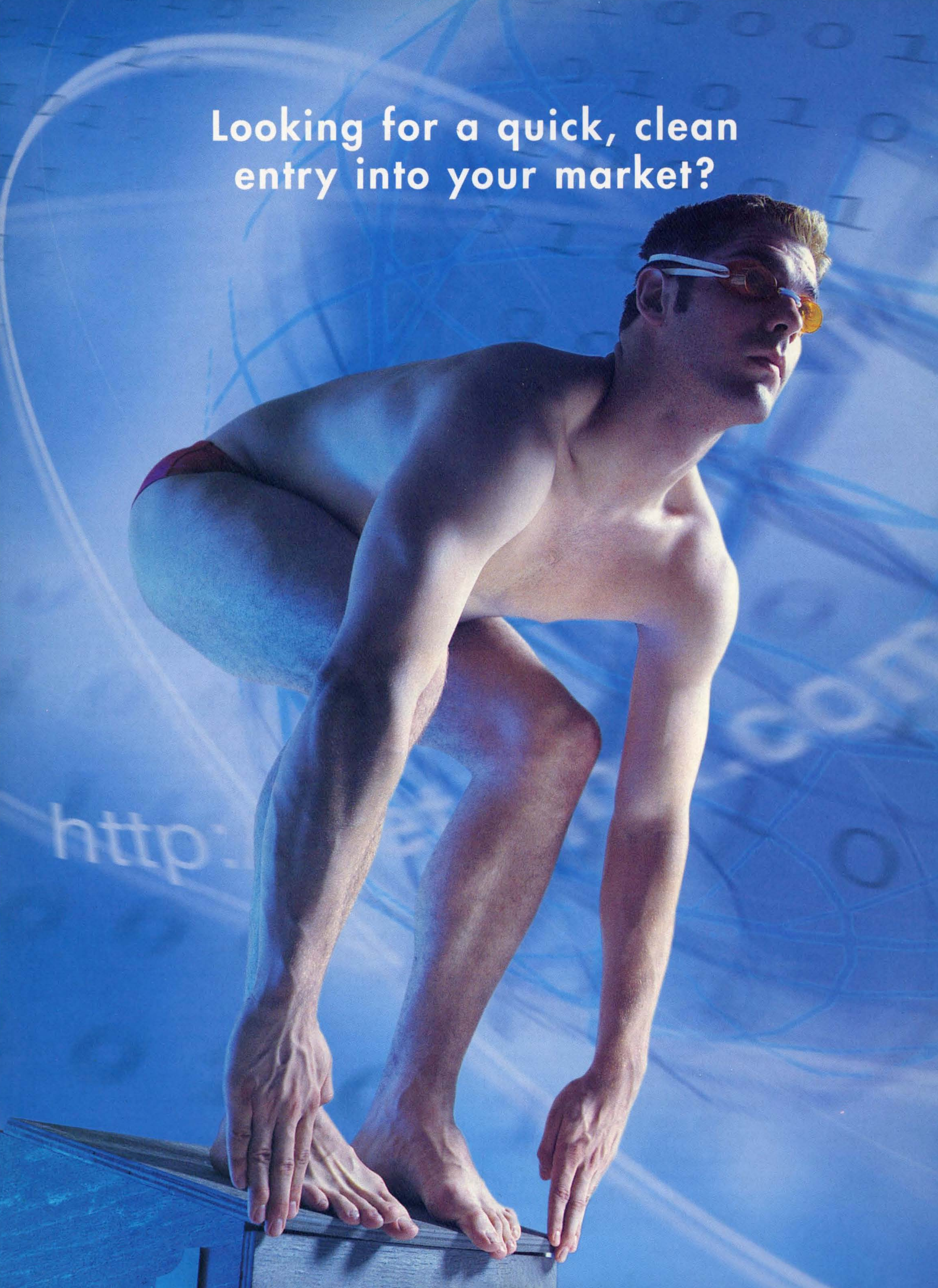
2. The test for address lines `stuck low` and `shorted` doesn't work when an address line is `stuck low`. When an address line is `stuck low`, a write to an address at some power of two will cause an unintended write to address 0. The current test only checks for the non-zero addresses, and will miss a bit stuck low.

Also, Mr. Barr referred to problems experienced with missing memory chips, indicating that it is possible to write to a memory location, and that the capacitance on the data lines may store the written value. I would add that this could only happen if the microprocessor cached or pipelined its instruction fetches, or if there were separate data buses for data and code. On all of the embedded systems that I've worked on, after writing a value to RAM, the micro must do at least one instruction fetch at its PC in order to even know to read back from that RAM location. In doing so, the address and data lines change between writing the value to RAM and reading it back.

William Moon
Eastman Kodak Company

ERRATA

In a recent article on Linux ("Linux, Interrupted," Thomas Besemer, August 2000, p. 49), the name of the author of the original CIO-CTR05 driver was omitted. That code was developed and is maintained by Dr. Warren Jasper, Associate Professor, North Carolina State University. Source code is provided with no warranty under the terms of the GNU Public License and derivatives of the driver, like Mr. Besemer's, must maintain the same licensing terms. —Ed.


Looking for a quick, clean entry into your market?
http:
.com

Start with the platform that lets you dive right in.

The QNX® realtime platform:
Download it FREE at get.qnx.com

Download the QNX realtime platform and immediately start building your next prototype. More than just an eval or demo, the download includes the full distribution of the QNX realtime platform and is yours free* for unlimited non-commercial use. Simply contact us when you're ready to create commercial product.

Once you get the QNX realtime platform, you'll know how to use it. That's because it's engineered from the ground up to support POSIX APIs and other standards. And built-in Linux compatibility means you can port most Linux apps.

You'll be more productive, too. The QNX realtime platform includes source for most components, including driver toolkits, OS utilities, TCP/IP stacks, startup code, games, and more. All on a market-proven RTOS with an architecture that's not only friendlier to work with, but boosts reliability and performance.

So download the QNX realtime platform and dive right in. You'll build feature-rich, reliable products faster—and get the competitive edge you need to win in your market.

What you get:

- **microkernel RTOS**
- **embeddable GUI**
- **rich set of OS modules**
- **hundreds of utilities and libraries**
- **drivers (disk, graphics, networking, USB, etc.)**
- **web browser**
- **media players and plug-ins, including RealPlayer™ and Flash Player**
- **complete self-hosted development environment, including GNU toolchain**
- **and much more...**

*All QNX technologies are free. Licensing fees for third-party software may apply.

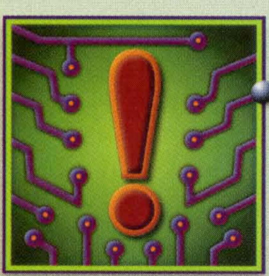
get.qnx.com
or call: 1 800 676-0566

QNX® Build a more reliable world™

Planning to add TCP/IP Ethernet to your design?

IF YOU CAN FIND A FASTER WAY, BE OUR GUEST

To see how NetSilicon impacts your development process, fold the page to join the arrows



Source hardware components
Build and test development board
Select RTOS and networking stack
Choose networking protocols
Choose development environment

The typical team of engineers needed to source and integrate hardware and software from multiple vendors.

So, your design requires Ethernet or Internet connectivity and you need a fast, cost-effective solution? NetSilicon's NET+Works™ adds TCP/IP Ethernet connectivity and eliminates design steps – cutting months off your development process. Our 32-bit, ARM-based system-on-chip, with development environment, RTOS, networking software and support gives you a complete, integrated solution. Work with us and see your design unfold in record time!

Visit our website at **www.netsilicon.com/es** to register for our free online seminars or to receive the "NET+Works Technical Overview" white paper.

800-243-2333
www.netsilicon.com

NetSilicon

STMicroelectronics Swallows Wafer

French semiconductor company STMicroelectronics has signed an agreement to acquire Waferscale Integration, a programmable system device (PSD) manufacturer. Before the deal, STMicroelectronics held a minority interest in Waferscale Integration. By investing $68 million more, it will achieve full ownership. WSI will continue to operate out of its Silicon Valley headquarters, but will function as a business unit of STMicroelectronics Memory Products Group.

ST, it would seem, sees an opportunity to use its extensive resources to facilitate the marketing of PSDs.

"We have been impressed by the performance of these devices," said Carlo Bozotti, corporate vice president and GM of ST's Memory Products Group, "as well as the advantages they offer embedded microcontroller applications."

"ST's vision, commitment, and resources will help us realize our product strategies," said Reza Kazerounian, who will manage the new business unit.

Besides acquiring the product portfolio, intellectual property, and technologies of a successful technology corporation, ST strengthened its presence in Israel and Silicon Valley, where both ST and WFI companies had held operations before the deal.

Wireless Market Grows

The wireless market is expanding. And it's not just hype. Three separate studies recently illustrated the prodigious influence wireless technology is having in various markets. One report declared that a global move to wireless data is underway, citing research that suggests that shipments of WAP-enabled handsets will rise from 33% of all handsets shipped in 1999 to 90% in 2004. In addition, another study claimed that broadband technologies will join the cellular market to drive the wireless communications integrated circuit into prominence in coming years. A third report stoked the fire, saying that one-third of surveyed database developers are now targeting wireless and 52% expressed interest in an infrastructure tool for creating mobile DBMs. Five hundred database developers were polled in this survey, which was carried out by Evans Data Corporation (www.evansdata.com). The first two studies referred to were conducted by Allied Business Intelligence (www.alliedworld.com).

Codevelopment tools on the rise

Continuing their impressive growth, codevelopment tool shipments will overtake those for logic analyzers and ICEs by 2003, a new study says. The study analyzed the market dynamics behind the current and expected growth of these market segments. It concluded that the growth of codevelopments tools was driven by intellectual property considerations, the growth of custom SOC technology, and time-to-market restrictions placed on complex hardware-software integration problems. Mentor graphics is the current market leader. Source: Electronics Market Forecasters Group, *www.electronic-forecast.com.*

1997-2003: worldwide shipments of embedded codevelopment tools

	1997	1998	1999
ICE/JTAG	148.5	159.5	172.3
Logic Analyzers	92.8	107.8	130.8
Device Programmers	65.9	64.7	68.2
ROM Emulators	7.9	8.1	8.2
Codevelopment Tools	25.1	46.1	75.6
Total	340.2	386.2	455.1

	2000	2003	CAGR
ICE/JTAG	186.4	236.5	8.2%
Logic Analyzers	156	261	19.3%
Device Programmers	71.3	82.6	4.6%
ROM Emulators	8.7	10.3	5.5%
Codevelopment Tools	108	274.2	42.8%
Total	530.4	564.6	20.6%

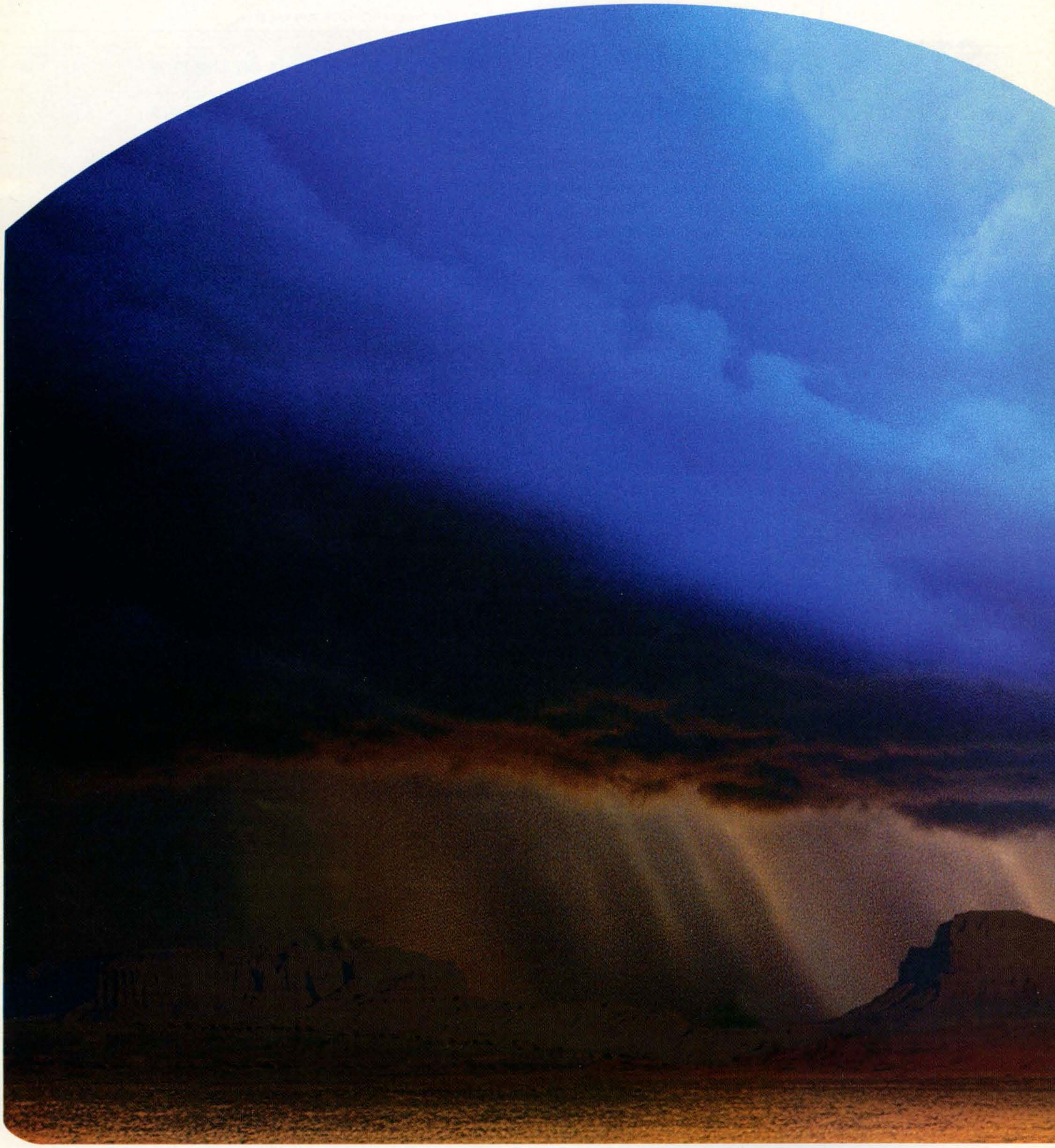
Briefly noted...

Toshiba has joined the Extend the Internet (ETI) alliance, an organization of more than 25 companies that aims to provide wireless internet products. ✱ **Mercury Marine Inc.** selected **Green Hills** to produce software that controls most of

NAMES IN THE NEWS

PeopleMation promoted **JAMES R. HANSEN** to the position of chief technology officer. ● **JOHNI CHAN** has been appointed vice president of engineering for I-Bus/Phoenix. ● Scenix announced that **DARREL M. BURNS** will join the company as vice president of engineering. ● Blue Wave Systems made **MALCOLM BROWNSELL** its chief operating officer. ● **PAUL MENCHINI** has joined Embedded Solutions Limited as its senior vice president of technology.

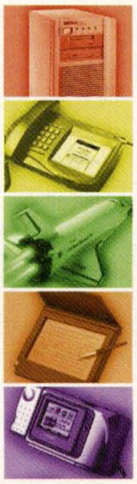
WindRiver®

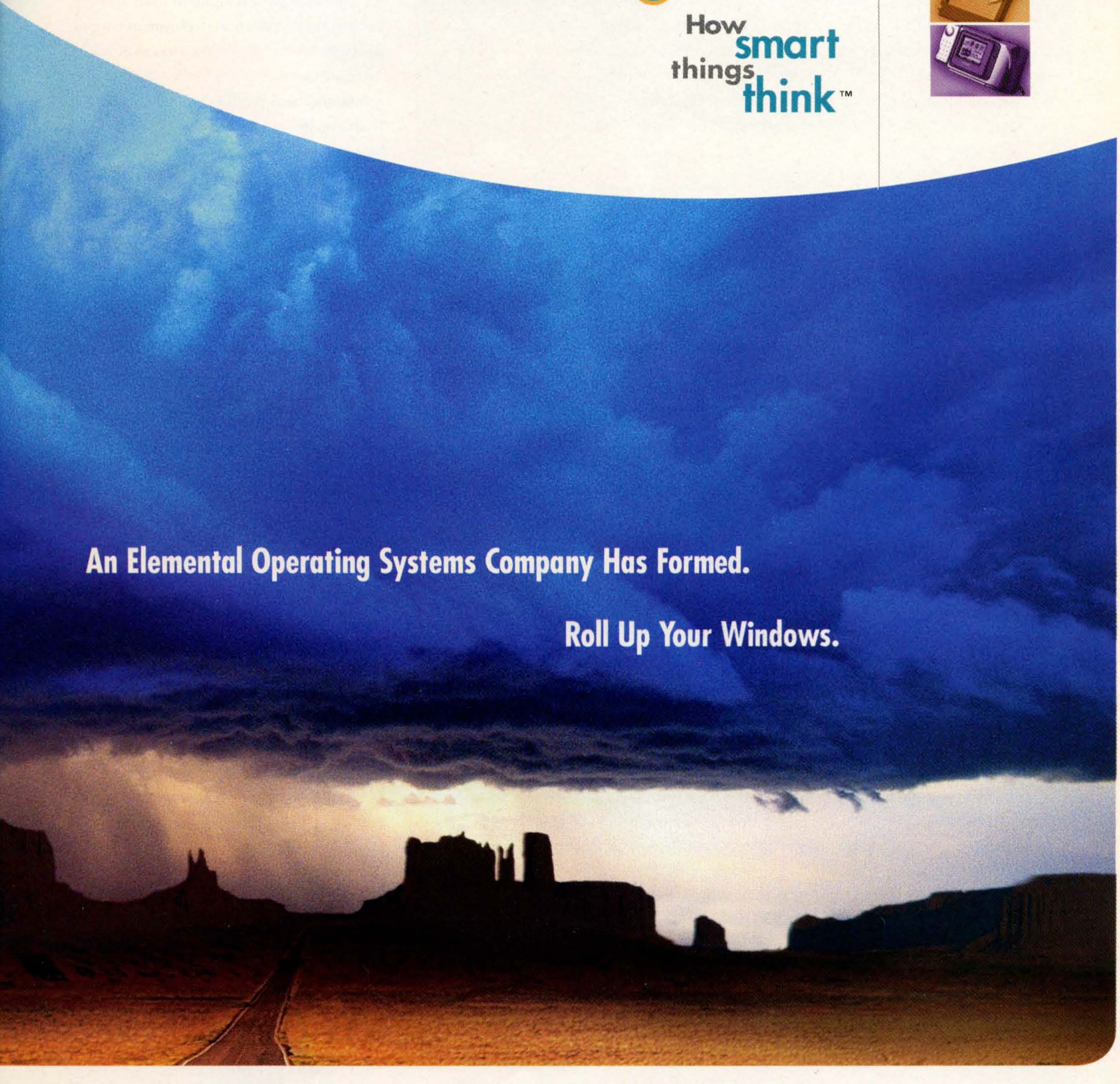


There's a new force sweeping across the landscape, ushering in a new era of embedded capability. An era of operating systems and development tools that are balanced, strong and consistent, like nature itself. What's behind this? The merger of Wind River Systems and Integrated Systems, Inc., including DIAB SDS, Doctor Design and TakeFive Software. Separately, these companies were leading the way in software for smart devices. Together, they will offer unprecedented expertise, drawing from a vast pool of resources to craft targeted solutions for their clients. So behold the power, and put it to work for you. Call Wind River at 1-800-545-WIND today.

www.windriver.com

How smart things think™





Software development tools, RTOS, and services markets to reach $2 billion by 2003

The world market for software development tools and RTOSes is forecast to exceed $918 million in 2000. Driven by the complexities of emerging designs, time-to-market considerations, and the lack of available engineering talent, the simulation-modeling segment of the embedded marketplace is experiencing the greatest growth. The services segment of the marketplace has become a strategic source of revenue to the embedded industry as vendors continue to find new marketing strategies for differentiating their products.

In 1999, shipments of embedded software development tools and RTOSes totaled $800.1 million, led by a rapid acceptance of simulation/modeling development tools. Design complexity, the need to thoroughly simulate complex systems (many with hardware in-loop), and the ability to automatically generate production quality code have resulted in significant growth for the simulation/marketing segment.

Source: Electronics Market Forecasters Group, *www.electronic-forecast.com.*

1999-2004 forecast: worldwide markets for software tools, RTOSes, and related services.

	1999	2000	2001	2002	2003	2004
RTOS	329.7	374.2	424.7	482.1	547.1	621
True IDE	86.9	97.3	109	122.1	136.7	153.1
Simulation-Modeling	170.6	212.7	287.15	387.65	523.32	706.5
SW Tools & IDE	212.9	234.2	265.81	301.65	342.42	388.16
Related Services	186.1	237	291.51	358.56	441.03	542.5
Total	986.2	1,155.4	1,378.2	1,665	1,990.6	2,411.7
Tools and RTOS alone	800.1	918.4	1,086.7	1,293.5	1,549.6	1,869.3

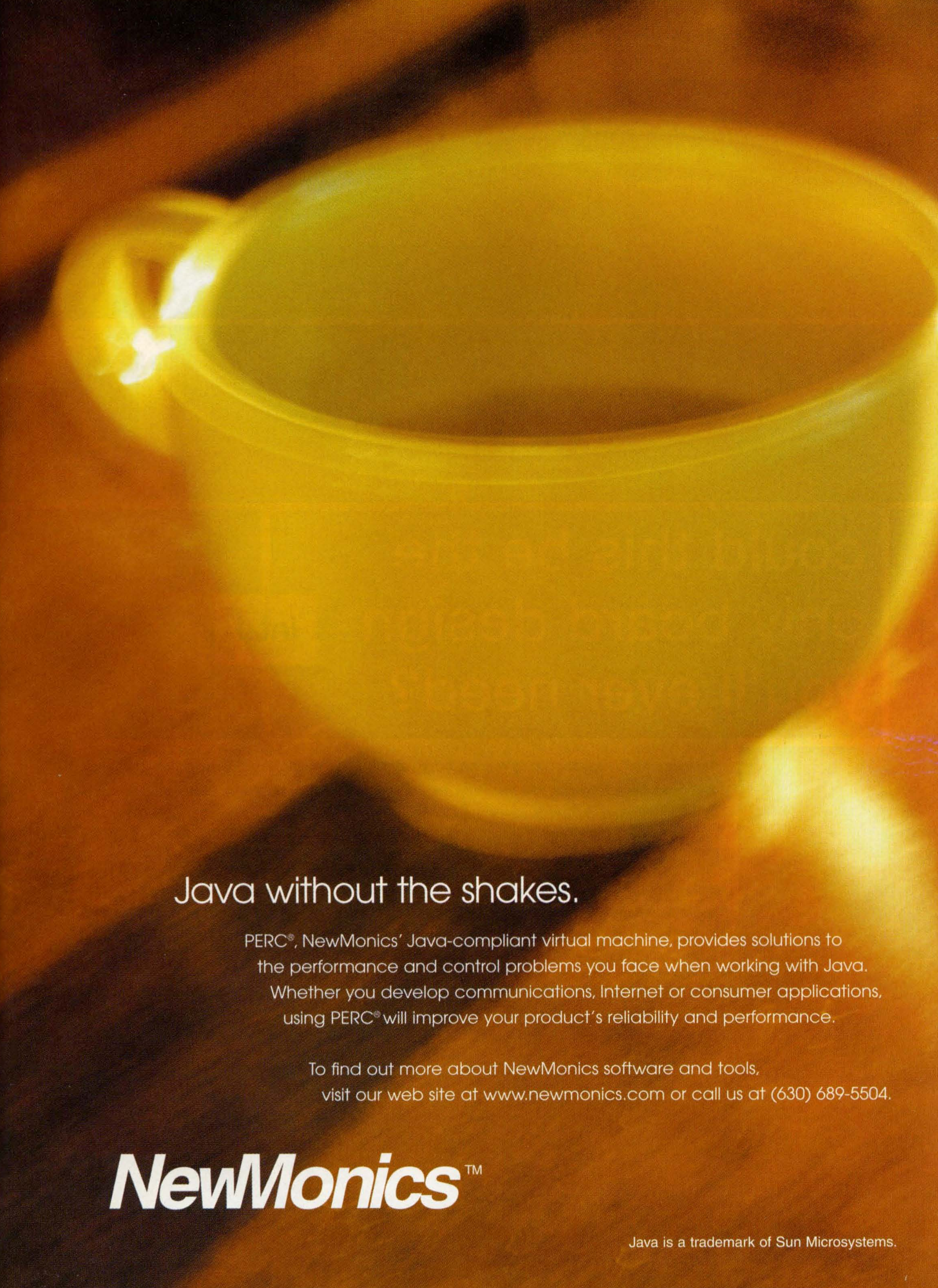
Geographic distribution of 1999 worldwide tool and RTOS shipments

North America	52.9%
Europe	29.2%
Asia/Australia	17.9%

THE NUMBERS GAME

Tundra Semiconductor announced a revenue increase of 62% for the second quarter of 2000. The company's net earnings increased 118%. ● **Virage Logic** announced an IPO of 3,750,000 shares priced at $12 per share. With the completion of the IPO, the company has raised gross proceeds of $49.5 million. ● **Voice Signal Technologies** secured $8.4 million in Series C equity financing from a group of investors led by **Stata Venture Partners**, an investment company that specializes in early- to mid-stage technology companies. ● **TriQuint Semiconductor** reported its financials for the second quarter of 2000. Revenues were up $70.6 million, an 85% increase over the revenues for the second quarter of 1999. Net income rose 280% to $16.4 million in comparison to the same period of 1999. ● Revenues at **Echelon Corporation** totaled $12.7 million, an increase of 19% over those from the same period in 1999. Their net loss for the quarter was $508,000, a decrease from the same quarter of 1999. ● **BP Microsystems** announced revenues of $11.9 million for the second quarter of 2000, a 125% increase over revenues for the same period last year.

the propulsion functions on Mercury Marine's entire line of outboard marine engines. ✳ **CAD-UL** and **Embedded Power Corporation** have teamed up to pair the RTXC RTOS with XDB for RTXC, an OS-aware version of CAD-UL's debugging tools. ✳ **Galil Motion Control** is relocating its headquarters to Rocklin, CA, an area just east of Sacramento. ✳ **Object Technology International** announced a distribution and support agreement with **MontaVista Software** for Linux-based development tools and embedded Java application deployment components. ✳ **Nortel Networks** has announced that it will use **Blue Wave Systems'** ComStruct communications product in a new Voice over IP system. ✳ **Technical Support,** an Omaha-based embedded systems engineering firm, has completed a project for **US Software** that expands their RTOS system technology into new telecommunications and networking markets. ✳ **2Wire** has chosen **M-Systems'** DiskOnChip as the local storage device for its family of HomePortal residential gateways. ✳ **inSilicon** has licensed the **ARM** MicroPack and PrimeCell Peripherals for use in combination with inSilicons' communications technology IP. ✳ **Wind River** has partnered with **QED** to optimize the Tornado development platform and the VxWorks OS for QED's RM7000 and RM5200 MIPS RISC processors. ✳ **LynuxWorks** announced that the LynxOS RTOS will now support **TimeSys Corporation's** TimeTrace tool. ✳ **STMicroelectronics** will use **Associated Computer Experts'** CoSy compiler development system for ST's embedded system compiler development. ✳ **VenturCom** has opened an office in West Sussex, England. ✳ **Triscend** has licensed the ARM7TDMI-S RISC microprocessor core for use in next generation ARM-powered CSoC products. ✳ **Lara Networks**, a company that produces processors for network applications, is setting up an Indian Design Center in Bangalore, India. ✳ **IBM** and **Sierra Monolithics** formed an agreement to use IBM's silicon germanium (SiGe) technology in SMI's microchips for optical networking equipment. ✳ Aisys announced its flagship product, a device driver design environment named DriveWay available as a free download to users of **Motorola's** 8-bit 68HCO8 family of microcontrollers. ✳ **Intrinsyc Software** and **MontaVista Software** have formed an alliance to produce Linux-based embedded systems.



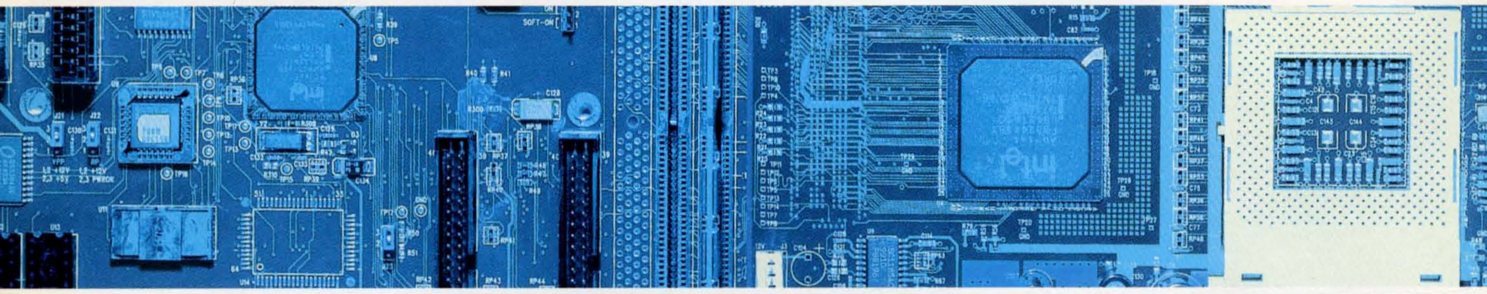
Java without the shakes.

PERC®, NewMonics' Java-compliant virtual machine, provides solutions to the performance and control problems you face when working with Java. Whether you develop communications, Internet or consumer applications, using PERC® will improve your product's reliability and performance.

To find out more about NewMonics software and tools, visit our web site at www.newmonics.com or call us at (630) 689-5504.

NewMonics™

Java is a trademark of Sun Microsystems.



could this be the
only board design
you'll ever need?

intel.

the new intel® scalable performance board design fits multiple processors for multiple purposes. which gives you the key to survival in today's economy: flexibility. thanks to intel, you can design a single board around the 810 or 440bx chipsets, then plug in different intel® pentium® III or celeron™ processors depending on the end use of the board. the result: time and money aren't spent designing a unique board for each processor. and, even better, one universal board can serve a wide range of customers and applied computing applications—ranging from communications infrastructure and appliances to pos terminals to industrial pcs. because in the surge economy, if you're not flexible, you're history. (download the design guide ➜ *http://developer.intel.com/design/info/070.htm*)

Jack W. Crenshaw

Moving Right Along

Jack reaches a temporary solution to the problem of minimization. Optimizations will come later.

When last seen, at the close of last month's episode, our hero was busily hammering away in his workshop, constructing software for his Zappo-Ray, portable anti-Brent function minimizer. His intention was to put together a set of snap-together modules that were both simple enough to read and understand, and effective enough to handle even tough extraterrestrial and pathological functions. The last module constructed was the stub version of the parabolic fit shown in Listing 1. (Incidentally, this listing corrects two errors that crept into the version presented last month. Bonus question for extra credit: what were the errors?)

Note the last two lines of `para_shrink()`, which swap the two endpoints of the bracketing region. These were put there to match the same two lines in `gold_shrink()`. They have to be there in `gold_shrink()`, to force it to bisect the regions on the left and right side on alternate calls. They don't really have to be there for `para_shrink()`, but it seemed a good idea to leave the output state in a condition similar to that which `gold_shrink()` would do.

At the close of last month's session, we plotted the three points being maintained by the minimizer, and noted the same bothersome effect I had observed earlier: the parabolic fit, if left unaided, tends to hang on to distant endpoints a lot longer than we would wish.

In the case of bisection methods, the algorithm has no choice but to pull those endpoints closer together. That's what bisection is all about. Either the new point is high enough to use as a new bracketing point, or it's low enough to bring the opposite point in. Either way, the endpoints move inexorably closer together. In fact, it's the distance between endpoints, rather than the stability of the midpoint, that we use as the criterion for deciding when we've converged. The method of parabolic fit doesn't work the same way, so twe can't guarantee that each endpoint will be alternately moved.

This tendency to hang onto distant points leads to two separate problems: First, the solution isn't very satisfactory because we're fitting a curve through points that are not well-conditioned (two points close together, third very far away). Second, we can't use the distance between bracketing points in our halt criterion. No doubt this problem is the reason Brent came up with a complex and inscrutable algorithm for deciding when the solution was good enough.

Our challenge this month is to come up with the logic to keep the parabolic fit under control, and to force the use of a bisection method to help ensure a stable and smooth descent to the minimum.

Where should we put this logic? Well, we can always put it in the driver function, external to function `para_shrink()`. I think it better to place it within `para_shrink()` itself. That is, we allow the function itself to decide whether the minimum it's recommending is acceptable, and have it return a Boolean depending on the result. To begin, define the logical variables:

```
#define FALSE 0
#define TRUE (!FALSE)
```

(I've put these definitions in my standard library header, jmath.h). Then alter `para_shrink()` to return a Boolean. For starters, we can just force it to always return `TRUE`.

Alternating methods

We seem to have established that letting the parabolic method run open loop is not very satisfying, since it tends not to pull in the endpoints. The next obvious approach might be to alternate between parabolic and bisection methods. We can do that by simply putting a counter in `para_shrink()`, and having it return `FALSE` every other pass. Not surprisingly, such an approach is going to take more function evaluations, but

For developers
who practice

safe

embedded
relationships.

To reduce development costs and speed time to market, you need a supportive, reliable embedded software partner with the right tools for the job. That partner is TASKING.

With TASKING, you'll find development tools for leading DSPs, including DSP56xxx and CARMEL. You'll have the best tools for leading 8-bit, 16-bit and 32-bit microprocessors and microcontrollers, including PowerPC, 68xxx, TriCore, C166, ST10, M16C, XA, SLE88, 8051, 196/296 and more. And you'll save even more time and money with integrated RTOS and embedded Internet solutions, as well as expertise from our Embedded Applications Services group.

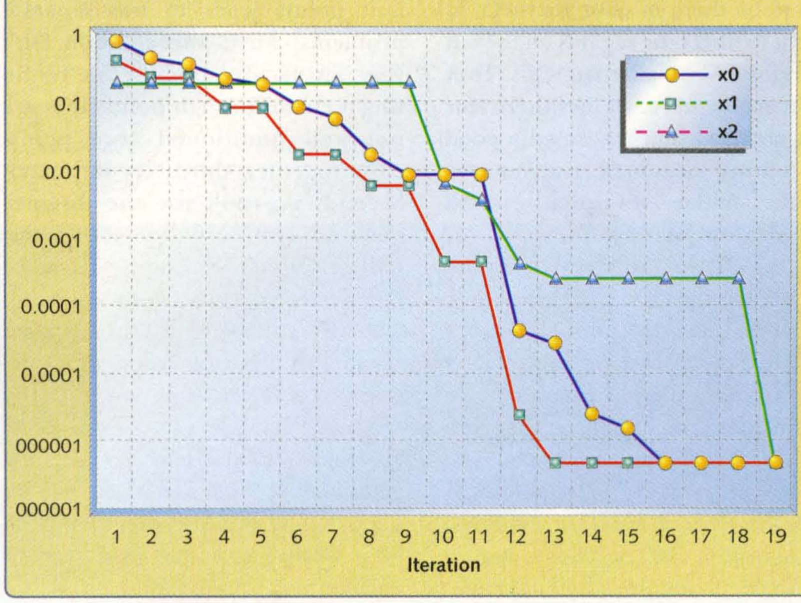
With our installed base of over 100,000 licensed users, world-class support and worldwide presence, you'll know you made the safe choice for your company. For more information, visit **www.tasking.com/tools.**

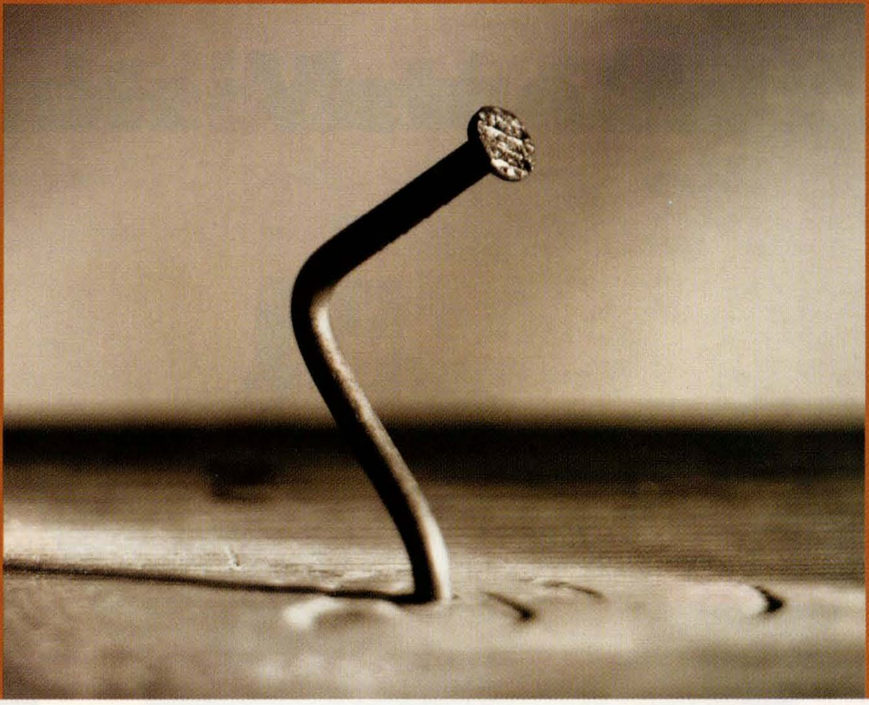
TASKING
The Embedded Communications Company

LISTING 1 Functions for parabolic fit

```
// estimate a minimum via parabolic fit
double para_fit(double x0, double x1, double x2,
                double y0, double y1, double y2){
  double m0 = (y1 - y0)/(x1 - x0);
  double m1 = (y2 - y1)/(x2 - x1);
  double c  = (m1 - m0)/(x2 - x0);
  return (x0 + x1 - m0/c)/2.0;
}

// shrink an interval using parabolic fit
void para_shrink(double &x0, double &x1, double &x2,
                 double &y0, double &y1, double &y2){
  double x = para_fit(x0, x1, x2, y0, y1, y2);
  double y = f(x);
  if(x > x1){
    swap(x0, x2);
    swap(y0, y2);
  }

  if((y > y1) && (x2 > x0)){
    x0 = x;
    y0 = y;
  }
  else{
    x2 = x1;
    y2 = y1;
    x1 = x;
    y1 = y;
  }
  swap(x0, x2);
  swap(y0, y2);
}
```

FIGURE 1 Alternating methods





To make no mistake.

That is what you seek.
To be always right. Ever accurate. To not look like a fool.
But that's not always easy. Not when you're exploring new ground.
When you will only find out what you don't know the hard way.
By trial and error.
We can help.

With test equipment that is accurate, easy and intuitive to use.
That can not only speed through a test, but through those wonderful folks in management.
And with the help and advice of our engineers who, like you, want to do what's right.
Make no mistake.

www.agilent.com/find/engineer 1-800-452-4844,* Ext. 6967

Agilent Technologies
Innovating the HP Way

MontaVista...

Embedded Linux.®

- Embedded Linux that is 100% pure, royalty-free, open source.
- Embedded Linux with the broadest hardware support.
- Embedded Linux for design, development and deployment.
- Embedded Linux leadership measured by what counts...customers.

MONTAVISTA™
SOFTWARE

490 Potrero Avenue
Sunnyvale, CA 94085
tel: 408-328-9200
fax: 408-328-3875
www.mvista.com

The Embedded Linux® Experts.

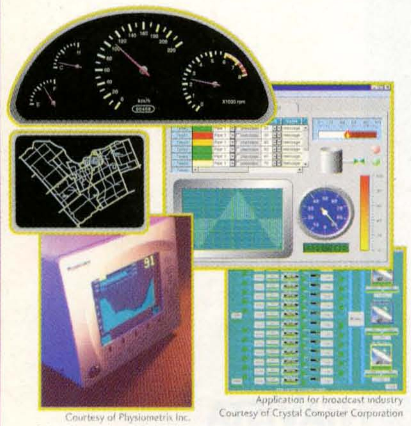
visual solutions *for the* **real world**

The **TILCON Real-Time Developer (TRTD)** is the ideal solution for custom GUI/MMI development in a variety of industries, such as: embedded device interfaces, medical displays, SCADA systems, virtual instruments and navigation systems.

TRTD will drastically cut development time. Innovative architecture provides a clean separation between application code and windowing systems. With its rich set of objects and ability to test interfaces without writing code, **TRTD** quickly moves your project from concept, to prototype, to product.

The **TRTD** API provides dynamic creation and editing of all objects. The API is identical across all platforms, including:

- NT Embedded
- Windows 2000/NT/95/98
- Windows CE
- RTX (VenturCom Inc.)
- LynuxWorks
- Linux: Red Hat, SuSE, Lineo
- QNX® Photon microGUI®
- QNX Neutrino®
- QNX X Free86
- new platforms coming soon


Courtesy of Physiometrix Inc.
Patient State Analyzer PSA 4000™
FDA Approved

Application for broadcast industry
Courtesy of Crystal Computer Corporation

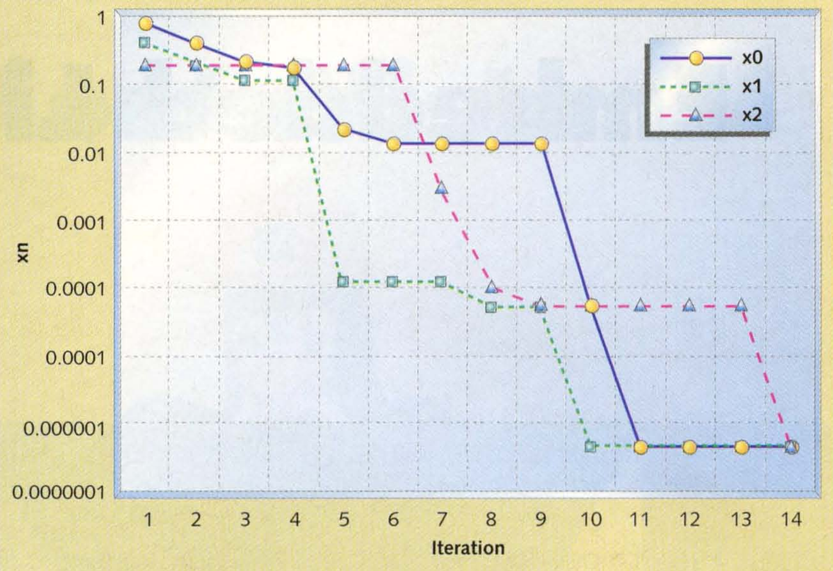
www.tilcon.com

TILCON

tel: 800-665-5928 (toll free in North America)
or 613-226-3917 | fax: 613-226-3631
e-mail: infonews@tilcon.com

Ref: #801

FIGURE 2 2 Parabolic, 1 bisection



LISTING 2 Balanced version of golden section

```
void gold_shrink(double &x0, double &x1, double &x2,
                double &y0, double &y1, double &y2){
  double x = gold(x0, x1);
  double y = f(x);
  if(y > y1){
    x0 = x;
    y0 = y;
  }
  else{
    x2 = x1;
    y2 = y1;
    x1 = x;
    y1 = y;
  }
  x = gold(x2, x1);
  y = f(x);
  if(y > y1){
    x2 = x;
    y2 = y;
  }
  else{
    x0 = x1;
    y0 = y1;
    x1 = x;
    y1 = y;
  }
}
```

FlashFlex51

Flash Microcontrollers

How Hard Is It To Do SuperFlash®/8051 Integration?

Ever Tried Teaching a Gorilla to Play "The Flight of the Bumblebee"?

Let's face it, to put *any* kind of nonvolatile memory together with an industry-standard microcontroller, you've got to know the score. That's why we know you'll appreciate FlashFlex51,™ our family of fully 8051-compatible microcontrollers—smoothly integrated with up to 36 Kbytes of high-reliability SuperFlash® memory.

Put them together and you've got simpler designs, shorter development cycles, smaller parts counts and significant cost-of-ownership savings. It's enough to make you go ape.

Part Number	Memory FLASH/RAM	Speed/Voltage	Interrupt Wake-up	Packages	Vol Ship
SST89C54	20KB/256B	33MHz @ 5.0V, 12MHz @ 3.0V	Yes	PDIP-40, PLCC-44, TQFP-44	Now
SST89C58	36KB/256B	33MHz @ 5.0V, 12MHz @ 3.0V	Yes	PDIP-40, PLCC-44, TQFP-44	Now

FlashFlex51 offers a killer solution for your embedded controller designs, especially for remote applications such as vending machines, set-top boxes and POS systems. Its unique dual memory bank architecture lets you alter flash contents while the 8051 keeps tootling right along—a feature we call In-Application Programming.™ So your in-field products can be upgraded in real time, without being shut down. And our security locks make sure your flash contents aren't accidentally overwritten or electronically swiped.

FlashFlex51's backup ensemble comes complete with SST's outstanding customer service, plus a whole suite of development tools. For details, visit our website. For an evaluation kit, contact your SST distributor or representative today.

SST®

SuperFlash. The Superior Flash Memory.

At the moment, the big question is: does this approach fix the parabolic fit's tendency to hang onto outlier points? Well, it might have, if I'd done it properly. In fact, though, it didn't, because I screwed up.

LISTING 3 Modified version of para_shrink

```
int para_shrink(double &x0, double &x1, double &x2,
                double &y0, double &y1, double &y2){
  double too_close;

  #define MAX_PARA 4
  static int count = MAX_PARA;
  if( count == 0){
    count = MAX_PARA;
    return FALSE;
  }
  double x = para_fit(x0, x1, x2, y0, y1, y2);
  too_close = (x2 - x0)/20;
  if(max(x-x0, x2-x) < too_close){
    count = MAX_PARA;
    return FALSE;
  }
  double y = f(x);
  if(x < x1){
    if(y > y1){
      x0 = x;
      y0 = y;
    }
    else{
      x2 = x1;
      y2 = y1;
      x1 = x;
      y1 = y;
    }
  }
  else{
    if(y > y1){
      x2 = x;
      y2 = y;
    }
    else{
      x0 = x1;
      y0 = y1;
      x1 = x;
      y1 = y;
    }
  }
  return TRUE;
}
```

let's not worry about that too much for now. We'll be thinking harder about optimizing after we better understand the behavior.

At the moment, the big question is: does this approach fix the parabolic fit's tendency to hang onto outlier points? Well, it might have, if I'd done it properly. In fact, though, it didn't, because I screwed up. The parabolic fit ended up giving me points that failed to satisfy our fundamental requirement, that the middle point is always lowest.

The problem is still in that pesky statement, which was:

```
if(y > y1)
```

last month, and in Listing 1:

```
if((y > y1) && (x2 > x0))
```

Turns out, neither test is correct. What I was trying to do was to swap P_0 and P_2 if x turned out to be between x_1 and x_2. That's because x is guaranteed to be between x_0 and x_1 for the golden section, and I wanted it to look the same way after a parabolic fit.

However, I forgot that, because of all the swapping, sometimes x_0 is larger than x_2, and sometimes not. The correct test is:

```
if((y > y1)*(x2 > x0) > 0)
```

(Recognizing that this can be a risky test if x_0 and x_2 are drawing closer together, as we hope they do).

So I can fix the test. Do we now get performance where the endpoints always move inward? No, we don't. Figure 1 shows the case where I alternate between golden and parabolic methods. In Figure 2, I've set the frequency to allow two parabolic fits, then a forced golden section. It doesn't seem to matter; the problem of endpoints stuck at the same values persists.

To swap or not to swap

We don't have to look very far to see the problem: if you examine the

FlashFlex51

Flash Microcontrollers

FOR YOUR NEXT EMBEDDED BASED DESIGN, GO DIRECTLY TO FLASHFLEX51.

SO YOUR HEAD DOESN'T GET EMBEDDED IN LOW-LEVEL DETAILS.

www.SuperFlash.com

You're doing an embedded design. Up front, you're faced with the formidable task of integrating a microcontroller with flash, mask ROM, EPROM, OTP devices or other memory types. Maybe a little time spent imitating the ostrich would give you temporary relief. Probably not. May we suggest you avoid the problem completely, and jump directly to square 5—the one that says: 'Use FlashFlex51 for a successful design.'

Because FlashFlex51 is a real oasis in a desert of hard-to-integrate solutions. One that puts a fully 8051-compatible microcontroller together with up to 36 Kbytes of SST's high-reliability SuperFlash® memory. So when you tell your colleagues you just simplified your design, shortened your development schedule, saved on parts, and lowered your cost of ownership—they can go stick their heads in the sand. Or wherever.

Part Number	Memory FLASH/RAM	Speed/Voltage	Interrupt Wake-up	Packages	Vol Ship
SST89C54	20KB/256B	33MHz @ 5.0V, 12MHz @ 3.0V	Yes	PDIP-40, PLCC-44, TQFP-44	Now
SST89C58	36KB/256B	33MHz @ 5.0V, 12MHz @ 3.0V	Yes	PDIP-40, PLCC-44, TQFP-44	Now

Another watershed event in flash/MCU capabilities is our unique dual-bank architecture. It enables you to alter flash contents in real time, while your system keeps right on running—something we call In-Application Programming.™ And with our built-in security locks, no one can accidentally (or purposely) overwrite or swipe your flash contents.

And if your business seems to be poking along at desert-crossing speeds—due to long mask ROM lead times and wasted parts—try comparing FlashFlex51. It's backed by a full suite of development tools, plus our legion of tech support professionals. For details, visit our website. For an evaluation kit, contact your SST distributor or representative today.

SST®

SUPERFLASH.
THE SUPERIOR FLASH MEMORY.

Microsoft Windows Powered

The Microsoft Windows Embedded Family.

The Microsoft® Windows® Embedded Family of operating systems offers a comprehensive software platform for bringing the next generation of intelligent and connected devices to market faster.

Choose from the widest range of embedded OS technologies that integrate seamlessly with Windows and the Web. Tap into a worldwide pool of more than 5 million Win32®-trained developers familiar with award-winning development tools, like Visual Studio®, to help develop your embedded applications.

With three OS's to choose from, there's one that's right for your next project.

*While supplies last. Online orders have a fee of $12.95 which includes shipping and handling.

It's not for everyone.

Only those interested in faster time to market.

The Microsoft Windows Embedded Family consists of:

Windows CE 3.0

The modular, real-time OS, perfect for small footprint 32-bit devices. Supports multiple processor architectures? Yes. Built-in network connectivity? Yes. Instant-On? Yes. Build your devices quickly with Windows CE Platform Builder 3.0.

Windows NT® Embedded 4.0

The full-featured embedded OS designed for systems based on PC architectures. Supports thousands of device drivers and Windows-based applications. Enables the richest applications and services for your embedded solution.

Windows 2000 with the Server Appliance Kit

Creating dedicated server appliances? Take advantage of Windows 2000 performance, reliability, and security. Use the Server Appliance Kit with predefined templates and headless server support to build your dedicated servers.

Microsoft® Windows CE

Microsoft® Windows NT Embedded 4.0

Microsoft® Windows 2000

Get time on your side.
Come visit us at www.microsoft.com/embedded/eval and order your Windows Embedded Evaluation Kit*.

Where do you want to go today?® Microsoft®

LISTING 4 Counting each endpoint

```
int para_shrink(double &x0, double &x1, double &x2,
                double &y0, double &y1, double &y2){
  double too_close;
  #define MAX_PARA 2
  static int count0 = 0;
  static int count2 = 0;

  if(count0 == MAX_PARA){
    count0 = 0;
    return FALSE;
  }
  if(count2 == MAX_PARA){
    count2 = 0;
    return FALSE;
  }
  double x = para_fit(x0, x1, x2, y0, y1, y2);
  too_close = (x2 - x0)/20;
  if(max(x-x0, x2-x) < too_close){
    return FALSE;
  }
  ++count0;
  ++count2;
  double y = f(x);
  if(x < x1){
    if(y > y1){
      x0 = x;
      y0 = y;
      count0 = 0;
    }
    else{
      x2 = x1;
      y2 = y1;
      x1 = x;
      y1 = y;
      count2 = 0;
    }
   }
   else{
     if(y > y1){
       x2 = x;
       y2 = y;
       count2 = 0;
     }
     else{
       x0 = x1;
       y0 = y1;
       x1 = x;
       y1 = y;
       count0 = 0;
     }
   }
   return TRUE;
```

code for `gold_shrink()`, you'll see that it always does just one bisection, and that one is always between x_0 and x_1. In practice, the whole point of bisections is to pull both endpoints in, but someone got tricky and hoped to save a little code by only doing one bisection per pass. That someone was me. My only excuse is that I had a little encouragement from both Brent and the *Numerical Recipes* authors, who used the same approach.

That approach limits the number of function evaluations per pass through the outer loop to one, which may make it nice for counting function evaluations, but has little to offer otherwise. We save duplicating a few lines of code, which makes the routine that much shorter and more compact. On the down side, we have all those stupid swaps, which not only confuse the heck out of me (as my errors in the swap test fully attest), but eat CPU cycles as well. A side effect is that we can't be sure whether x is increasing to the left or the right.

In retrospect, I believe that someone (again, me; who else?) got entirely too tricky for his own good. Given a choice between tight and tricky, and large but simple, I'll take simple every time. So let's begin the process of simplification by writing `gold_shrink()` the way it should have been in the first place, and have it reduce both sides of the interval each call. We'll require, and assume, that:

$$x_2 > x_0$$

throughout the process (except when they're equal, of course, in which case we're done). The new version is shown in Listing 2.

Function `para_shrink()` can now be simplified a bit, since we know the order of the x's. We can also eliminate all those silly swaps. The new code for this function is shown in Listing 3. The corresponding performance is shown in Figure 3.

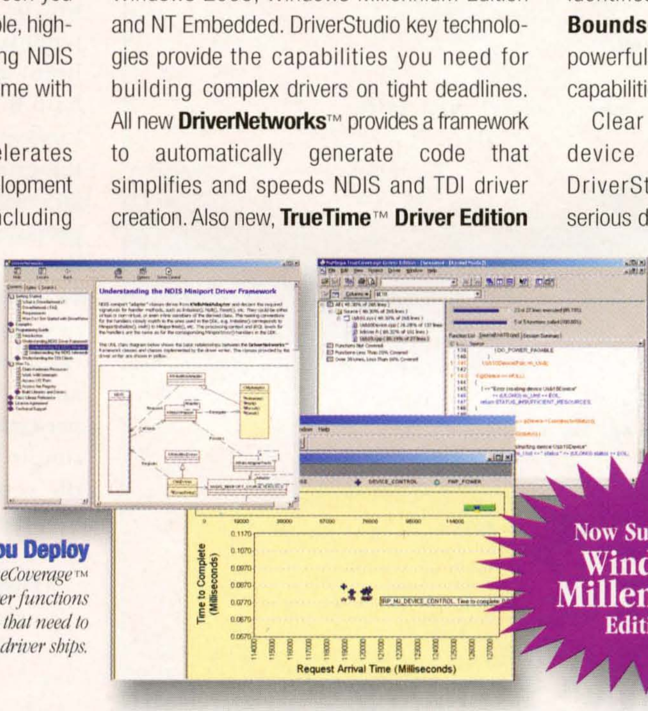
Are Device Driver Development Hurdles Getting in Your Way?

Leap the hurdles of complex driver development with NuMega DriverStudio!

CLEAR THE HURDLES that stand between you and reliable device drivers! Deliver reliable, high-performance device drivers—including NDIS and TDI drivers—in a fraction of the time with **NuMega® DriverStudio™**.

DriverStudio improves and accelerates critical phases of the device driver development lifecycle for Windows® platforms including Windows 2000, Windows Millennium Edition and NT Embedded. DriverStudio key technologies provide the capabilities you need for building complex drivers on tight deadlines. All new **DriverNetworks™** provides a framework to automatically generate code that simplifies and speeds NDIS and TDI driver creation. Also new, **TrueTime™ Driver Edition** identifies performance bottlenecks. And **BoundsChecker® Driver Edition** offers powerful driver analysis and error detection capabilities.

Clear the hurdles fast! Deliver reliable device drivers faster with NuMega DriverStudio—everything you need for serious device driver development.


Understanding the NDIS Miniport Driver Framework
Time to Complete (Milliseconds)
Request Arrival Time (Milliseconds)
DEVICE_CONTROL
PNP_POWER

Accelerate Network Driver Development

DriverNetworks provides a method to define essential characteristics of your NDIS network driver and generates a driver skeleton to accelerate your development process.

See the Performance

TrueTime Driver Edition collects and plots performance data, including IRP completion times, DPC latencies, and interrupt timing, enabling you to visualize your driver's performance throughout its execution.

Test Before You Deploy

Using customizable colors, TrueCoverage™ Driver Edition identifies driver functions and individual lines of code that need to be tested before the driver ships.

Now Supports Windows Millennium Edition!

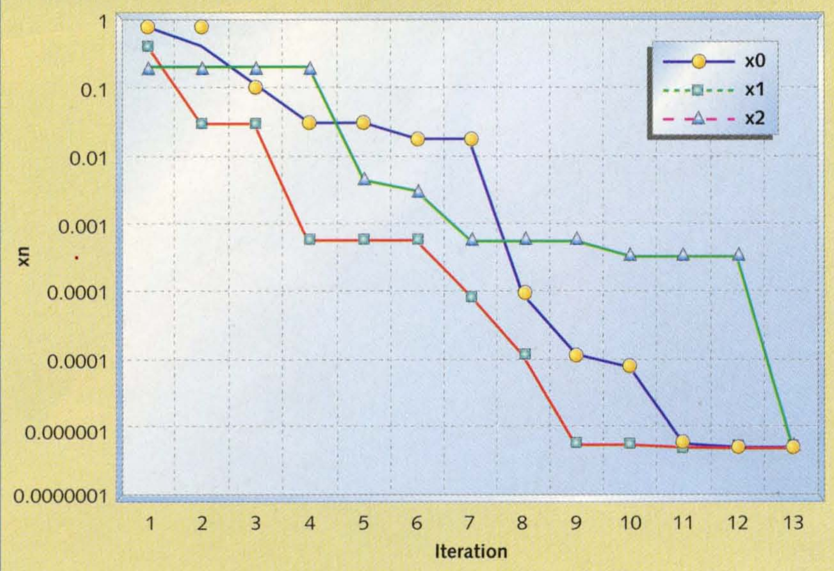
Accelerate Device Driver Development with NuMega DriverStudio!

Call today to request the *"Windows 2000 NDIS Driver Development and Debugging"* technical paper or download the paper now by visiting **www.numega.com/ndisdriver.shtml**

1-800-4-NUMEGA
www.compuware.com
30-Day Money Back Guarantee

COMPUWARE
People and software for business applications™

FIGURE 3 Parabolic bisection



LISTING 5 A general-purpose minimizer

```
double para_search(double (*f)(double), double &x0, double &x2, double Eps){
  #define MAX_ITER 30
  double eps = Eps * (abs(x2-x0));

  // initial last values make sure we try at least twice
  double last1 = 1e20;
  double last2 = -1e20;

  double x1 = gold(x0, x2);
  double y0 = f(x0);
  double y1 = f(x1);
  double y2 = f(x2);
  for(i=0; i< MAX_ITER; i++){
    if(!para_shrink(x0, x1, x2, y0, y1, y2))
         gold_shrink(x0, x1, x2, y0, y1, y2);
    if((x2-x0) < eps)
      break;
    if(max(abs(x1-last1), abs(x1-last2)) < eps)
      break;
    last2 = last1;
    last1 = x1;
  }
  if(i == MAX_ITER)
  cout << "Para_search: No convergence after " << MAX_ITER << " trials." <<
 endl;                                       return x1;
}
```

Now that's more like it! True, we still have variables that seem to stick at one value. But look how dramatically the others are changing as we go. After only eight passes, all three errors are less than 0.001. Compare that to Figure 1, where it takes 12 passes to accomplish the same result, or Figure 2, which takes 10. Admittedly, we're now evaluating the function twice for each bisection pass, but even so, the difference is impressive. I don't mind a value getting stuck at, say, 1.0, if the other two are rapidly squeezing up next to it. That's the best we can really hope for in such a method.

Preventing bad estimates

One thing we still must check on is the suitability of the estimates coming out of the parabolic fit. If, for example, it gave a value for x_1 that's, say, almost equal to x_0, we can expect that the next parabolic fit will be poor. I've flagged this as a potential problem in the past, and Brent worried about it too. However, both Brent's original Algol code and Press et al.'s C translation only reject the estimate if it's very close (within an epsilon value on the order of the precision in the solution) of one of the bracketing points. I'm going to be much more aggressive on this one, and reject it if it's within 5% of the total current interval between x_0 and x_2. The structure of para_shrink() allows us to reject, simply by returning a false value for the return Boolean. The code fragment is:

```
if(max(x-x0, x2-x) < too_close){
  count = MAX_PARA;
  return FALSE;
}
```

I'd also like to change the criterion for forcing a bisection step. Currently, we just count off parabolic steps, and then force a bisection whether the method needs it or not.

inferno®

Before You Invest in Embedded Linux look at the Facts!

	Inferno	Embedded Linux
Do you know the Designers	Yes, created by the inventors of the UNIX operating system and the C programming language	No
Designed for Embedded Systems	Yes	Server operating system modified for embedded systems
Get all the Source Code	Yes	Maybe
Supported Architectures	MIPS, x86, StrongARM and PowerPC	x86 and ?
Emulated Environments	Windows 95/98, Windows 2000/NT, Linux/x86, FreeBSD and others	None
Write Once Run Anywhere	Yes	No
Unicode Character Set Support	Yes	Maybe
Royalty Free	Yes	Maybe
Protect your Intellectual Property	Yes	Maybe
Available Today	Yes	Maybe

The Network Operating System for distributed application development on network devices

vita nuova®

Tel: 1-866-DANTEOS

www.vitanuova.com

At this point, we almost have a serviceable function, in that it takes a mixture of parabolic and golden section steps, and it chooses between the two based on reasonably intelligent, rather than pre-programmed, rules.

LISTING 6 A test driver

```
#include <iostream.h>
#include <stdlib.h>
#include <math.h>
#include <constant.h>
#include <jmath.h>

/* NOTE:  Files constant.* and jmath.* define pi, the Golden ratio,
* and also define abs as a macro rather than a function.  If you
* don't do this, better change calls to abs to fabs.
*/

double f(double x){
    return cos(2*pi*x*x*x);
}

void main(void){
  double eps = 1.0e-7;
  double x0 = 0;
  double x2 = 1;
  double x = para_search(f, x0, x2, eps);
  cout << x << endl;
}
```

For the test case, it has always been needed in the sense that one of the boundary points seems stuck. But this may not always be true. When the boundary points are not stuck, it's foolish to abandon a method that's working. A better method would be to keep a separate count for each boundary point, and kick out when one appears to be stuck. Listing 4 shows function `para_shrink()` after this change.

Knowing when to quit

I think this header is particularly apt, both for this month's session, and also for the study of minimizing a function. I'm ready to quit, aren't you?

At this point, we almost have a serviceable function, in that it takes a mixture of parabolic and golden section steps, and it chooses between the two based on reasonably intelligent, rather than pre-programmed, rules. The one thing still lacking to make the function production-quality is a good stopping criterion. Recall that, for bisection-only case, the criterion is simple: we stop when the interval between x_0 and x_2 is reduced sufficiently far. Thanks to the forced changes in both boundaries, we could still use this criterion, but in fact, the parabolic fits seem to be homing in considerably faster than that last boundary point gets pulled in. So we'd be taking a few too many steps.

I have what I think are some super ideas for a robust halting criterion, but they would require too many changes to our code to include them this month. Brent used the change, not in two successive parabolic estimates, but in the estimates two cycles apart. Press et al. suggest only that "the reason ... seems essentially heuristic." I'll go Brent one step better, and require that we have three parabolic estimates within an epsilon error. The code is collected into a general-purpose function in Listing 5. A sample driver program, complete with test function, is shown in Listing 6.

For your interest, the method converges to 1^{-7} accuracy in 17 iterations: five bisections and 12 parabolic fits. It uses a total of 25 function evaluations. I know, I know, that's more than Brent's method. That is no doubt because each golden section makes two evaluations, not just one. At this point, it's a price I'm prepared to pay to get rock-solid robustness.

I still believe we can improve on the algorithm some more. For example, I want to try my ideas for a better halting criterion. However, I won't be doing that next month. I want to give both you and I some time to wring out the algorithm. Have at it, try it on as many functions as you can, and see if you can break it. Meantime, I think we could all use a rest from Brent's method and its cousins, so next month I'll be starting a new topic.

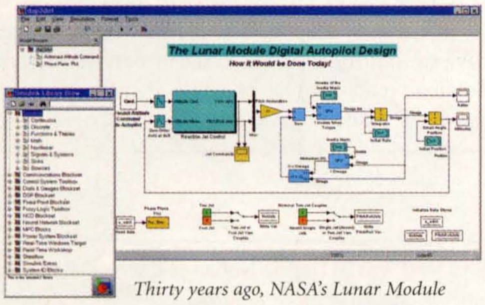
See you then. **esp**

Jack W. Crenshaw 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 is the author of Math Toolkit for Real-time Programming *(CMP Books, 2000). He holds a PhD in physics from Auburn University. Crenshaw enjoys contact and can be reached via e-mail at jcrens@earthlink.net.*

NEW
SIMULINK 3
AVAILABLE
NOW

What will you think of next?

New Simulink 3 is the integrated design and simulation solution that gives you the power to explore, refine and realize your control, communications and DSP designs in a fraction of the time needed for traditional methods. Powerful extensions let you create event-driven simulations and generate code automatically.


The Lunar Module Digital Autopilot Design
How it Would be Done Today!

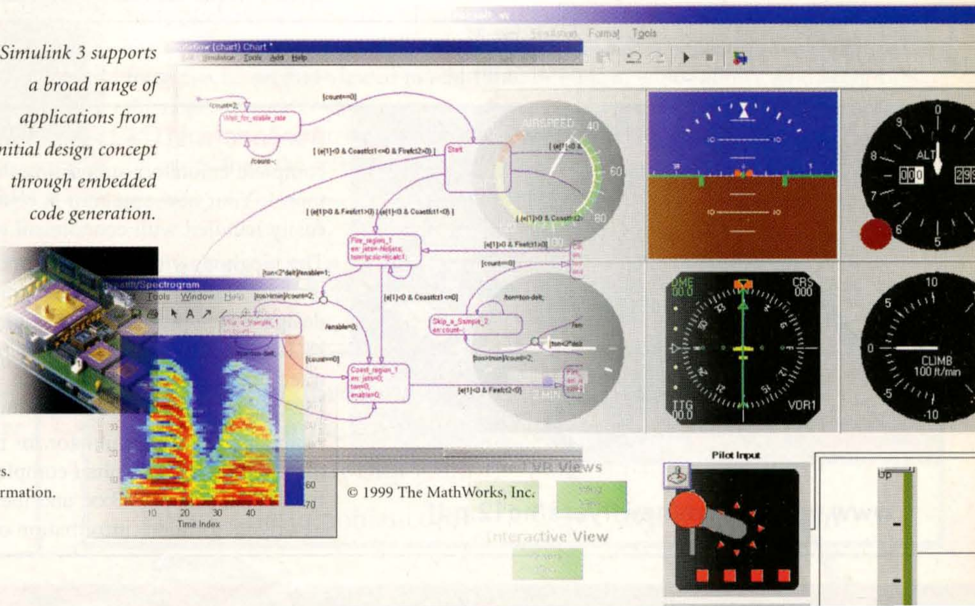
Thirty years ago, NASA's Lunar Module Autopilot took thousands of hours to design. Today, it could be done by one engineer in about a week using Simulink 3.

Act now. Get data sheets, application examples, user stories and product demos.

See how Simulink 3 can help you build superior products in less time. For more information, or to buy now, visit www.mathworks.com/espb.

SIMULINK®

Simulink 3 supports a broad range of applications from initial design concept through embedded code generation.



The MATH WORKS Inc.

Visit www.mathworks.com/espb
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.

nOHau

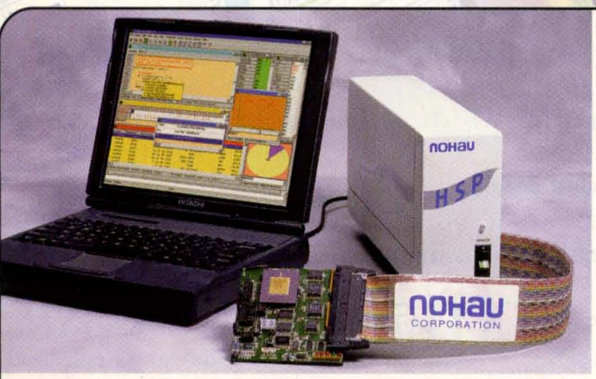
In-Circuit Emulators

www.nohau.com/newflyers/st10.pdf & c166.pdf

Infineon C166 and ST Microelectronics ST10: Nohau has the broadest and fastest line of C166 and ST10 emulators in the world. Up to 80 MHz with no cycle stealing from the bondout CPU. You are assured that your programs always run full speed as intended with no CPU stalling. The Shadow RAM displays writes in real-time at all processor speeds.

These emulators are hand held portable models allowing operation anywhere you and your laptop can go: or share in your lab. Connection to your LPTx: port or via an ISA card for versatility. Emulation memory can be mapped down to a granularity of 2 bytes allowing wrapping around any peripheral. This easily accommodates your targets in various stages of development.

Nohau provides personality modules to adapt to new derivatives as they are introduced so you never have to wait for support. You can start development using the emulator in stand-alone mode before the target chips are available. These emulators are designed and Made in the USA.

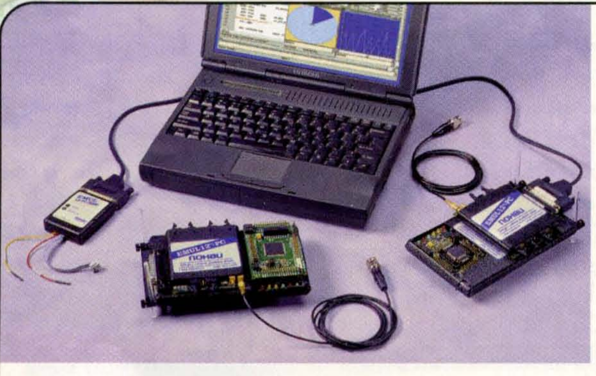


www.nohau.com/newflyers/8051.pdf, c500.pdf, & dallas.pdf, 251.pdf & xa.pdf

8051 and Infineon C500 Families: Nohau is the world's largest 8051 emulator manufacturer. Nohau has more bondout experience than anyone else. This assures you of trouble free single-chip emulation with processors from Intel, Dallas, Philips, Infineon, Atmel and others. Nohau supports nearly all 8051 derivatives. Our website at www.nohau.com/newflyers/support.pdf gives you the complete list of all devices supported.

Nohau offers a wide range of options to suit any need and budget you have including rentals. Nohau technical support is legendary and with reps everywhere, you always get fast answers to get you developing sooner.

Nohau supports the Philips XA and the Intel and Temic 251 family if you use these upgrade paths from the 8051. If you use the Infineon C500 and C166 families together in your projects, both emulators use the same user interface. This eases moving from one family to the other hence increasing productivity. Nohau works seamlessly with all popular compiler and RTOS developers.



www.nohau.com/newflyers/hc12.pdf

Motorola HC12: Finish your developments on time with the most complete emulators supporting the B32, BC32, D60, DA128, DG128 and more! Your new emulator is obsolete-proof since future derivatives will be easily handled with economical personality modules.

The pipeline is fully decoded preventing false triggering on fetched, yet unexecuted instructions. The ports are reconstructed using Nohau's own FPGA design which provides true single chip operations. No chip resources are used by the emulator and full speed, full feature operation is assured. The optional trace board allows triggering on user-specified events to increase productivity. FLASH and EEPROM programming is supported for appropriate chips.

Nohau has a BDM emulator for the above parts plus the HC12A4, HC16 and the 683xx family. Nohau completes its Motorola support with full emulators for the HC16 and 683xx, and the HC11. See the website www.nohau.com/newflyers for more information or call your local Nohau representative.

Cut Your Development Time!

with Nohau in-circuit emulators.

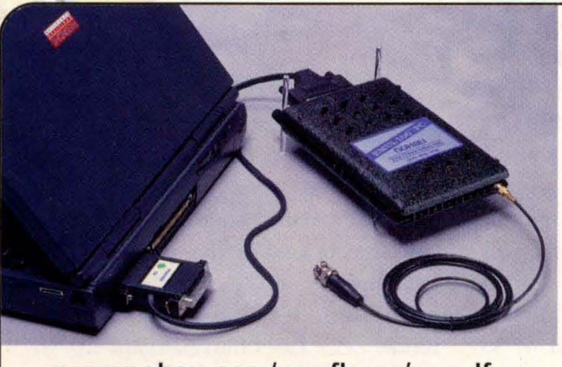
www.nohau.com/newflyers/m16c.pdf

Mitsubishi M16C: Nohau suppoorts this family with a new emulator which has the ability to program the FLASH in your target CPU. The photo illustrates this setup. Like most Nohau products, this emulator is handheld.

The pipeline is fully decoded and the optional trace board displays cycles filtered by the triggers. The entire 1M space is emulated with a granularity of 2 bytes. Shadow RAM displays all writes in real-time in several numerical or graphical formats. The emulator operates in single-chip or expanded mode.

It operates to 16 MHz and with a Vcc of 2.7 to 5.5 volts, it will easily adapt to your target. The triggers can be set and the trace viewed without stealing cycles from the CPU thanks to the internal housekeeping controller. All Nohau emulators operate in this fashion giving you the truest emulation possible.

To speed up your M16C project, contact Nohau today for the latest information.

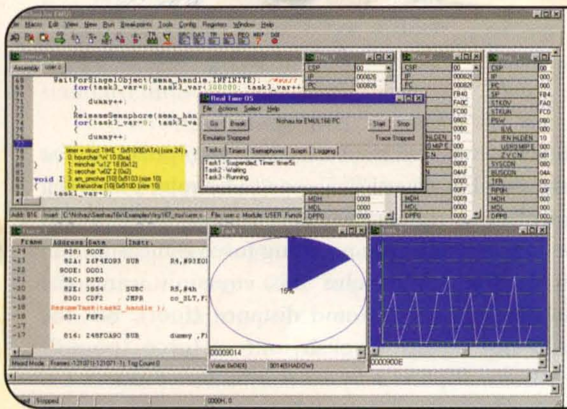


www.nohau.com/newflyers/xa.pdf

Philips XA and others: Nohau supports many other families including the XA, 251, 196, 296, HC11, HC16, 683xx. More are under current development. They all use the same Seehau interface for continuity and the same Nohau quality that assures you of a faster development process. For data sheets on these products, browse www.nohau.com/newflyers. For information on how to use an emulator in your development, call Nohau or your local rep.

Check other areas of the Nohau website for application notes, useful links to our partner's websites, new product announcements and software upgrades.

Nohau has shipped over 18,000 emulators since 1986 and is the world's largest supplier of 8 and 16 bit emulators. Nohau reps and offices are everywhere offering the best in customer service. Nohau is headquartered in Silicon Valley USA and has major offices in Sweden, Germany, China and Great Britain.



Seehau: Nohau's new user interface: Seehau provides you with a sophisticated yet friendly debugging environment. For a complete description see the website www.nohau.com/newflyers/seehau.pdf. Call Nohau for the latest demo CD or call your Nohau rep for a personal demonstration at your site.

Seehau can operate over TCP/IP for remote debugging, has integral RTOS support, multiple configurable windows, a MACRO scripting language including an integrated Visual Basic (Sax Basic), and much more.

Data writes can be displayed in many numerical and graphical formats and the trace can be set and viewed in real-time. A built-in editor allows modifying code, then calling your compiler and then displaying any error messages - all within one environment. Seehau runs under Windows NT/95/98. Nohau offers no cost upgrades from the web and they can be retrofitted to existing Nohau emulators.

Your Action List:

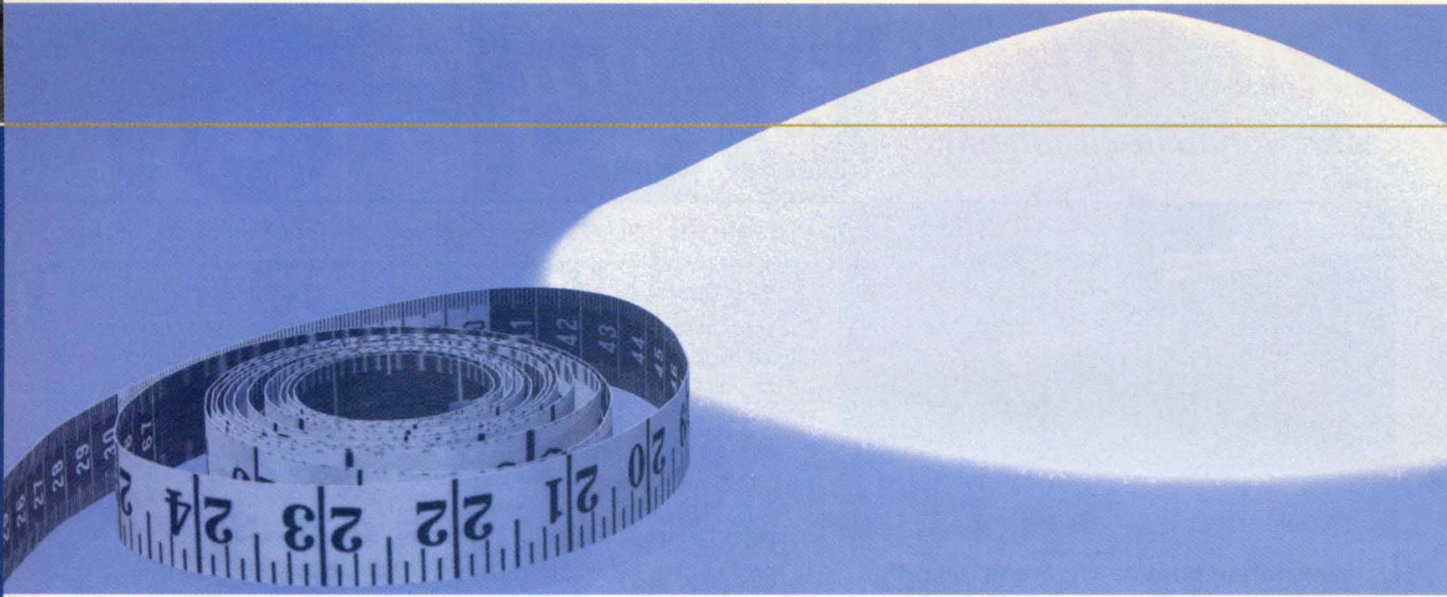
- ☐ Visit our website for app notes and "The Designer's Guide to In-Circuit Emulation"
- ☐ Call your local Nohau rep to see how they can help you complete your projects faster.
- ☐ Nohau reps offer complete development solutions for easy integration of projects.
- ☐ Call Nohau for a free evaluation period.

www.nohau.com

Nohau Corporation
51 East Campbell Avenue
Campbell, California 95008
Email: sales@nohau.com

Tel: (888) 886-6428
Tel: (408) 866-1820
Fax: (408) 378-7869
Web: www.nohau.com

JOHN FUSCO



Measure Twice, Cut Once

Carefully named variables help reduce confusion. Follow this naming convention to track units and avoid mismeasurement.

In the fall of 1999, NASA experienced the failure of its Mars Climate Orbiter. The failure was attributed to miscalculations in the thrust used to correct anomalies in the orbiter's trajectory on its way to Mars. These miscalculations, it was later discovered, were due to one subcontractor using English units instead of metric units. NASA specified the output of a piece of ground software to be in metric units of impulse (Newton-seconds); the software produced output in English units of impulse (pound-seconds).

This article discusses problems associated with the lack of a useful software abstraction for measurement units and presents some simple ideas for dealing with them.

Standard units

The Systéme International d'Unités (SI, for short) standardizes the units that are used in science and engineering. A feature of SI units is that you can solve all the fundamental equations in physics without any extra manipulation. For instance, the familiar equation $F=ma$ can be solved by using the SI unit of mass (kilograms) and acceleration (meters per second per second). The resulting force is measured in Newtons (N). You could solve this equation using the English units of mass (slug) and distance (foot). But the result would not be the English unit of force (pound) unless you also multiplied it by a nonintuitive constant.

SI classifies units into two different types: base units and derived units. A *base unit* is a unit that cannot be decomposed into other units. Length is an example of a base unit, since it cannot be expressed in terms of other units, like time or mass. A *derived unit* can be expressed in terms of multiple base units and may have a proper name. The SI unit of force is a derived unit that has a proper name. The Newton is derived from the SI units of mass, distance, and time. This can be illustrated using Newton's second law:

Mandating SI standard units in embedded software is not practical when the inputs and outputs are anything but standard.

Force = mass * acceleration

Alternately:

Newton = kilograms*meters/seconds/seconds
1 Newton = 1 kg-m/s^2

In addition, SI provides a vocabulary for scaling the standard units by powers of ten. These familiar prefixes are shown in Table 1.

Why not use SI units for everything?

Mandating SI standard units in embedded software is not practical when the inputs and outputs are anything but standard. Such a requirement can directly impact the performance and cost of the system. For instance, a system that measures time in one-microsecond clock ticks would need a data type that can represent 1/1,000,000th of a second in order to keep the data in SI units. Since C does not provide a fixed-point type, the only other choices are to use a floating-point type or an integer in non-standard units.

Cost-sensitive embedded systems avoid floating-point computation whenever possible. Using a CPU without a floating-point coprocessor can cut costs. Floating-point computations on an integer CPU require the use of a floating-point emulation library to take the place of the coprocessor. The emulation library takes up precious memory, which can affect the cost of the system. Floating-point operations are emulated with function calls that may be routed via an exception handler. Operations that would normally compile to a single instruction for an integer are turned into function calls for emulated floating point. Even the fastest integer processor will suffer a significant performance penalty when it is forced to use emulated floating-point operations. When a floating-point coprocessor is available, the space required to store 4-byte floats might be too costly when 2-byte integers will suffice.

For most programmers, the obvious solution to this problem is to use a scaled integer, that is, an integer in non-standard units. For our one-microsecond clock tick, the simple choice is to represent time in units of microseconds. Strictly speaking, this is a standard unit because we are only scaling the time by a power of 10. Scaling SI units by powers of 10 is as simple as slapping on one of the prefixes shown in Table 1 in front of the name.

Unfortunately, scaling by a power of 10 is often no help in embedded software. For example, the output of an analog-to-digital converter (ADC) is typically a binary value—meaning the range of output values is a power of two. The input range, on the other hand, can be constrained in a completely arbitrary fashion. So, in general, the output of an ADC will be $X/2^y$, where X is the input range and y is the number of bits in the output. Even if the actual input (x) is in standard units, your output will be some standard unit divided by a power of two. SI doesn't help much here, since there is no vocabulary for representing standard units scaled by powers of two. That's probably just as well, since it is not very often that your input is an integral multiple of a standard unit either.

Use and abuse of units in software

It's hard to adhere to any convention, let alone SI, in your software, when all your inputs are in non-standard units. Devices like timers, ADCs, and encoders spit out integers in whatever units a hardware designer can dream of. Sometimes these units make sense and sometimes they're just arbitrary, but mostly they're not standard. It's up to the embedded programmer to make some sense out of the chaos.

It would be easy just to blame hardware designers, but there's plenty of blame to go around. Programmers can also play fast and loose with the data and confuse anyone who tries to follow. Many a shortcut has been used to save CPU cycles that make the code read like a college textbook where "the proof is left as an exercise for the reader." While shortcuts are often necessary in embedded software, it is not necessary to confuse everyone who looks at the code.

Some common shortcuts you may have run across include interchanging terms like "frequency" and "period," or "time" and "distance" in your code. This may seem trivial as long as you are familiar with the design, but it can confuse a newcomer. Another example would be hard-coding scale factors in your code to convert from one system of units to another, with no mention as to what is going on.

Sometimes, we hide vital information in a design document that could have easily been included in the code in the form of a comment or an informative name. Programmers will make assumptions without always checking the documentation—that's human nature. Some assumptions may seem obvious to the programmer, but can still be wrong. It may have seemed obvious to the programmers writing the software for the Mars Climate

RUPERT ADLEY

The C and C++ languages do not give you many choices when it comes to representing scalar variables. You basically have two choices: integer or floating point.

TABLE 1 SI standard prefixes

Scale factor	Prefix	Symbol
10^{-9}	nano	n
10^{-6}	micro	µ
10^{-3}	milli	m
10^{-2}	centi	c
10^{-1}	deci	d
10	deka (or deca)	da
10^{2}	hecto	h
10^{3}	kilo	k
10^{6}	mega	M
10^{9}	giga	G

Orbiter that thrust should be expressed in pounds. If they had checked the design document, they would have found otherwise.

The problem of units in software

The C and C++ languages do not give you many choices when it comes to representing scalar variables. You basically have two choices: integer or floating point. The language makes no assumptions about what the data represents, so you get no complaints from the compiler when you do things like:

- Assign units of one type to a variable in different units; for example, "length = time"
- Add or subtract variables that are not in the same units; for example, "length = length + time"
- Multiply or divide variables in different units and store the result in a variable in the wrong units; for example, "frequency = time/cycles"

In a nutshell, what we need to define is a unique scalar type for each unit of measure, such as length or time, that is derived from one of the native scalar types (int, double, float). The goal is to define some simple rules and let the compiler enforce them. Some languages, like Ada, have this ability built-in. Once you define your own scalar type, an Ada compiler won't allow you to mix up the math unless you explicitly instruct it to do so. Unfortunately, this is not so trivial in C and C++.

The `typedef` keyword, which provides the simplest way to define your own scalar type in C and C++, creates nothing more than an alias. The compiler doesn't care if you mix up an expression using a "meter_type" and a "second_type," as long as they are both scalars. In most cases you won't get a warning, much less an error. To make matters worse, program checkers like lint don't care about user-defined scalar types either! You may as well just use the preprocessor.

The next logical step is to define a class, but that opens up a new can of worms. In C++ a class cannot inherit from a scalar. For instance, the following C++ statement will not compile:

```
class METER : double  {};
  // Does not compile
```

If you want to create your own scalar class, you must settle for something like the following:

```
class METER { double value; };
```

This is unfortunate because now the compiler knows absolutely nothing about scalar operations for this class. You must write code for every scalar operation, including several you may not have thought of. For instance, in addition to writing operator methods for "+", "-", and so on, you must supply the code to perform all type conversions, which are necessary to evaluate a simple expression like the following:

```
velocity = (METERS_PER_SEC) meters
  / (METERS_PER_SEC) seconds;
```

You have to define a method for the `METERS_PER_SEC` typecast in both the `METERS` type and the `SECONDS` type. You also have to provide various constructors and assignment operators before this class will work like a plain scalar. A complete example would take several pages and can be found in the first reference at the end of this article. I'm sure you can find one in your favorite C++ textbook. You will also have to revise these classes regularly to provide access to new derived units. This is not a low maintenance solution.

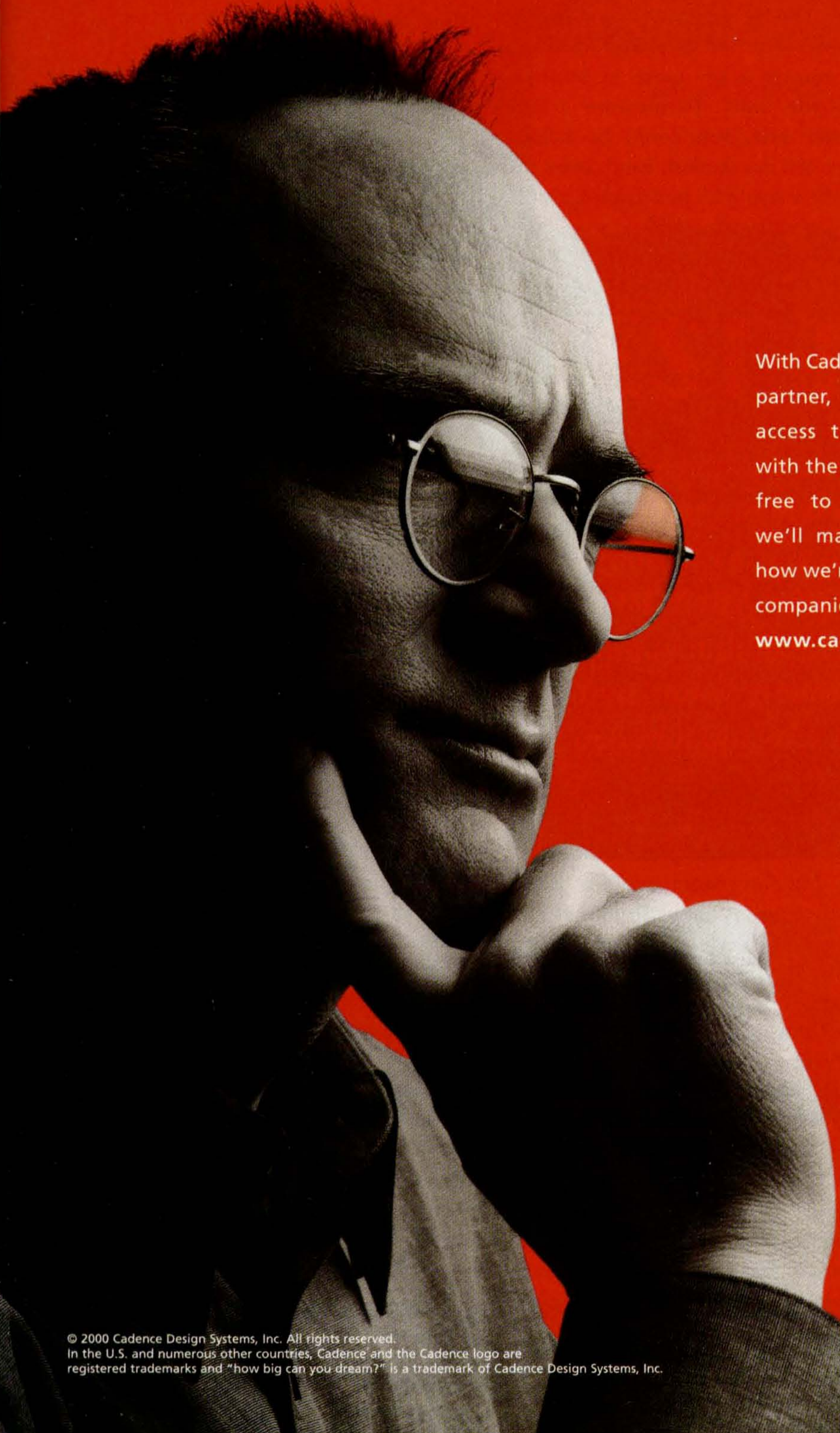
All we wanted was for the compiler to enforce better type checking of scalars. But to do that we are forced to write lots of extra code that adds no value to the application. As if that weren't bad enough, using a scalar class such as this will produce inefficient code. It forces the compiler to insert a call to a constructor or method where it would normally insert a single machine instruction. Even with inline code and aggressive optimization, I doubt that this code could ever be as efficient as code written with plain scalars.

For me, the ultimate question regarding a solution like this is "does it add value to the code?" I have three simple criteria I use for deciding if a solution adds value:

- Does it reduce the amount of code that must be written? Fewer lines of code means fewer opportunities for bugs
- Is it low maintenance? Code that has to be constantly modified to accommodate every application does not help. The act of modifying the code introduces more opportunities for bugs
- Is it easy to understand? If you can live with high maintenance code, it had better be easy to understand. The more difficult it is to understand the easier it is to make a mistake when "maintaining" it. A kludge solution that works well for only one application is not much of a solution

It takes time to put together an expert design team.

Sometimes as long as 24 hours.



Cadence

With Cadence® as your Design Services partner, you'll have quick and easy access to hundreds of designers with the expertise you need. So feel free to dream up those ideas—we'll make them happen. To see how we're granting wishes for other companies, call 800.746.6223 or visit **www.cadence.com/24hours**

cadence®

how big can you dream?™

If following a particular convention starts to impact your performance or development cost, it's time to ditch the convention.

You will have to answer these questions for yourself. But let's consider our humble goal. I just wanted to create an abstract scalar type that would tell the compiler to prevent me from making stupid mistakes like "length = time" or "seconds = milliseconds." I also wanted to be able to overload operators so that the answers would come out in the right units automatically. Any programmer who wanted to use this class would have to be familiar with the implementation in order to use different units. So much for low maintenance.

So we have gone to extremes to do what should be a very simple thing, and we still don't have a good solution. This is probably why, when it comes to using the right units, most of us rely on code review and testing to ensure that we get it right.

Getting it right

If you are going to rely on humans to do the job of checking units, you might as well make it easy on them. There are lots of ways to screw things up, so the best you can do is to try to make that less likely. The first line of defense in this area is the names you choose for your variables.

No modern programming languages I know of enforce any rules about how you name your data, or allow you to enforce your own. Many coding conventions out there say how you should name your variables, but they are usually focused on aesthetics. It seems we are more concerned about which letter should be capitalized, or where to put the underscore than we are about whether the name is inconsistent or misleading. Often we don't realize how ambiguous some terms are until we have to revisit the code much later. What seems clear today may be incomprehensible six months from now.

Consider a term like "rate" used to describe a change in some value over time. You might name a variable "transfer_rate" or "flow_rate," but you are missing a key piece of information—the units. To measure a data transfer rate, you could be talking about bits per second, baud, bytes per second, megabytes per second, and so on. Typically you might put a comment in the declaration to indicate the units, but if that declaration is in another file, chances are good that no one will look for it.

Why not use the name to indicate the convention in use? For example, we could use the name "transfer_rate_bits_per_sec." This tells us exactly how it was calculated and how to use it. What if you are using system clock ticks as your unit of time measurement? It would be a waste of CPU cycles to scale your clock ticks into seconds. In this case, you could skip the scaling and use a name like "transfer_rate_bytes_per_tick," making it clear that you are not using seconds for measurement. This is not part of any standard convention, but it communicates an important piece of information that the reader needs to know. Another programmer should at least get a clue from the name that he needs to know what a clock tick is before he makes any assumptions about this value.

Putting units in your variable names may seem clunky at first, but it adds value to your code. For one thing, it gives more meaning to a variable than a generic term like "rate" or "length." If the name consists of real-world measurements, the reader can infer information about the variable just by knowing about the application. For instance, if a variable contains the speed of a car in miles per hour, you have a fairly good idea what the range of values is for that variable. You also know that an expression like "speed_mph = distance_meters/delta_time_seconds" is wrong. You can see that just by looking at the names. No need to sift through header files or design documents, it's right there for everyone to see.

What I like about this approach is that it is simple to apply and follow. You do not need strict rules to apply this, just common sense. You don't need a dictionary to follow the code as long as you stick to familiar abbreviations. A code review can help alert you to ambiguous abbreviations or unfamiliar terms.

Another benefit of this approach is that incomplete compliance is better than none at all. Even with tens of thousands of lines of legacy code, you can still improve the code by applying this to just the code you add or modify. Unlike some naming conventions that don't work unless they are strictly followed throughout the code, putting units in variable names can be done anywhere at any time.

Naming guidelines

Following a convention like SI goes a long way toward making your code easy to follow, but if following a particular convention starts to impact your performance or development cost, it's time to ditch the convention. That's where the names are most important. You should be free to use whatever units are necessary to get the job done, but use the names to inform the reader.

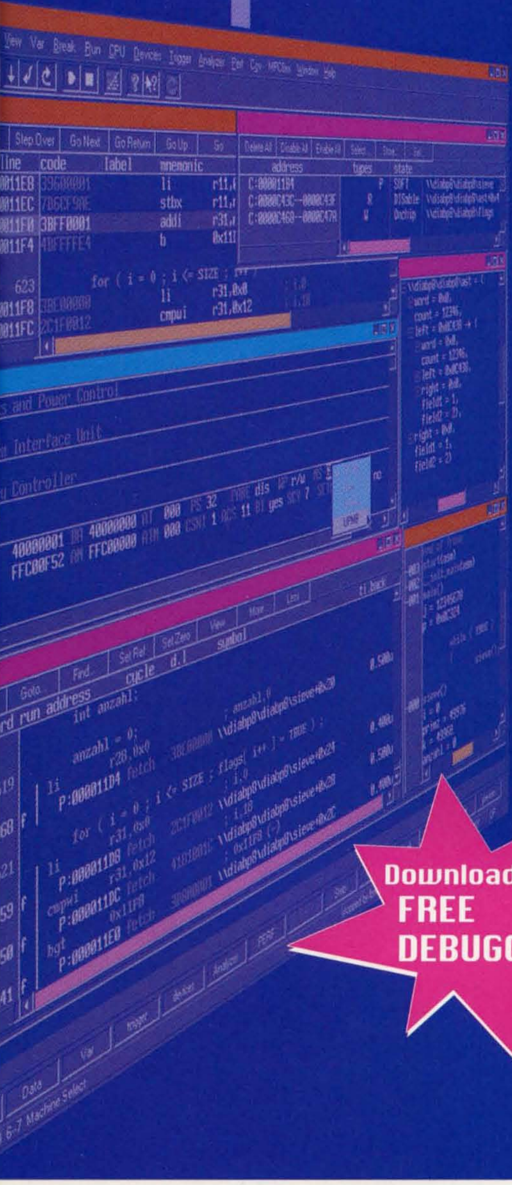
The goal is to make the code as easy to follow as possible, so that code reviews will catch as many errors as possible. Stick to familiar terms and abbreviations in your names using SI as an example (see Table 1). Keep in mind that the SI abbreviations are case sensitive. For instance, "M" is the abbreviation for "Mega," but "m" is the

TRACE32®

IN-CIRCUIT-EMULATORS & DEBUGGERS

ONE IDE SERVES ALL!

TRACE32-PowerView



TRACE32-ICD
the highly cost effective In-Circuit Debugger

TRACE32-FIRE
the Fully Integrated Risc Emulator at highest speeds

TEST THE BEST!

- *Professional tools exceed highest demands*
- *The ultimate in technology and speed*
- *Greatest variety of supported CPUs (ARM, C167, H8S, PowerPC, SH2, ST10 and more)*
- *Unique modularity and open concept*
- *Test and compare our comfortable tools by downloading our simulators*

electronica 2000
Munich, Germany
Nov. 21-24, 2000

Download FREE DEBUGGER!

Lauterbach, Inc.
4, Mount Royal Ave
Marlborough, MA 01752
Fax: 508-303-6813
Phone: 508-303-6812
e-mail: info_us@lauterbach.com

Lauterbach, Inc.
13256 SW. Hillshire Drive
USA-Tigard, OR 97223
Fax: 503-524-2223
Phone: 503-524-2222
e-mail: info_us@lauterbach.com

LAUTERBACH

Free simulators available in the Internet:

http://www.lauterbach.com/download.html

You should be free to use whatever units are necessary to get the job done, but also use the names to inform the reader.

How Many Ways Can You Tell Time?

The ANSI C clock() function is one of many time related functions in the ANSI standard that leaves much to be desired in terms of a name. The time() function is just as vague, but this at least has the advantage of a long legacy that you can at least assume that programmers will be familiar with it.

The clock() function is supposed to tell you how much CPU time your program has consumed, but there are a few gotchas. For one thing the name tells us nothing about the units in the return value, and the return type "clock_t" is equally uninformative. I have seen more than one manual that told me to divide the return by CLK_TCK to get the time value in seconds, but this is wrong. Again, we have a macro with a bad name. The only thing CLK_TCK has in common with clock() is that they use the same units (whatever those are). It seems that the unit of clock_t and the definition of CLK_TCK are implementation defined. Unfortunately some early implementations of ANSI C left out a critical macro—the one which tells you the units, which is more clearly named CLOCKS_PER_SECOND.

So what is CLK_TCK then? CLK_TCK is supposed to tell you how precise the timer used by clock() is. That is, if CLOCKS_PER_SECOND tells you that the units of clock() are in microseconds, then CLK_TCK tells you how many microseconds there are between each clock tick. Obvious isn't it? This gets even more confusing when your implementation incorrectly defines CLK_TCK and CLOCKS_PER_SECOND as the same value. This is true of the Cygnus distribution of GCC for Windows (also known as Cygwin B20). Faulty software that uses CLK_TCK instead of CLOCKS_PER_SECOND will work until a compiler change makes these macros take on different values. Fortunately, hardly anyone uses this function anyway. I can't imagine why not.

POSIX improved things with the clock_gettime() function. This function takes as its parameters a clock ID and a "tv" structure containing two fields—tv_sec and tv_nsec. Notice how the structure element names tell you that the time is in seconds + nanoseconds. The clock ID allows for implementation-defined clocks, but every implementation is required to provide a CLOCK_REALTIME clock ID. So this function can accommodate high precision clocks and is a bit more informative than the simpler ANSI functions.

abbreviation for "meter" and "milli." When in doubt, spell it out. This is especially true if you are going to use some of the less common abbreviations. For example:

- 1/1000th of a volt = "mv" or "mV" are okay, "mVolts" or "milliVolts" are even better
- 1/100th of a volt = "mvx10," "mVx10," "mVoltsx10," or "milliVoltsx10" are okay but "centiVolts" is would be unusual and "cV" would be cryptic
- 1/10th of a volt = "mVoltsx100" or "milliVoltsx100" but "deciVolts" would be unusual and "dV" would be cryptic

These names start to get ugly when you use derived units like "velocity_meters_per_second," but at least they're not ambiguous. Sometimes you should consider inventing your own units to avoid confusion. For instance, maybe you want to use a pixel as a unit of distance. In that case, don't call it a millimeter, call it a pixel. You can then define a well-named constant like `PIXELS_TO_MILLIMETERS` that will convert your pixels into appropriate units.

Another example of when you might want to invent your own units is when you need to manipulate raw data from an ADC. Chances are that the raw data is in arbitrary units, so no SI units apply unless you scale the data first. In this case just call it something like "adcCounts." Any programmer or reviewer should understand that these are raw counts that need to be scaled before they can be used.

Of course, assigning units to every scalar does not make sense. To paraphrase Freud, sometimes an `int` is just an `int`. Examples of this would include loop counters and bit flags. Use good judgement and listen to code review feedback. In the end you should have code that is easier to understand and fewer bugs. **esp**

John Fusco received his BSEE from Polytechnic University in Brooklyn NY. He has been designing and programming embedded systems for over 10 years. Currently he is with GE Medical Systems where he designs image reconstruction software for CT scanners. You can e-mail him at fuscoj@execpc.com.

References

1. Eckel, Bruce. *Thinking in C++, Vol. 1 (2nd Edition)*. Englewood Cliffs, NJ: Prentice-Hall, 2000.
2. For more information about standard units, consult the National Institute of Standards and Technology: *physics.nist.gov/cuu/Units/index.html*. Special Publication 330 discusses SI units in detail.
3. For more information about the details behind the loss of the Mars Climate Orbiter, see the official report available from JPL: *ftp://ftp.hq.nasa.gov/pub/pao/reports/1999/MCO_report.pdf*.

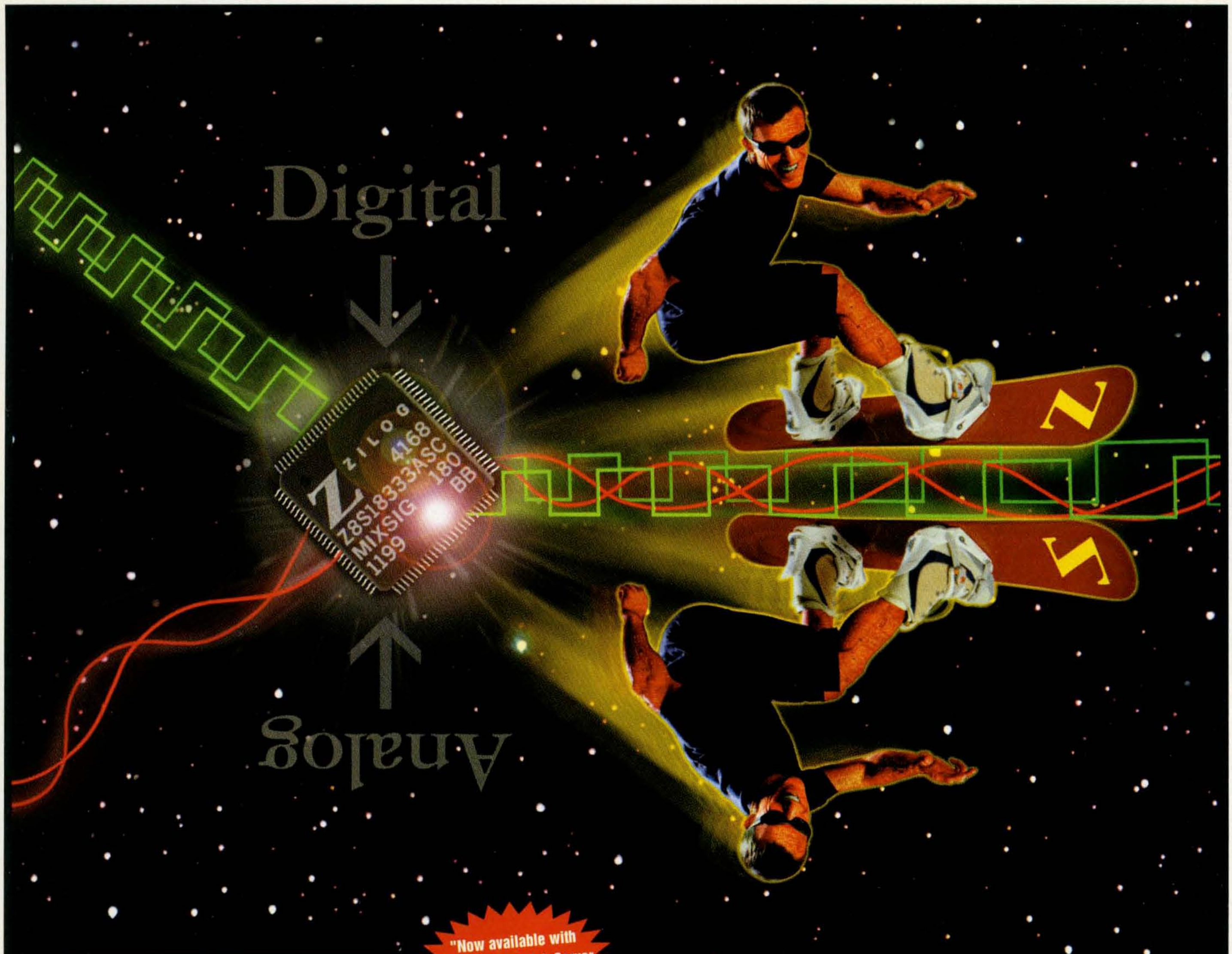
Some of the biggest names in the business come to us
for smart networking solutions. We're glad to chip in.

Looking for a smart solution for your next-generation router, switch, server, modem, or desktop device? There's only one name to know: DigitalDNA from Motorola. DigitalDNA chips, systems, software and ideas — embedded solutions that help smart companies create smart products. And you'll find it in leading-edge technology like 0.10μm copper interconnect CMOS-based solutions. The world's fastest copper interconnect SRAMS. And a wide portfolio of highly scalable PowerPC cores. So if you're ready to go big time with your networking or computing design, we're happy to help.

www.digitaldna.motorola.com

DigitalDNA™
from Motorola

THE HEART OF SMART™



"Now available with an Embedded Web Server Software Suite for Internet applications."

Z80S183 Features

• 8-ch 10-bit 500Ksps ADC	• 2K SRAM
• 10-bit 1 us DAC	• 6 low power modes
• 100% Z80/180 code compatibility	• 1K boot ROM w/virtual loader
• 1 Megabyte address space	• 8-ch Programmable Output Generator / improved PWM
• Real Time Clock / Calendar w/alarm	• Watch Dog Timer
• New ZDI Debug Interface	• Power On Reset
• 2 Enhanced UARTs	• Low Power PLL oscillator
• 2 High speed DMA controllers	• Economical 100-VQFP package
• 2 16-bit Counter-Timers	• DC to 33 Mhz @ 5 V +/- 10%
• 32-bits General Purpose I/O	• DC to 20 Mhz @ 3.3 V +/- 10%

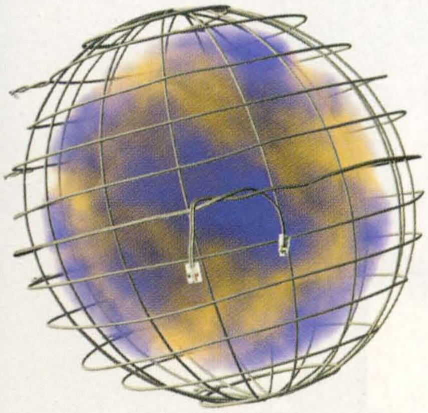
To speed your design integration time, we're giving you mixed signals.

The Z80S183 is a mixed signal Z80 with a super-rich peripheral set all on one chip.

ZiLOG extreme connectivity©

www.zilog.com/Z183

Michael Barr



Disconnecting

Internet Appliance Design

Even in this connected age, it's sometimes necessary to disconnect for a while. For example, you can't roam to every square kilometer of the planet with any wireless technology currently available. So while networks are valuable, being off-network is a practical reality that must be dealt with—by both users and developers.

Several different behavioral models exist for handling off-network situations. In the case of cell phones, the network generally handles disconnections automatically. Incoming calls may be routed to a (network-side) voice mailbox instead of your unreachable handset. If the caller opts to leave a message, the network will let you know about that as soon as you are again within range.

Another off-network behavioral model operates like a digital camera, wherein the device is still useful, though perhaps in more limited ways. For example, you may only be able to photograph a few dozen images before running out of storage capacity in your digital camera. And you certainly won't have access to the image manipulation software and color printing capabilities that make digital photos so much fun.

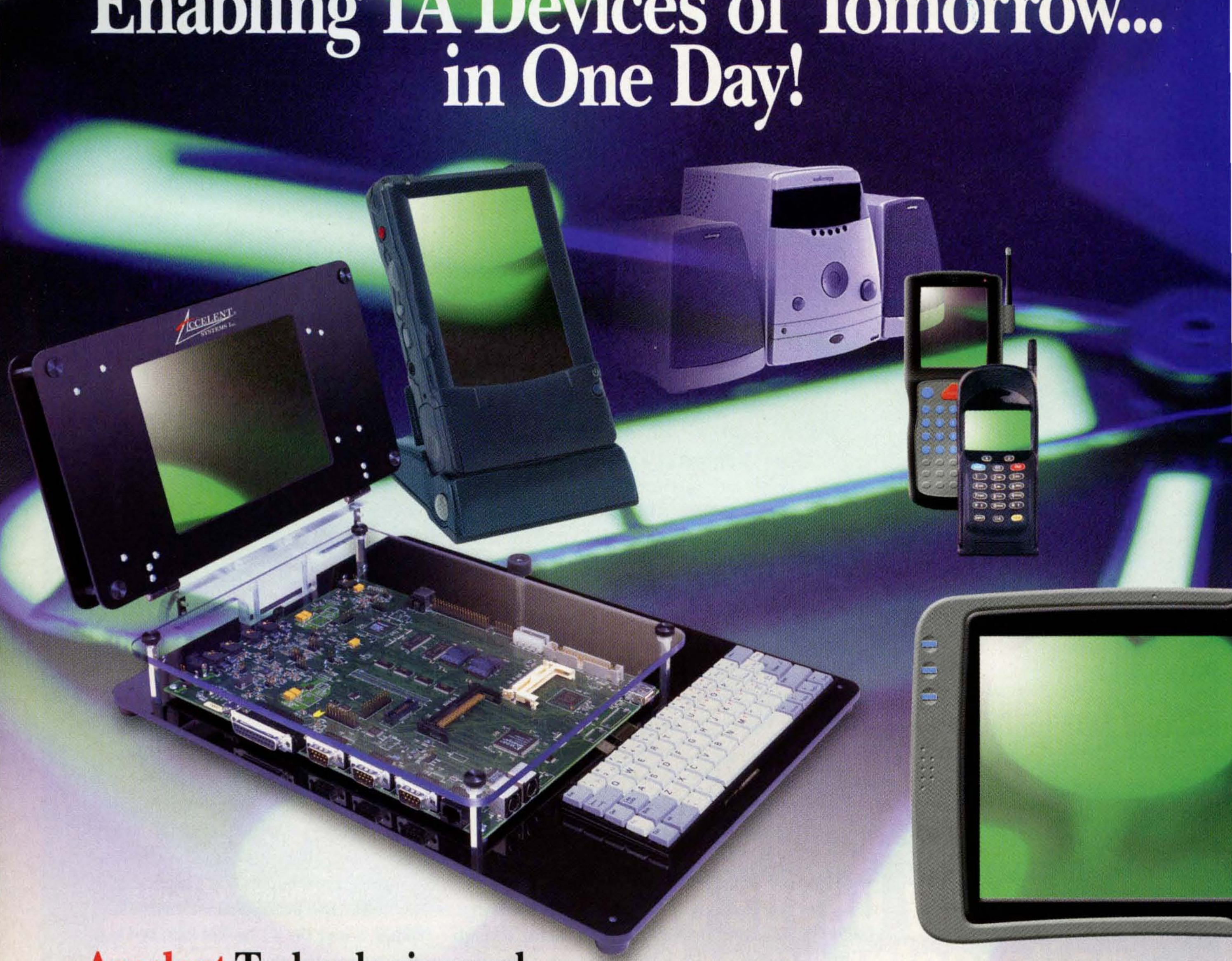
This month, the Internet Appliance Design section focuses on two technologies for adding new capabilities to embedded devices. The first of these is the Wireless Application Protocol. Despite its heavy association with Web-enabled cell phones, WAP has much broader potential applications. Specifically, its Wireless Markup Language—a stripped down version of HTML—could be used to develop interactive data content for any display-limited device. The scripting language WMLscript is also available to embedded developers for all sorts of applications.

Another useful connection technology is USB. One of the perceived problems with this PC-centric bus is the need to develop a number of operating system drivers (Windows 98, Windows 2000, and Linux to name just three) for each new device you might care to develop. However, that's not always entirely true. Many embedded devices can get by with a bare bones amount of USB firmware and no new drivers at all—at least on modern Windows platforms. This opens the door to embedded devices not intended to be utilized primarily alongside PCs to include USB support anyway. For example, a piece of stereo equipment might have a USB connector and firmware tossed in, so that a user desiring to hook that product up to their computer could do so. (I can already hear the lawyers for the record labels grumbling at that specific thought.)

I hope you enjoy these Internet Appliance Design articles and the ones to come. I'm taking a few months away from writing this column. The decision to disconnect is a difficult one, but sometimes you have no other choice. Right now, I'm too busy with the additional responsibilities of being editor-in-chief. I do hope to return to this space sometime early next year. **esp**

Michael Barr is the editor in chief of Embedded Systems Programming. *He holds BS and MS degrees in electrical engineering from the University of Maryland. He is 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.*

Enabling IA Devices of Tomorrow... in One Day!



Accelent Technologies and Products for Fastest IA Product Development

As the leading enabler of IA devices, Accelent Systems can drastically accelerate the IA product development cycle. The power of the Accelent solution is in the underlying technologies that have been developed for devices running on Windows® CE or Linux for ARM® *(including Intel® StrongARM®)* processors. Accelent has packaged its technologies and system software into development and reference platforms, such as WebPad and Pocket PC among others, that allow you to focus on your application. As such, you are able to develop a more complete, reliable IA product in a fraction of the time and cost. That's why we are known as the Fastest IA Solutions Provider™ – FSP™!

Test drive our technology by visiting the ***FSP One Day Design Center™*** at ***www.accelent.com*** or call us with your project requirements and we'll give you ***free operational test code in One Day.*** Qualified visitors will receive a ***free FSP Atomic Clock**** *(as a symbol of our precision solutions)* and will be entered into a monthly drawing for an ***AudioRamp iRad Internet Stereo*** that was developed using our solutions!

ACCELENT® SYSTEMS Inc. ***Fastest IA Solutions Provider***

Phone: 330-864-2300 ▪ Fax: 330-864-8674 ▪ www.accelent.com

*While supplies last, limited quantity per project.

JEFF STEFAN

Working with WAP

You can't run a desktop Web browser on a cell phone or PDA, but the Wireless Application Protocol can give you Web-like access to data, even when resources are limited.

The Wireless Application Protocol (WAP) is emerging as a standard wireless protocol and browser framework for small, limited display devices. WAP is an Internet-enabling protocol, based on XML, that allows data access for cell phones, PDAs, and other low-horsepower systems. WAP is a reaction to two phenomena: the explosive growth of the World Wide Web and digital wireless devices.

Access to the Internet and Web on the desktop is a fairly mature technology, even despite its short existence. For the everyday user, modem speeds are tolerable, access is better than ever, and a large number of Internet service providers make mass access affordable. For companies and power users, technologies such as Digital Subscriber Lines, or DSL, offer ultra high speed, always-connected Internet access. This isn't the case with wireless Web access. Wireless Web is new, immature, and inherently slow. Desktop browsers, such as Netscape Navigator and Internet Explorer easily support complex text and graphics and render them relatively quickly, but the opposite is true with a wireless device. Wireless devices are slow and display-handicapped, but mobile Web access is still a great convenience. Several wireless phone manufacturers recognized this early and formed the WAP Forum (*www.wapforum.org*) to create a specification for a Web access environment based on current Internet and Web technology.

Clients, servers, and proxies

Information on the Web is manipulated by various technologies organized into three basic categories: clients, servers, and proxies. Clients are the programs you use. They allow you to traverse the Web and retrieve and submit information. Servers contain information available to clients all over the world. Proxies act as intermediaries between clients and servers.

The WAP model is based on a scaled down version of the standard Web model.

Typical client applications are web browsers such as Netscape Navigator and Internet Explorer. Browsers make information available to a user by navigating through hypertext links and forms. Unseen client applications may contain Web-enabled connectivity without the user really knowing it. An example is an application that logs data from a remote location and posts the data on a Web page. The client application accesses the data posted on the Web page and presents the data to the user in what appears to be a stand-alone application.

Servers process requests from multiple clients across the Web. Desktop clients typically request servers to retrieve Web pages, navigate hypertext links, and access databases. Database access is achieved via the Common Gateway Interface, or CGI. A server invokes a CGI script using client data to access, process, and return data to the client.

A proxy acts on behalf of the clients it serves, and behaves as a sort of funnel. Clients usually reside behind firewalls. Proxies receive requests from clients behind a firewall and wait for responses outside the firewall. All transactions go through the proxy, and proxies may flag or reject some transactions. Proxies can also be advantageous, especially in intranets. An intranet is an internal network that utilizes Internet protocols. Proxies can cache documents and Web pages, allowing faster access to frequently requested information.

Web browsing on wireless non-PCs

Web technology is fairly mature on PCs and workstations, mostly due to the capability and speed of desktop machines and the rapidly evolved features supported by HTML. The Internet and the World Wide Web are based on wired technology.

It's a different story on non-PCs, such as cellular phones and wireless PDAs. The computational power and display capabilities of handheld wireless devices are feeble compared to desktop devices. Having a four-line LCD display on a cellular phone with primitive graphics capability seems extravagant. PDAs are mild exceptions with bitmapped images and touch-screens. A desktop computer with a Megabit connection makes the Internet seem blazingly fast, but be aware that a cellular phone may be able to receive data only as quickly as 300bps.

With the throughput, display, and computational limitations of handheld devices, running a browser that interprets and displays complex HTML information is out of the question. The alternative is to create a new, scaled down, highly structured language and protocol for limited capability wireless devices and still enable Web access. That's the idea behind WAP.

WAP architecture fundamentals

The WAP model is based on a scaled down version of the standard Web model. The WAP model is consistent with most Web architectural conventions, such as standard naming via URLs and standard protocols. Content typing and content formats are consistent with Web content typing and format. This means that WAP applications and types, such as lists, images, and display formats, are direct analogs to Web applications.

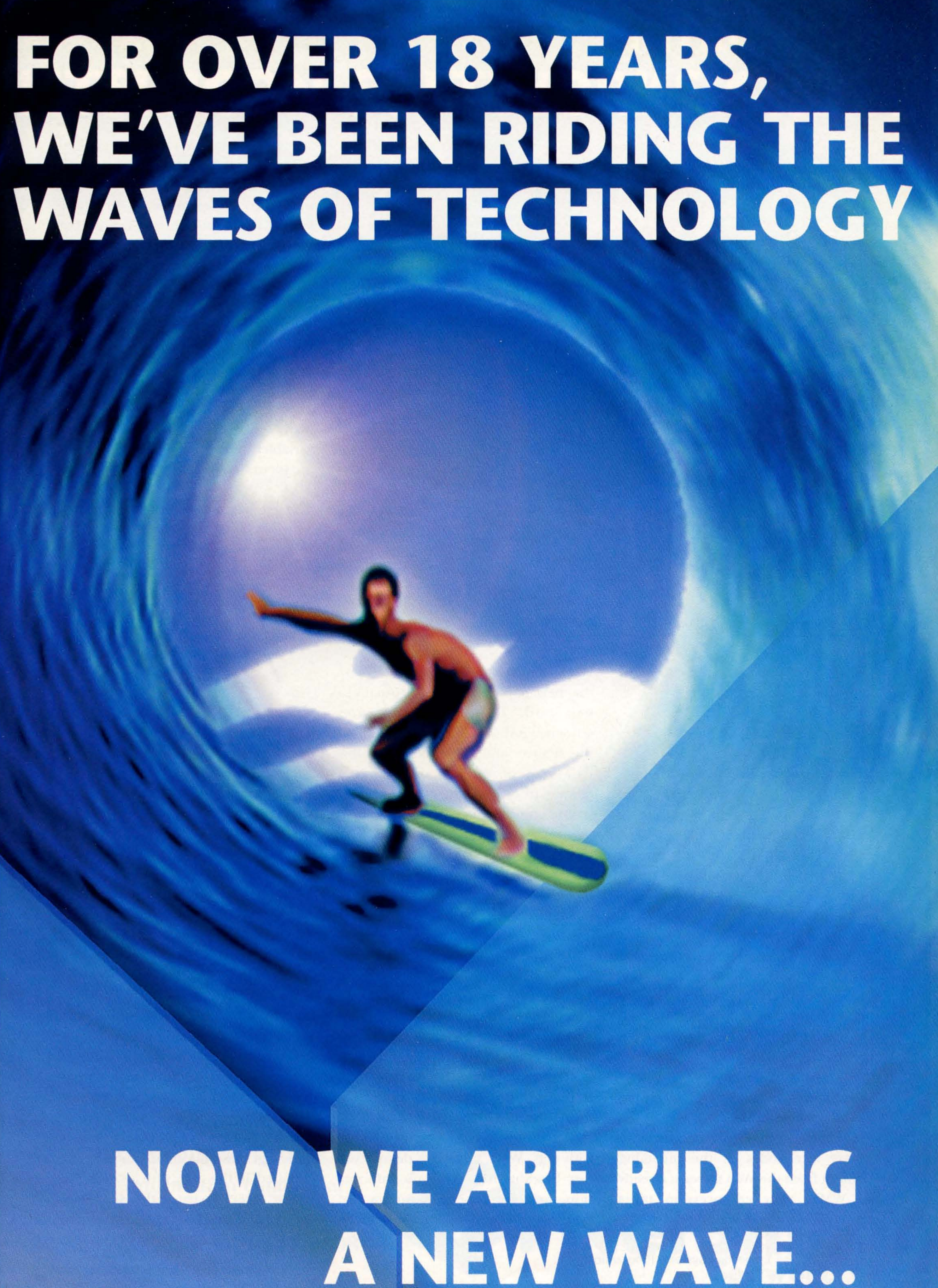
A *card* is the fundamental WAP display unit. A card represents one screen's worth of data. The basic data can be text, a form, a single-bit bitmap, or a menu. There can be one or more cards per WAP client application. Multiple cards reside in a deck. A WAP browser, or user agent, decides how the information in a card is rendered on its particular display. The rendering and application functionality resides in the top-level application layer called the Wireless Application Environment (WAE). The WAE encompasses a micro-browser, supports code written in WML and WMLScript, and also includes the Wireless Telephony Application (WTA). The WTA contains direct access to telephone related functions. Instead of calling the WAP browser a micro-browser or client program, the WAP documentation calls it a User Agent.

WML

The Wireless Markup Language (WML) is derived from the eXtensible Markup Language (XML). XML is itself a markup language derived from the Standard Generalized Markup Language (SGML), best known as the daddy of HTML.

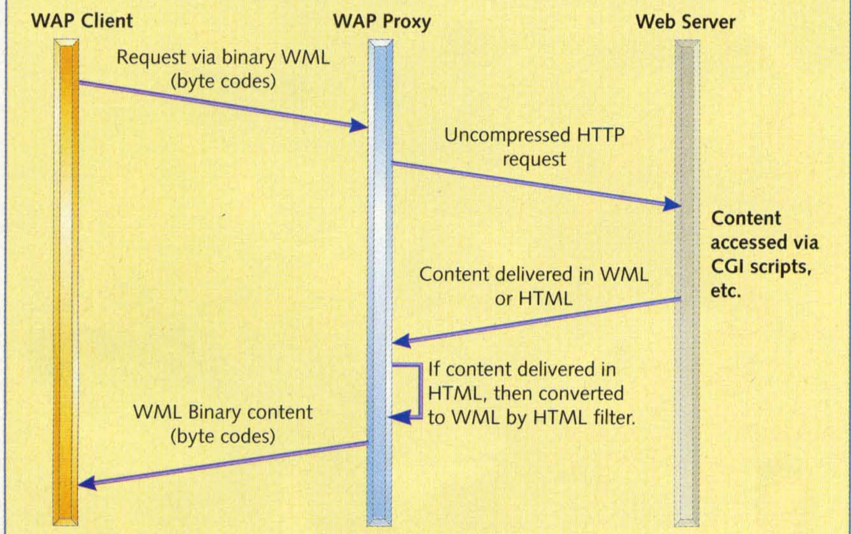
XML allows designers to custom design document types. Whereas everyone developing in HTML is burdened with its all-inclusive nature, XML can be used to create any type of document. HTML must be all things to all application domains. Not so with XML. Custom languages for specific application domains can be created using XML. Rather than accommodating everyone, an XML-based markup language can service a specific domain, such as limited resource wireless devices. This reasoning prevailed when the decision was made to base WML on XML.

WML is designed with small in mind: a small display, a small amount of memory in the target device, and a small amount of computing power. WML applications are composed of cards that reside in *decks.* A deck can contain multiple cards. A card contains *units of user interaction.* These units of interaction are simple menu selections, text entry fields, text formats, and the like. WML also provides an event-handling capability that is used to navigate from card to card


FOR OVER 18 YEARS, WE'VE BEEN RIDING THE WAVES OF TECHNOLOGY
NOW WE ARE RIDING A NEW WAVE...

Every language has a syntax and syntax elements. WML specifies several syntactic units.

FIGURE 1 WAP transaction sequence


WAP Client
WAP Proxy
Web Server
Request via binary WML (byte codes)
Uncompressed HTTP request
Content accessed via CGI scripts, etc.
Content delivered in WML or HTML
If content delivered in HTML, then converted to WML by HTML filter.
WML Binary content (byte codes)

within a deck and to control execution of WML scripts.

WML structure and transactions

WML transmissions usually originate at the WAP client, or user agent. The user agent makes a request via binary WML byte codes. The request is typically a URL. A WAP proxy takes the binary byte codes and uncompresses them. The uncompressed WML request is then translated to an HTTP request. The proxy issues the request to a Web server via uncompressed HTTP. Content is accessed at the server, returning HTML or other data accessed on the Web server via CGI scripts. The content delivered from the server is HTML or WML, if the Web server is WML-savvy.

At the WAP proxy, the incoming HTML is converted to WML by an HTML filter, then compressed into binary byte codes. If the incoming content from the Web server is already in WML, only compression takes place. The content is delivered to the WAP client and the user agent displays the new content, which is usually a new card. The entire transaction sequence is shown in Figure 1. With all this activity and the inherently slow data rate of wireless devices, one comes to appreciate the minimal nature and design of WML.

WML syntax elements

Every language has a syntax and syntax elements. WML specifies several syntactic units. These are *entities*, *elements*, *attributes*, *comments*, and *variables*. An entity is defined as a self contained unit. WML contains two types of entities: numbers and characters. Elements define all deck information, and are delimited by tags. Tags are similar to tags in HTML. The structure is `<tag> information_content </tag>`. If there is no content, then the tag form is `<tag/>`.

Attributes are values associated with tags. WML tags have default values. If additional or specific information must be associated with a tag, then an attribute value needs to be specified within a tag. The general form of an attribute within a tag is `<tag attr="some_value"/>`. The important things to watch for is the lack of white space between the `attr` keyword and the equal sign. The value associated with `attr` must be enclosed in double quotes.

Comments are identified by parsing an exclamation point and two minus signs as the first characters after a left angle bracket. Comments are ended by two minus signs and a right angle bracket. The form is `<!-- comments go here -->`. WML also allows variables in cards and decks. Variables are identified by a single dollar sign, `$`, followed by the variable name. Variables can be of the form `$identifier`, `$(identifier)`, or `$(identifier:conversion)`.

A simple WML deck

One of the simplest forms for a WML deck is a single card with a text prompt:

```
<wml>
<card>
<p>
Working with WAP
</p>
</card>
</wml>
```

A WML deck is enclosed by the tags `<wml>` and `</wml>`. This denotes the beginning and ending of a deck, while `<card>` and `</card>` denote the beginning and ending of a card. The `<p>` and `</p>` denote a paragraph, and the enclosed text, "Working with WAP," is displayed.

This is an entirely static example deck, with no provision for user input, and no soft keys specified. The most common construct for assigning soft keys and taking action is the `<do>` element. A typical `<do>` statement could be `<do type="ACCEPT" label="Next">`. This tag, like the others, must be terminated, in this case, with a `</do>`. The `type="ACCEPT"` part of the `<do>` statement indicates an action type. The

INTRODUCING NETSILICON SOFTWORKS GROUP

NetSilicon & Pacific Softworks joined together to create a new force in the world of embedded software for intelligent devices.

NetSilicon will continue to offer its highly integrated NET+Works hardware and software solutions. The new Softworks Group will provide strategic networking technology to the whole market including NetSilicon's integrated solutions.

The Softworks Group is committed to supporting and expanding its customers for Internet and Web protocols and applications for multiple processors, RTOS and developement tools. Key products include FUSION TCP/IP, PPP, Web Browser, Graphics Toolkit, SNMPv3, RIP2, OSPF2, NAT and SSL.

SUPPORTED PROCESSORS INCLUDE:

ADI SHARC
AMD NET186
ARM Thumb, ARM7 and ARM9
Intel x86,i960 and StrongARM
Hitachi SH Family
Hyperstone E1-32 and E1-16
Motorola 68EN360, Coldfire, MPC860/860T and MPC850
Philips Tri-media
ST Micro TP2/TP3/55xx
Siemens C16x family
TMS320 family
And Many More to come !

Protocols:	TCP/IP, UDP/IP, DHCP, BOOTP
Web:	Web Browser, Web Server and Graphics Toolkit/GUI
Internet:	PPP (Client and Server), SLIP, CSLIP, CHAP, PAP
Applications:	FTP, TFTP, TELNET, DNS, SMTP, POP3, RPC
Network Management	SNMP v1, v2 and v3, MIB Compiler
Security:	SSL
Routing Protocols:	OSPFv2, RIP/RIP2, BGP4, NAT
RTOS:	ThreadX™

Check out the new web site at
www.netsilicon.com/softworks
For additional information email us at
softworks@netsilicon.com
ph: 805-499-7722 or 800-541-9508
fax: 805-499-5512

NetSilicon
Softworks Group

Like any language, WML has syntax rules and requires a parser to determine if a submitted program is correct.

LISTING 1 Two card example navigating with <do>

```
<wml>
  <card1>
    <do type="ACCEPT" label="Next">
      <go href="#card2" />
    </do>
    <p>
      Working with WAP
    </p>
  </card1>
  <!— Card number 2 follows —>
  <card2>
    <do type="ACCEPT" label="Last">
      <go href="#card1" />
    </do>
    <p>
      Working with More WAP
    </p>
  </card2>
<wml>1
```

ACCEPT action type accepts the user press of the soft key and takes the action specified within the <do> and </do>.

A common construct with <do> tags is the <go> element. The <go> element attempts to navigate to another card or another deck. If the <go> element specifies another card, that card is displayed. If a deck is specified, the deck is retrieved and the first card is displayed. A WML listing that includes two cards and a <go> statement is shown in Listing 1.

Two soft keys are assigned in this example, one labeled "Next" in Card1 and one labeled "Last" in Card2. Card1 contains a small text block, "Working with WAP," that is displayed first. If a user accesses the "Next" soft key, Card2 is retrieved and the text "Working with More WAP" is displayed. Both cards are cached on the client device, and the client user agent positions the text independently of the WML program since no formatting information is included. Different user agents display this simple program differently, depending on the client's display capability. A Nokia phone will display the program slightly differently than a Motorola phone.

How does the user agent know that Card1 and Card2 are already on the client device? First, no URL is specified. Second, a hash mark, #, precedes the card name. The hash mark says the card is local. Think of it as a "load immediate" type of instruction.

As a WML author, how do you know if your program is correct? Like any language, WML has syntax rules and requires a parser to determine if a submitted program is correct. WML, being based on XML, has a well specified *document type*. A document type declaration must appear at the beginning of every WML document. The type declaration is:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFO-
  RUM//DTD WML 1.1//EN" www.
  wapforum.org/DTD/wml_1.1.xml>
```

This rather cryptic statement says the WML document is based on XML version 1.0, and the version of WML used is 1.1 and can be found at the URL indicated.

WML navigation and events

Navigation in WML is tied closely to events and tasks. Events are usually bound to tasks and result in navigating to a new URL. Events can be intrinsic or explicit. Intrinsic events enable WML elements to generate events when interacting with a user. If a particular type of action occurs, the event is generated. Intrinsic events are recognized by four WML elements: *ontimer*, *onenterforward*, *onenterbackward*, and *onpick*. These are types associated with an *onevent* element. For example, an onenterforward event is generated when a new card is entered. Explicit events are generated by constructs such as <do> statements, as explained previously.

Tasks

WML supports four major "inter-card traversal" mechanisms, or tasks. The tasks are *go*, *prev*, *noop*, and *refresh*. The go task can take up to 10 steps and finally resolves to navigating to a URL. The prev task can take up to seven steps and results in popping a URL off the history stack and locating a card assigned to the URL. The noop task does no processing. The refresh operation redisplays the current card. This operation is used if any variable's state changes affect the display and need to be displayed. If any of the tasks fail to execute, the currently displayed card is maintained without any changes to variables, bindings, or state.

WMLScript

WML, like HTML, is static. WMLScript, like JavaScript, adds dynamic capabilities to WML.

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

WMLScript functions follow specific rules and have rigid constraints, such as always returning a value, passing parameters by value only, and not allowing function nesting.

WMLScript adds programming language constructs such as if-then statements. WML is weakly typed, and it's easy to lose track of variables between native WML and WMLScript.

WMLScript looks much more like a conventional programming language than WML. Since WML is a static language, some serious issues come up, such as user data input validity testing, target device resource access, and local message generation to users. How does WML know if a user entered a name instead of a phone number? What if the user enters something wrong and an error message must be displayed to alert the user? Does this mean that a new card must be accessed through a WAP proxy and Web server just to get a simple error message? WMLScript addresses these problems and offers enough high level language features to solve them.

WMLScript is function-oriented. There is no call to `main()`, as in a C program. All functions are stored in *compilation units* and stored on a server for execution. The forms of WMLScript functions are shown in Listing 2.

WMLScript functions follow specific rules and have rigid constraints, such as always returning a value, passing parameters by value only, and not allowing function nesting. WMLScript supports two types of functions: external and internal. External functions are declared when the function is used outside of its compilation unit. This is like making a class member public in C++. If a function is used strictly within its local compilation unit, then the function is declared normally and is not visible to the outside world. A block of code is like a block of code in C or C++.

Listing 3 shows how to link WMLScript functions to a WML page. This example uses the WMLScript library function `setVar()` to set a variable in a WML program. The example displays the number 5 on the client's display. The function `GetNum()` is called from within the `<go>` element in Card1. The WML program stops execution, `GetNum()` is referenced and interpreted, and the results are returned in the WML variable NUM in Card2. The WMLScript function uses

LISTING 2 WMLScript function formats

```
// for externally accessible functions
   extern function FunctionID(params)
     {
        BlockOfCode;
     };

// for local calls only
   function FunctionID(params)
     {
        BlockOfCode;
     };
```

LISTING 3 WML/WMLScript variable interaction

```
<wml>
  <!— This file is named num.wml —>
  <!— Card1 calls GetNum() —>

  <card1>
  <p>
    <do type="accept" label="Get">
      <go href="num.wmls#GetNum()"/>  <!— call GetNum() —>
    </do>
  </card1>

<!— Card2 displays result of GetNum() —>
  <card2>
    <p>
      Num Is: $(NUM)
    </p>
  </card2>
</wml>

// This function is in num.wmls
extern function GetNum()
{
  var number = 5;
  WMLBrowser.setVar("NUM", number);
  WMLBrowser.go("num.wml/#card2);
};
```

PUT THE INTERNET IN

AND YOU'LL BE AMAZED AT WHAT COMES OUT

YOU ALREADY KNOW THAT CAPTURING INFORMATION FROM NON-NETWORKED DEVICES MEANS CONNECTING THEM TO THE INTERNET. BUT DO YOU KNOW HOW EASY IT IS? DO YOU KNOW THAT WITH OUR DEVICE SERVERS, YOU CAN ADD INTELLIGENCE

WITH DEVICE SERVER TECHNOLOGY™ YOU CAN INTERNET-ENABLE ANY DEVICE.

TO ANY DEVICE AND TAP INTO A WHOLE WORLD OF POWER AND INFORMATION? WELL YOU CAN. YOU CAN INCREASE THE LIFE AND USEFULNESS OF ALL YOUR LEGACY SYSTEMS — WITHOUT HAVING TO MODIFY THEM. YOU CAN INTERNET-ENABLE THE DEVICES YOU HAVE TODAY AND EVEN THE ONES YOU'LL HAVE TOMORROW. WHAT'S MORE, YOU CAN IMMEDIATELY BEGIN ACCESSING AND SHARING INFORMATION ACROSS NETWORKS AND COMMUNICATING MORE EFFECTIVELY THAN EVER BEFORE. AND BY COMBINING OUR DEVICE SERVERS TOGETHER WITH OUR ADVANCED SOFTWARE OR OTHER REVOLUTIONARY APPLICATIONS, YOU'LL HAVE A COMPLETE INTERNET-ENABLED, TURN-KEY SOLUTION. SO THERE'S NO NEED FOR BULKY AND EXPENSIVE PC STATIONS. THIS IS THE POST PC ERA. WELCOME TO IT.

INTELLIGENT INTERNET CONNECTIVITY

By utilizing Device Server Technology™ you can make all your devices more intelligent and valuable, allowing them to interact dynamically and independently with other devices, data servers and clients anywhere via the Internet. And you can do it all quickly and affordably — without any device modifications.

ACCESS From easy installation to easy access, Device Server Technology enables new levels of connectivity and communication.

MANAGE Device servers are managed by using standard web browsers, so there's no need to modify the device.

CONTROL Our unique Software Developer's Kit enables you to write your own software and modify or enhance the devices' performance over the Internet.

From hardware to software, Lantronix provides complete turn-key solutions. We're ready to become your partner in the Post PC era. For more information regarding our Device Server Technology, please visit our website at **www.deviceserver.com** or call us at **877-462-9681.**

LANTRONIX

INTELLIGENTLY LEVERAGE THE INTERNET.

The WAP architecture is built on a six-layer protocol based on the Open Systems Interconnect (OSI) model.

LISTING 4 User input check

```
function badValue(number)
{
  if (number > 10)
  {
    Dialogs.alert("Number > 10");
  };
};
```

the WMLBrowser library functions `setVar()` and `go()` to set the value of the WML variable NUM and to return to Card2.

A WMLScript source code file is parsed by a lexical and syntactic analyzer and translated into binary byte codes. The result of a WMLScript compilation is a compilation unit, which contains user-created functions along with calls to pre-defined library functions. The byte codes contained in a compilation unit are interpreted by a WMLScript interpreter. The functions within a compilation unit residing on a server are accessed by referencing a URL followed by a hash mark (`#`), the function name, and a parameter list. An example is `call http://www.host.com/script#AddIt(1,2)`. The function is `AddIt()` with parameters `1` and `2`. The byte-codes for the function are retrieved and processed by the WMLScript interpreter on the client.

WMLScript library functions

WMLScript supports a large array of useful library functions. The library functions are grouped into six classifications: `Lang`, `Float`, `String`, `URL`, `WMLBrowser`, and `Dialogs`. WMLScript library functions are called by prefixing the specific library. For example, to use `getVar()`, the call is `WMLBrowser.getVar("MyVar")`, similar to accessing a member function in C++.

The `Lang` library is a catch-all source of functions that are related to core WMLScript. The `Lang` functions range from determining the absolute value of a numerical parameter to determining the character set supported by a WMLScript interpreter. The `Float` library supports floating-point functions typically used in an application, such as `floor()` and `ceil()`. The `String` library contains string manipulation functions such as `compare()` and `length()`. The `String` library also contains some handy instructions such as `find()`, which locates a substring, `squeeze()`, which removes extra white space between strings, and `trim()`, which eliminates excess leading and trailing white space in a string.

The URL library contains functions that manipulate URLs. The URL syntax supported is as follows:

```
<scheme>://<host>:<port>/<path>;
  <params>?<query>#<fragment>
```

The URL library functions range from checking the validity of a URL to resolving absolute URLs. The majority of the URL functions are prefaced by `get`, which returns URL content and string information. Functions such as `getParameters()` and `getPort()` return the parameter list used in a URL and the port number specified in a URL, respectively.

The WMLBrowser library contains the functions that link to WML variables. Important functions such as `getVar()` and `setVar()` retrieve and set variables in a WML program. Other functions in the `WMLBrowser` library have the same name as base WML functions, such as `go()` and `prev()`. This duplicate naming can be a source of confusion if an author isn't careful.

The `Dialogs` library contains user interface functions, such as prompts and alerts. These beneficial functions add quality to the user interface and save time, since the dialogs can be cached on the client device. If a simple prompt, confirmation, or alert is needed, the user agent doesn't need to generate requests to the WAP proxy, but simply accesses the code on the client. These are useful functions when responding to invalid user input. If a user inputs an out-of-range number, a simple call to `alert()` displays the error, as shown in the code segment in Listing 4.

WAP protocol stack

Underneath the Wireless Application Environment lies the rest of the WAP Protocol Stack. The WAP architecture is built on a six-layer protocol based on the Open Systems Interconnect (OSI) model, as shown in Figure 2. If you get involved with WAP development, you'll be in one of two WAP implementation camps. You'll either write WML and WMLScript applications, or implement the stack that enables the applications. Writing WML and WMLScript is relatively easy, but porting the remainder of the stack to a target machine is a difficult and daunting task.

The first underlying layer is the Session Layer. The Session Layer contains the Wireless Session Protocol, or WSP. The WSP offers both connection-oriented and connectionless services. Connection-oriented services maintain a virtual connection throughout a session. Sessions may be suspended and resumed at a later time. The connectionless services provide datagrams that do not maintain a connection, similar to UDP.

Underneath the Session Layer is the Transaction Layer, which contains the Wireless Transaction Protocol, or WTP. The WTP is designed with thin clients in mind. The Transaction Layer supports three transaction classes: unreliable one-way requests, reliable one-way requests, and reliable two-way request and reply transactions.

The next two layers down the stack are the Security and Transport Layers. The security protocol is based

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
SH 1, 2, 3, 4	68HC16
SPARCLite	80251
i960	Z80/180
NEC V85x	

See our website for additional microprocessors we support.

With thousands of embedded applications and 25 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.

Stop by booth 1536 and learn about U S Software's new voice over IP technology

- **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/DOS compatible file system with CompactFlash and CD-ROM support.
- **NETPEER™** Distributed networking platform with comprehensive desktop interface including GUI, database, multimedia and more.
- **SUPERPEG™** Embedded GUI for SuperTask!®
- **IRPRO™** IrDA protocol stack with SIR and FIR

*Netpeer™ is delivered as a library.

800-356-7097

WWW.USSW.COM

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

The Web is now part of our daily lives and culture, but it is still somewhat chained to the desktop. WAP is designed to break that chain.

on the Secure Sockets Layer (SSL) protocol. Data integrity is ensured at this layer, and data is encrypted using a private key algorithm. The Transport Layer contains the WDP, or Wireless Datagram Protocol. This layer provides the WAP equivalent of TCP and UDP, with the emphasis on connectionless datagrams. The Transport Layer allows the upper layers to operate transparently to the data requirements of the bearer services. On the bottom of the stack are the Bearer Services. Bearers are the digital phones and the data services they provide.

Developing WAP applications

There are free WML and WMLScript development tools available on the Web. Two are currently available from Phone.com (formerly Unwired Planet) and Nokia. The full WML specification set is available from the WAP Forum. The Phone.com tools require a connection to a server, but this can be localized by configuring a server on the local machine the development tools are downloaded on. Most developers use the free Apache server and configure it locally. The Nokia toolset doesn't require an external server, although you can use one if you like. The Nokia development kit requires the Java runtime library, and recommends 64MB of RAM.

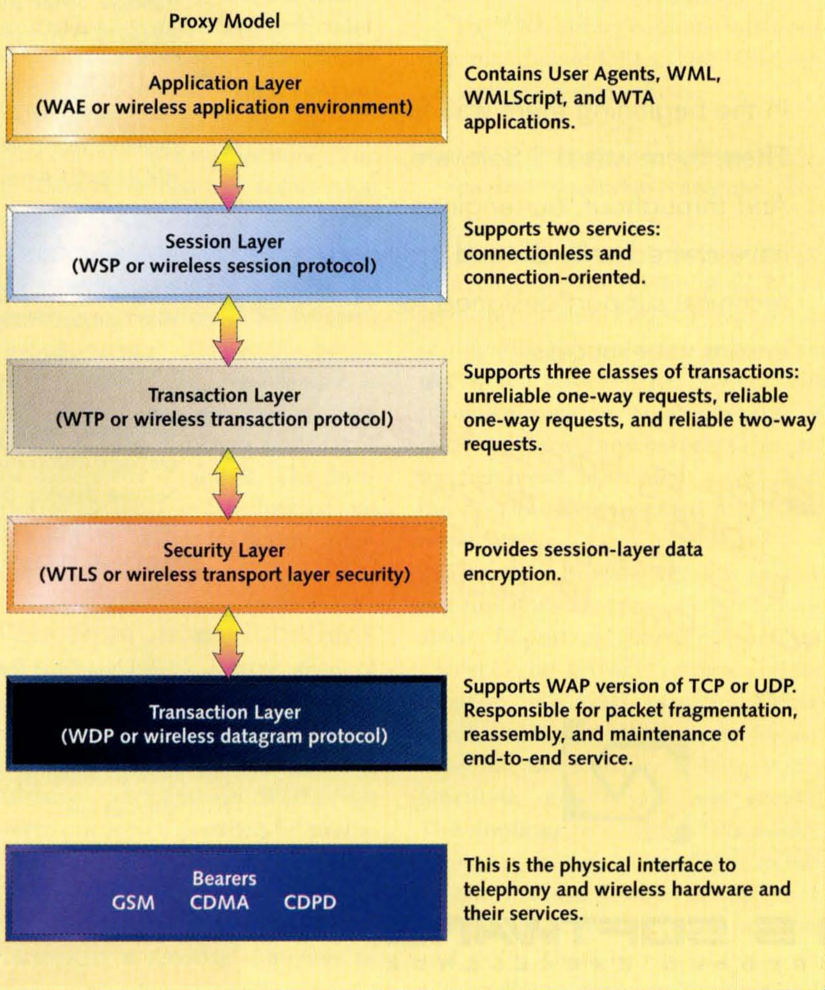
Both development toolsets have well-thought-out user interfaces, and each has a virtual phone handset on the right of the screen where you can see your WML/WMLScript application in action.

Go forth

The Web is now part of our daily lives and culture, but it is still somewhat chained to the desktop. WAP is designed to break that chain. Embedded systems programmers have a great opportunity to create WAP applications for wireless and wired platforms yet to be imagined. **esp**

Jeff Stefan is a system engineer at OnStar. He has worked in embedded systems software for many years, is a frequent speaker at the Embedded Systems Conferences, and is the author of over 15 software articles. He is currently writing his first book, entitled Embedding Artificial Intelligence. *His e-mail address is jmstefan@mindspring.com.*

FIGURE 2 The WAP six-layer protocol stack


Proxy Model
Application Layer
(WAE or wireless application environment)
Contains User Agents, WML, WMLScript, and WTA applications.
Session Layer
(WSP or wireless session protocol)
Supports two services: connectionless and connection-oriented.
Transaction Layer
(WTP or wireless transaction protocol)
Supports three classes of transactions: unreliable one-way requests, reliable one-way requests, and reliable two-way requests.
Security Layer
(WTLS or wireless transport layer security)
Provides session-layer data encryption.
Transaction Layer
(WDP or wireless datagram protocol)
Supports WAP version of TCP or UDP. Responsible for packet fragmentation, reassembly, and maintenance of end-to-end service.
Bearers
GSM CDMA CDPD
This is the physical interface to telephony and wireless hardware and their services.

Resources

WAP Forum. This is the WAP mother ship. The full set of WAP technical specifications are found on the WAP technical link: *www.wapforum.org*.

Nokia Wireless Data Forum. This offers a WAP server trial kit and the Nokia WAP Toolkit along with a well written documentation set that's loaded with good examples. The WAP Toolkit is free but requires developer registration: *www.forum.nokia.com/developers/wap/*.

Phone.com. Go to the Developer Web Site to download the development kit. Like Nokia, the toolkit is free but requires registration: *www.phone.com/developers/index.html*.

Solutions Today... For Tomorrow's Embedded Technology

Let EBSnet help you build your next embedded device!

- All Source Code is Provided
- 100% ANSI "C"
- Royalty Free
- Competitive Prices

Prompt Competent Support

- Consulting
- Satisfaction Guaranteed!

For More Information, Visit Our Web Site

www.ebsnetinc.com

1-800-428-9340

P.O. Box 873 • 39 Court Street
Groton, MA 01450
Phone: 978 448 9340 • Fax: 978 448 6376

http://www.ebsnetinc.com • email: sales@ebsnetinc .com

EMBEDDED TCP-IP NETWORK STACK

The EBSnet Cross Development System provides comprehensive TCP/IP protocols and network applications for embedded CPUs. RTIP 3.0-MPC provides support for Motorola microprocessors including PowerPC, MPC860, 68K, 68360, and Coldfire. Systems are also available for 808x, 80186, 386, ARM, SparcLite, Mips, Hitachi, Mitsubishi, Philips, and Infineon.

INTERNET PROTOCOLS

SUPPORTS UDP, TCP, ARP, RARP, BOOTP, ICMP, IGMP, DNS

DEVICE DRIVERS

Ethernet, 100 Base-T, Uart, PCMCIA based, PCI based

KERNEL DRIVERS

AMX®, CMX®, SMX®, Nucleus®, RtKernel®, Pharlap®, TNT®, ETS®, TNTRT®, PSOS®, RTXC®, RTPX, Polled (no kernel), UCOS®, ECOS®, VRTX®, Threadx®

PORTING LAYER

Ports easily to any realtime OS/CPU

DIAL-UP

SLIP, CSLIP, PPP, modem support

APPLICATION PROTOCOLS

SNMP, DHCP, NFS, FTP, TFTP, Telnet, Mail (SMTP, POP3, IM AP), RIP, and Virtual/Memory File Systems

WEB SERVER

Full featured embedded Web Server (with diskless option)

EMBEDDED WEB BROWSER

WEBSTER, EBSnet's Embedded Web Client, is a complete tool for developing WEB browser interfaces for interactive internet appliances and other embedded systems that wish to use HTML to present a user interface.

WEBSTER is written in portable C++, will port to any target that can support the PEG Graphics Library and uses standard socket calls to access the network.

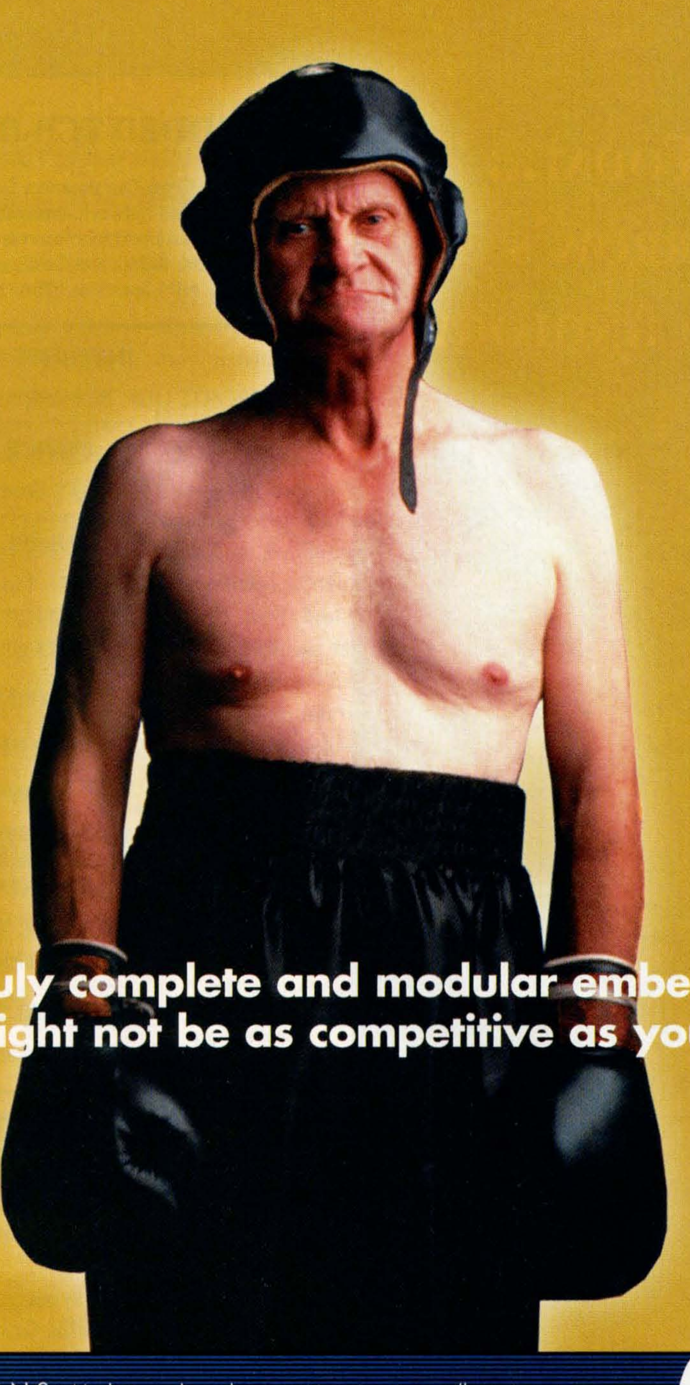
WEBSTER is compatible with EBSnet's RTIP but can be standalone.

ERTFS-DOS COMPATIBLE FILE SYSTEM

Comprehensive, portable, high performance DOS compatible file I/O for embedded systems.

ebSnet Inc

Y2K



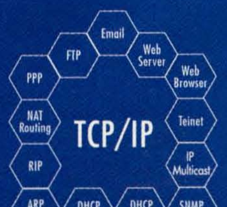
Without a truly complete and modular embedded solution you might not be as competitive as you think.

PROVEN SOLUTIONS: Market needs and competitive situations will continue to change. That's why we have provided solutions over the last ten years that are modular so you can quickly adapt and easily meet these challenges with a minimum of engineering overhead.

interniche
technologies, inc.

Why work with a multitude of companies for your deeply embedded software needs? Interniche solutions include everything you need for building Internet connectivity, embedded Web and DHCP servers, Browsers, routing and more. Today's design needs demand a broad range of solutions. Solutions that are CPU and RTOS independent, and modular so the design-in costs are low.

What's more, we offer more royalty free applications than anyone else. And our solutions are supported by the engineers who write them. That adds up to truly complete solutions. Solutions that let you go the distance.
So follow the lead of companies like ARM, Intel, Ericsson and Green Hills Software, Inc. and design in Interniche solutions today.



Solutions for Building Embedded Web & DHCP Servers, Routing & more.

For more information call 1-408-257-8014. Email sales@iniche.com or go to iniche.com today. In Europe call 46-40-45-85-49.

JAN AXELSON

HIDs Up

The inclusion of a HID-class USB driver in Windows 98 and 2000 makes connecting your embedded device to a PC easier than ever. Here's everything you need to know to create a Human Interface Device.

If you're developing a device that will connect to PCs, chances are these days you'll be using the Universal Serial Bus (USB). USB is designed to serve as a replacement for most of the PC's legacy ports: parallel, serial, and so on.

USB is versatile enough for use with many common peripheral types as well as more specialized devices. Standard peripherals that can use USB include mice, keyboards, disk drives, printers, and scanners. USB is also becoming popular for data acquisition units, test instruments, and monitoring and control devices. Just about anything that in the past would have connected to an RS-232 or parallel port is a candidate for USB.

For many specialized devices, the human interface device (HID) class offers a quick path for USB developers. Although the HID specification was written to meet the needs of mice, keyboards, and similar input devices, it's broad enough to be useful for other devices that need to exchange data in both directions at moderate rates.

Because Windows 98 and 2000 include HID-class drivers, there's no need to write a device driver for your new gadget. And the firmware requirements to classify a device as HID are minimal, consisting mainly of a series of data structures that describe the HID interface and the data to be exchanged.

The main limitation of HID is speed, with a maximum transfer rate of 64KB/s. This is much less than the full-speed bus rate of 12Mbits/s, but still fast enough for plenty of applications.

This article is an introduction to HID development. I'll describe the firmware requirements to enable the operating system to detect the HID and exchange information with it. I'll also introduce the API functions available in Windows for communicating with HIDs. And I'll show how to use a couple of

LISTING 1 These descriptors for a USB joystick are included in the device firmware. They include the device, configuration, and interface descriptors required for every device, plus the HID class descriptor and a descriptor for an interrupt IN endpoint

```
device_desc_table:
        db 12h              ; Descriptor length (18 bytes)
        db 01h              ; Descriptor type (Device)
        db 00h,01h          ; Complies to USB Spec. Release (1.00)
        db 00h              ; Class code (0)
        db 00h              ; Subclass code (0)
        db 00h              ; Protocol (No specific protocol)
        db 08h              ; Max. packet size for Endpoint 0 (8 bytes)
        db B4h,04h          ; Vendor ID (Cypress)
        db 1Fh,0Fh          ; Product ID (joystick = 0x0F1F)
        db 88h,02h          ; Device release number (2.88)
        db 00h              ; Mfr. string descriptor index (None)
        db 00h              ; Product string descriptor index (None)
        db 00h              ; Serial No. string descriptor index (None)
        db 01h              ; Number of possible configurations (1)
end_device_desc_table:

config_desc_table:
        db 09h              ; Descriptor length (9 bytes)
        db 02h              ; Descriptor type (Configuration)
        db 22h,00h          ; Total data length (34 bytes)
        db 01h              ; Interface supported (1)
        db 01h              ; Configuration value (1)
        db 00h              ; Index of string descriptor (None)
        db 80h              ; Configuration (Bus powered)
        db 32h              ; Maximum power consumption (100mA)

Interface_Descriptor:
        db 09h              ; Descriptor length (9 bytes)
        db 04h              ; Descriptor type (Interface)
        db 00h              ; Number of interface (0)
        db 00h              ; Alternate setting (0)
        db 01h              ; Number of endpoints supported
        db 03h              ; Class code ()
        db 00h              ; Subclass code ()
        db 00h              ; Protocol code ()
        db 00h              ; Index of string()

Class_Descriptor:
        db 09h              ; Descriptor length (9 bytes)
        db 21h              ; Descriptor type (HID)
        db 00h,01h          ; HID class release number (1.00)
        db 00h              ; Localized country code (None)
        db 01h              ; No. of HID class descriptors to follow (1)
        db 22h              ; Report descriptor type (HID)
                            ; Total length of report descriptor
        db (end_hid_report_desc_table - hid_report_desc_table),00h

Endpoint_Descriptor:
        db 07h              ; Descriptor length (7 bytes)
        db 05h              ; Descriptor type (Endpoint)
        db 81h              ; Encoded address (Respond to IN, 1 endpoint)
        db 03h              ; Endpoint attribute (Interrupt transfer)
        db 06h,00h          ; Maximum packet size (6 bytes)
        db 0Ah              ; Polling interval (10 millisecs)

end_config_desc_table:
```

free tools that will assist you in developing the firmware and ensuring that Windows can communicate with the device.

For a primer on USB in general, see Jack Ganssle's introduction to the topic, "An Introduction to USB Development" (March 2000, p. 79).

What is a HID?

The HID-class specification is the product of a working group sponsored by the USB Implementers Forum (*www.usb.org*). The forum is the organization formed by the companies responsible for developing the USB specification.

The idea behind classes is to make development easier by defining requirements and behaviors shared by devices with similar functions. An operating system can include a device driver based on a class specification, and a device that conforms to the specification can use the class driver instead of having to provide its own special-purpose driver. This is a big time saver.

The HID class was one of the first USB classes to be supported under Windows. No doubt this was because the class includes essential peripherals like keyboards and pointing devices, which were intended from the outset to use USB. The class specification and additional documents and tools are available from the USB Implementers Forum's Web site.

The term *human interface* suggests that these devices interact directly with humans. With a keyboard or mouse, a human's actions are indeed what determines the data that will be sent to the host. Other examples of HIDs include front panels, remote controls, telephone keypads, and game controls. But a HID doesn't have to have button, knobs, or switches. The specification mentions bar-code readers, thermometers, and voltmeters as other devices that could fall into the HID class.

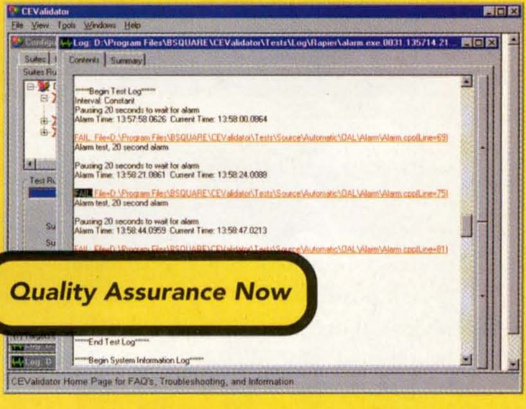
HIDs can also receive data from the host. With a force-feedback joystick,

For every era there's product testing

BSQUARE's QA products and services make your product testing painless.

Put your quality assurance testing on autopilot with BSQUARE's CEValidator™ family of test automation products. Run and record systematic and comprehensive sets of tests for any Windows® CE-based project with the click of a mouse. Complete, thorough testing is at your fingertips. Want us to test it for you? BSQUARE is also your source for custom testing and QA packaged services for all your Windows-based development needs. The QA process has never been faster, easier–or safer.

BSQUARE has helped companies like Motorola®, JVC® and Symbol® Technologies get great products to market fast. **Test us out! Download free evaluation versions at www.bsquare.com/qa**


CEValidator
File View Tools Windows Help
Log: D:\Program Files\BSQUARE\CEValidator\Tests\Log\Rapier\alarm.exe.0031.135714.21...
Contents Summary
Quality Assurance Now
CEValidator Home Page for FAQ's, Troubleshooting, and Information

```
*****Begin Test Log*****
Interval: Constant
Pausing 20 seconds to wait for alarm
Alarm Time: 13:57:58.0626  Current Time: 13:58:00.0864

FAIL: File=D:\Program Files\BSQUARE\CEValidator\Tests\Source\Automatic\OAL\Alarm\Alarm.cpp(Line=69)
Alarm test, 20 second alarm

Pausing 20 seconds to wait for alarm
Alarm Time: 13:58:21.0861  Current Time: 13:58:24.0088

FAIL: File=D:\Program Files\BSQUARE\CEValidator\Tests\Source\Automatic\OAL\Alarm\Alarm.cpp(Line=75)
Alarm test, 20 second alarm

Pausing 20 seconds to wait for alarm
Alarm Time: 13:58:44.0959  Current Time: 13:58:47.0213

FAIL: File=D:\Program Files\BSQUARE\CEValidator\Tests\Source\Automatic\OAL\Alarm\Alarm.cpp(Line=81)

*****End Test Log*****

*****Begin System Information Log*****
```

BSQUARE Solutions

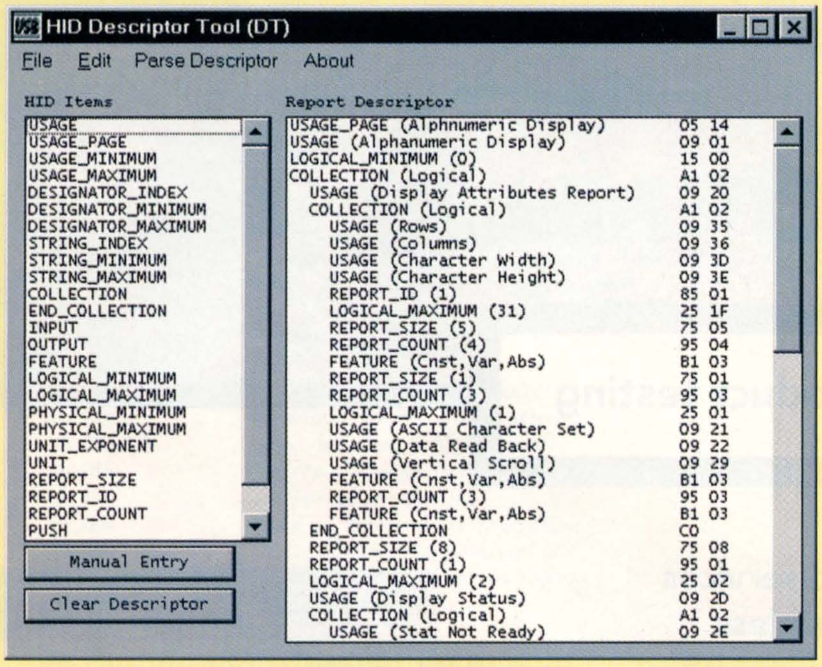
- **Platform development and connectivity**
- **QA tools and services**
- **OS licensing and tools, training, end-user software**
- **Engineering services and support**
- **Solutions for set-top boxes, Web pads, Windows-based terminals, mobile & wireless devices and more**

Innovating tomorrow's devices™

b SQUARE

www.bsquare.com

In short, a HID can be any device that can function within the limits defined by the specification.



```
HID Items
USAGE
USAGE_PAGE
USAGE_MINIMUM
USAGE_MAXIMUM
DESIGNATOR_INDEX
DESIGNATOR_MINIMUM
DESIGNATOR_MAXIMUM
STRING_INDEX
STRING_MINIMUM
STRING_MAXIMUM
COLLECTION
END_COLLECTION
INPUT
OUTPUT
FEATURE
LOGICAL_MINIMUM
LOGICAL_MAXIMUM
PHYSICAL_MINIMUM
PHYSICAL_MAXIMUM
UNIT_EXPONENT
UNIT
REPORT_SIZE
REPORT_ID
REPORT_COUNT
PUSH
```

```
Report Descriptor
USAGE_PAGE (Alphnumeric Display)          05 14
USAGE (Alphanumeric Display)              09 01
LOGICAL_MINIMUM (0)                       15 00
COLLECTION (Logical)                      A1 02
  USAGE (Display Attributes Report)       09 20
  COLLECTION (Logical)                    A1 02
    USAGE (Rows)                          09 35
    USAGE (Columns)                       09 36
    USAGE (Character Width)               09 3D
    USAGE (Character Height)              09 3E
    REPORT_ID (1)                         85 01
    LOGICAL_MAXIMUM (31)                  25 1F
    REPORT_SIZE (5)                       75 05
    REPORT_COUNT (4)                      95 04
    FEATURE (Cnst,Var,Abs)                B1 03
    REPORT_SIZE (1)                       75 01
    REPORT_COUNT (3)                      95 03
    LOGICAL_MAXIMUM (1)                   25 01
    USAGE (ASCII Character Set)           09 21
    USAGE (Data Read Back)                09 22
    USAGE (Vertical Scroll)               09 29
    FEATURE (Cnst,Var,Abs)                B1 03
    REPORT_COUNT (3)                      95 03
    FEATURE (Cnst,Var,Abs)                B1 03
  END_COLLECTION                          C0
  REPORT_SIZE (8)                         75 08
  REPORT_COUNT (1)                        95 01
  LOGICAL_MAXIMUM (2)                     25 02
  USAGE (Display Status)                  09 2D
  COLLECTION (Logical)                    A1 02
    USAGE (Stat Not Ready)                09 2E
```

FIGURE 1 The Descriptor Tool can help you build and test a report descriptor for your device. When your descriptor passes the tool's scrutiny, it's ready to be stored in the device

users can experience effects that match their actions, such as more resistance when pulling the stick to cause a simulated airplane to climb. Other possible HIDs are remote displays, robots, and devices controlled by a virtual control panel on the host computer.

In short, a HID can be any device that can function within the limits defined by the specification. The major features and limitations are:

- A full-speed HID can transfer up to 64,000 bytes per second (64 bytes in each 1ms frame). A low-speed device is guaranteed only 800 bytes per second (eight bytes every 10ms)
- A HID can request the host to poll the device periodically to find out if the device has data to send
- All data exchanged by a HID resides in defined data structures called *reports*. A single report can contain up to 65,535 bytes. The device's firmware must include a *report descriptor* that describes the data to be exchanged. The report format is flexible enough to handle just about any type of data

Device requirements

In many ways, HIDs are no different from other USB devices. Every USB peripheral must contain an intelligent controller that knows how to respond to requests and other events at its USB port.

The controller must have one or more endpoints, which are buffers that store received data or data waiting to be sent. Each endpoint has a number and direction. Endpoint 0 is bidirectional and is required for the control transfers used in enumerating the device.

Every device must respond to a series of control requests defined by the specification. For example, when the host sends a Get_Status request, the device must return status information in the expected format. Every device must also store the descriptors that the host will request.

The device's serial interface engine (SIE) handles the details of sending and receiving the individual bits on the bus, much as a UART does for asynchronous serial communications. Beyond this, the amount of firmware support required to complete a transfer varies with the transfer type and the chip architecture.

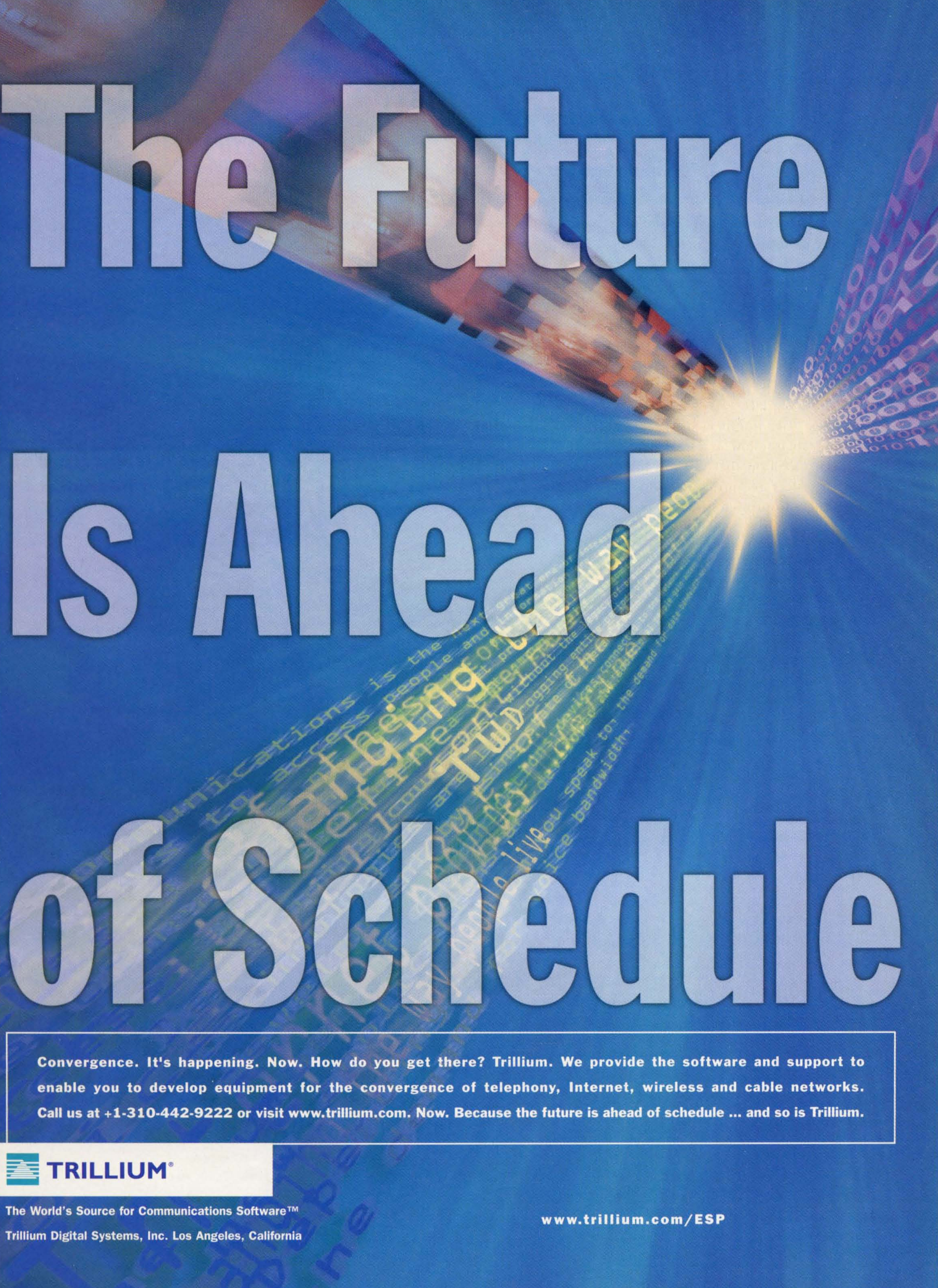
Besides the requirements that apply to all devices, HIDs have additional requirements:

- A HID must have an interrupt IN endpoint for sending periodic data to the host. An interrupt OUT endpoint for receiving periodic data from the host is optional
- A HID must contain a class descriptor and one or more report descriptors
- A HID must support the HID-specific control request Get_Report and may support the optional request Set_Report
- For interrupt IN transfers, the device must place the report data in the interrupt endpoint's buffer and enable the endpoint. For interrupt OUT transfers, the device must enable the endpoint and retrieve received report data (usually signaled by a device interrupt) from the interrupt endpoint's buffer.

Retrieving descriptors

The host computer identifies an attached USB peripheral when it retrieves a series of descriptors during enumeration. The enumeration process is triggered when a device is plugged into the bus or when the host powers up with a device attached.

Listing 1 shows a series of descriptors for a USB joystick. Every USB



Convergence. It's happening. Now. How do you get there? Trillium. We provide the software and support to enable you to develop equipment for the convergence of telephony, Internet, wireless and cable networks. Call us at +1-310-442-9222 or visit www.trillium.com. Now. Because the future is ahead of schedule ... and so is Trillium.

TRILLIUM®

The World's Source for Communications Software™

Trillium Digital Systems, Inc. Los Angeles, California

www.trillium.com/ESP

A HID must also have one or more report descriptors, which are requested after the host has detected that the device is a HID and assigned the class driver to control it.

device has one *device descriptor* and one or more *configuration descriptors.* If multiple configuration descriptors are available, the host selects one during enumeration. Each configuration descriptor in turn supports one or more *interface descriptors.* In a HID's interface descriptor, the class code field is set to 3. Following the interface descriptor is a HID *class descriptor* that specifies the number of *endpoint descriptors* that follow.

Each direction that will use interrupt transfers requires an endpoint descriptor. Control transfers use the default endpoint, which is always enabled and doesn't require its own descriptor.

A descriptor for an interrupt endpoint specifies the endpoint number and direction, the transfer type used (interrupt), the maximum packet size for each transaction (one to 64 bytes for full speed, one to eight bytes for low speed), and the requested maximum latency between transactions.

A HID must also have one or more *report descriptors,* which are requested after the host has detected that the device is a HID and assigned the class driver to control it.

LISTING 2 This sample report descriptor is for a device that sends and receives two vendor-defined bytes of data

```
hid_report_desc_table:

     db 06h, A0h, FFh  ;          Usage Page (vendor defined)
     db 09h, A5h       ;          Usage (vendor defined)

     db A1h, 01h       ;          Collection (Application)
     db 09h, A6h       ;          Usage (vendor defined)

;The input report
      db 09h, A7h      ;          Usage (vendor defined)
      db 15h, 80h      ;          Logical Minimum (–127)
      db 25h, 7Fh      ;          Logical Maximum (128)
      db 75h, 08h      ;          Report Size (8)  (bits)
      db 95h, 02h      ;          Report Count (2)  (fields)
      db 81h, 02h      ;          Input (Data, Variable, Absolute)

;The output report
      db 09h, A9h      ;          Usage (vendor defined)
      db 15h, 80h      ;          Logical Minimum (-128)
      db 25h, 7Fh      ;          Logical Maximum (127)
      db 75h, 08h      ;          Report Size (8)  (bits)
      db 95h, 02h      ;          Report Count (2)  (fields)
      db 91h, 02h      ;          Output (Data, Variable, Absolute)

      db C0h           ;          End Collection

end_hid_report_desc_table:
```

Windows' Device Manager uses .inf files to decide which driver to assign to a device. HIDs can use the default file for HIDs (`hiddev.inf` for Windows 98, `input.inf` for Windows 2000), or you can provide an .inf file that contains your Vendor and Product IDs. The advantage of a custom .inf file is that you can include a device description and manufacturer name that will appear in the Control Panel, in place of the generic "Standard Device" listing.

Reports

A HID can send and receive reports using a device's control endpoint and interrupt endpoint(s). HIDs don't support USB's bulk or isochronous transfer types.

Control transfers have no guaranteed delivery time. The host controller fits the transfers in as best it can. Ten percent of the bus bandwidth is reserved for control transfers, but the host may allocate some of the time to other devices.

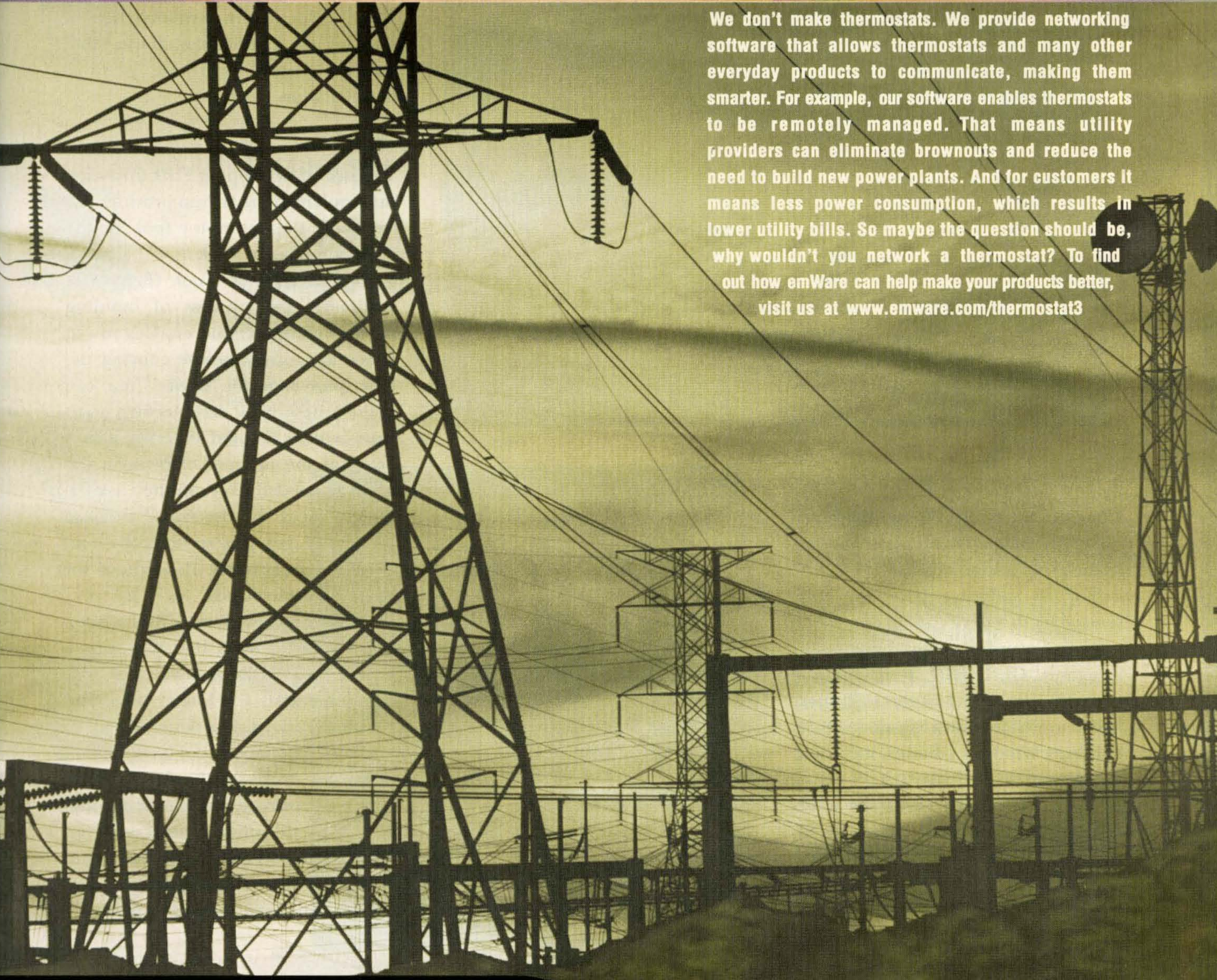
Interrupt transfers have a guaranteed maximum latency, or time between transaction attempts. Each transaction carries a data packet. An interrupt endpoint can request a maximum latency from 1ms to 255ms, 10ms to 255ms for low-speed devices. For example, if the latency is 10ms, the host may initiate a new transaction anywhere from 1ms to 10ms after the last transaction attempt.

Which transfer type the host uses depends on the type of report requested and, sometimes, on the device hardware and version of Windows. A HID can exchange three types of reports: input, output, and feature. Input and output reports are best for data that must be sent or retrieved periodically, such as keypresses. Feature reports are best for data that transfers occasionally or isn't time-critical—such as setup or configuration information.

When the host requests input reports, the device sends data to the host in interrupt transfers. The host

why would you want to network a thermostat?

= 72 degrees • power consumption = 4 kilowatts per hour • regulate = auto • high set point temp = 72 degrees • low set point temp
rop = 4.2 degrees • fan = low • energy discount = 18% • light = on • switch = off • filter efficiency = low • current mode = heat



We don't make thermostats. We provide networking software that allows thermostats and many other everyday products to communicate, making them smarter. For example, our software enables thermostats to be remotely managed. That means utility providers can eliminate brownouts and reduce the need to build new power plants. And for customers it means less power consumption, which results in lower utility bills. So maybe the question should be, why wouldn't you network a thermostat? To find out how emWare can help make your products better, visit us at www.emware.com/thermostat3

to conserve energy.

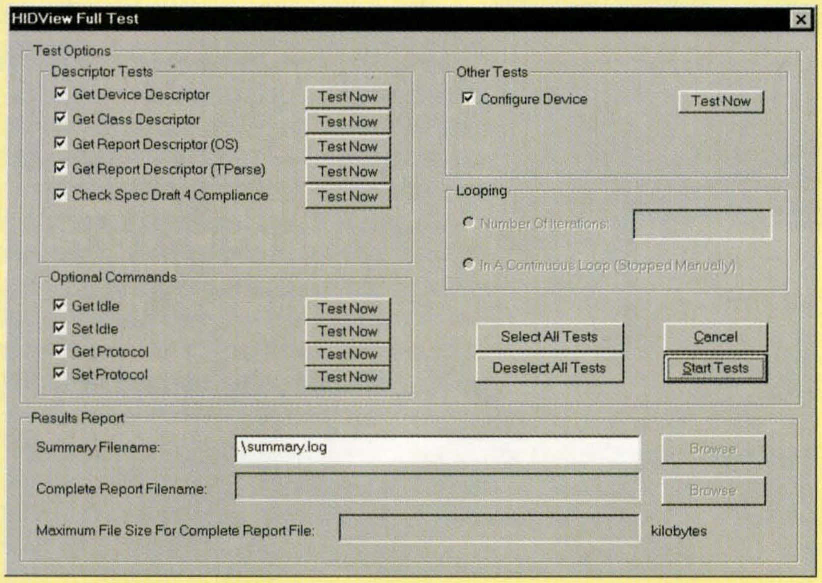
emWare® ANY DEVICE, ANY NETWORK™ ALLIANCE™ e-smart™

device networking software that makes products better

go to www.emware.com/thermostat3 or call toll-free 1.877.436.9273

A HID can send and receive reports using a device's control endpoint and interrupt endpoint(s).

FIGURE 2 The Compliance Tool performs generic device tests as well as tests specific to HIDs. It's available free from the USB Implementors Forum



HIDView Full Test

Test Options

Descriptor Tests
- [✓] Get Device Descriptor — Test Now
- [✓] Get Class Descriptor — Test Now
- [✓] Get Report Descriptor (OS) — Test Now
- [✓] Get Report Descriptor (TParse) — Test Now
- [✓] Check Spec Draft 4 Compliance — Test Now

Other Tests
- [✓] Configure Device — Test Now

Looping
- ○ Number Of Iterations:
- ○ In A Continuous Loop (Stopped Manually)

Optional Commands
- [✓] Get Idle — Test Now
- [✓] Set Idle — Test Now
- [✓] Get Protocol — Test Now
- [✓] Set Protocol — Test Now

Select All Tests | Cancel
Deselect All Tests | Start Tests

Results Report

Summary Filename: .\summary.log — Browse

Complete Report Filename: — Browse

Maximum File Size For Complete Report File: kilobytes

schedules the transfer requests according to the maximum latency requested in the endpoint descriptor.

For an output report, the host sends data to a device using control or interrupt transfers. The ability to do interrupt OUT HID transfers was added in version 1.1 of the HID spec, and is available under Windows 98 SE and later Windows editions, including Windows 2000.

If the HID interface has an interrupt OUT endpoint, Windows 98 SE or later will use it for output reports, and Windows 98 Gold (original) will use control transfers. Device firmware would need to support both transfer types to enable a device to receive output reports either way. If the interface doesn't have an interrupt OUT endpoint, the HID driver uses control transfers for output reports.

Feature reports can travel in either direction; they always use control transfers. To send an OUT feature report, the host sends a Set_Report request, followed by the report data, and the device returns status information to indicate the success or failure of the transfer. To receive an IN feature report, the host sends a Get_Report request, the device sends the report data, and the host returns status information to indicate the success or failure of the transfer.

If data is needed only occasionally and timing isn't critical, feature reports keep the bus from clogging with periodic transfer attempts when data isn't needed.

Another advantage of feature reports is the ability to support multiple report formats. The host can specify a report number in the control request. In contrast, in an interrupt transfer, the host just requests or sends data bytes without having the chance to name a specific report.

Report formats can be simple or complicated. Listing 2 shows a descriptor for a very basic report for a device that sends and receives two bytes.

Items in the report descriptor can specify how the data should be used (for example, which axis or button on a mouse), what units to apply, the range of valid values, and more. The items and values to use for common device types are defined in the HID specification and the HID Usage Tables document available from *www.usb.org*. A device that doesn't fit a standard usage can use vendor-defined items.

Tools

To assist in developing HID firmware, the Implementers Forum provides two free tools: the Descriptor Tool and the USB Compliance Tool.

The Descriptor Tool (Figure 1) automates the process of building report descriptors and enables you to check the format before copying the descriptor into the device. You can build a descriptor by selecting items from a pick list and entering values (such as the number of bytes in the item) as needed. You can then ask the tool to check the descriptor and flag errors. To get you started, the tool comes with example descriptors for common device types.

The USB Compliance Tool is actually a suite of tools that performs a series of basic tests on any device, plus additional tests for HIDs, hubs, and communications-class devices.

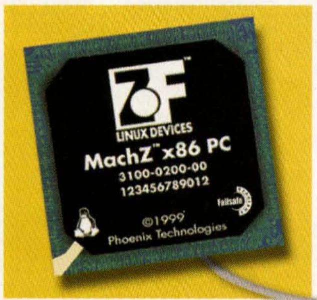
The Compliance Tool loads its own diagnostic driver for the device under test. You can read and check descriptors and send other control requests. The HID tests (Figure 2) enable you to check the report descriptor and perform additional class-specific tests.

Windows communications

When you have the HID firmware up and running, you're ready to start accessing the device from applications on the host computer. (This article concentrates on Windows, but HID support is also available or in progress for other operating systems.)

Finally

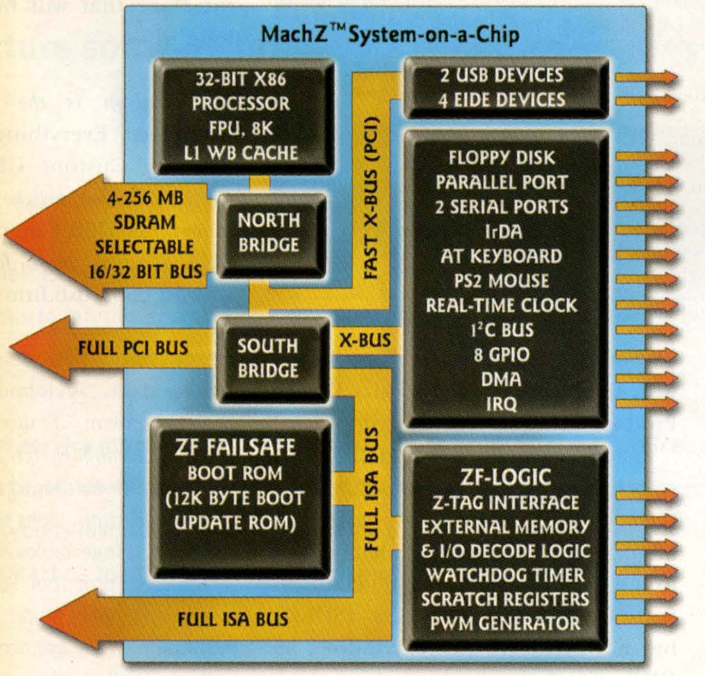
a *real* x86 System on a Chip



(actual size)

Introducing the "FailSafe MachZ" for the next generation of embedded systems

You can design the MachZ system into virtually anywhere your imagination can take you. It's very small and at a 1/2 watt it actually runs cool to the touch. Think about it: you could embed this little power house in a system that is completely sealed. Consider the possibilities, consider the facts. Then put your engineering brain to work.



Ultra Low Power <500mW@100MHZ

Auto-Boot FailSafe System allows crash-immune Internet upgrades - FailSafe Boot, Flashless Operation, Dual WD-Timer, Z-tag 2M-bit/s Flash download, Scratch Registers

Fully PC Compatible runs all x86 code natively

Easily Interface HomePNA, Ethernet, Modems, Graphics Controllers, PC/104, MPEG, XDSL, ISDN, IrDA, Flash, CAN, GPS

Includes BIOS and Linux O/S or WindRiver® VxWorks RTOS

phoenix™ technologies

Oh, and did we mention that it runs cold?

Failsafe EMBEDDED

388 BGA Runs at a cool 1/2 Watt not a typical 8 to 10 Watts

ZF LINUX™ DEVICES

800.683.5943 • www.zflinux.com

When you have the HID firmware up and running, you're ready to start accessing the device from applications on the host computer.

TABLE 1 Windows' HID API functions

API function	Documentation	Use in HID communication
HidD_GetHidGuid	Windows 98/2000 DDK	Returns the GUID associated with HIDs
SetupDiGetClassDevs	Windows NT/2000 DDK and NT/2000 Platform SDK	Returns an array of structures containing information about all installed HIDs
SetupDiEnumDeviceInterfaces	Windows NT/2000 DDK and NT/2000 Platform SDK	Returns a pointer to a structure that identifies an interface in the array returned by SetupDiGetClassDevs
SetupDiGetDeviceInterfaceDetail	Windows NT/2000 DDK and NT/2000 Platform SDK	Returns a device pathname for a specified device interface
CreateFile	Windows Platform SDK	Opens a handle to a HID using the pathname returned by SetupDiGetDevice-InterfaceDetail
HidD_GetAttributes	Windows 98/2000 DDK	Returns the Vendor ID, Product ID, and Version for a specified HID
HidD_GetPreparsedData	Windows 98/2000 DDK	Returns a handle to a buffer with information about a device's capabilities
HidD_GetCaps	Windows 98/2000 DDK	Returns a structure describing a device's capabilities
HidD_GetValueCaps	Windows 98/2000 DDK	Returns a structure describing the values in a device port
HidD_GetButtonCaps	Windows 98/2000 DDK	Returns a structure describing buttons in a device report
HidD_GetFeature	Windows 98/2000 DDK	Reads a feature report from a specified HID
HidD_SetFeature	Windows 98/2000 DDK	Sends a feature report to a specified HID
ReadFile	Windows Platform SDK	Reads in input report from a specified HID
WriteFile	Windows Platform SDK	Sends out output report to a specified HID
HidP_SetButtons	Windows 98/2000 DDK	Sets the button data in a report

The Windows 98 and 2000 driver development kits (DDKs) include a fairly complete tutorial on user-mode (application-level) HID communications. The examples use C syntax, but with some translating you can use Visual BASIC, Delphi, or any other language that can call API functions. Table 1 lists the major API functions used in HID communications.

The first step in talking to any HID is finding it. This requires a series of API calls that steps through all attached HIDs, looking for one that matches your criteria. You can look for a standard device type like a joystick or keyboard, or you can look for a match with the Vendor and Product IDs or Product String in your descriptors.

The API functions that identify and return a device pathname for each HID are Windows Device Management Functions documented in the Windows NT and 2000 Platform Software Development Kits (SDKs), but not included in the Windows 98 DDK.

When you have a pathname, you can use CreateFile() to open a handle to the device, then use HID-specific API functions to read the Vendor and Product IDs, Product String, and other information about the device's intended use and abilities.

Once you've identified the device you're looking for, you're ready to exchange reports using ReadFile() and WriteFile() (for input and output reports) or HidD_Get_Feature() and HidD_Set_Feature() (for feature reports).

One caution about input reports: ReadFile() is a blocking system call that won't return until the device has returned the expected amount of report data. Using ReadFileEx() or overlapped I/O doesn't help. So it's best to place code that calls ReadFiles() in its own program thread, so it doesn't hang the main application while waiting for data.

Between the support provided by Windows and other operating systems and the wide availability of USB ports, HIDs are worth a look. A HID is likely to be the quickest route to a finished PC peripheral, and you can be confident that you've chosen an interface that will be around for a long while. **esp**

Jan Axelson is the author of USB Complete: Everything You Need to Develop Custom USB Peripherals. *Portions of this article are adapted from material in* USB Complete. *Jan hosts a USB Central page for developers at* www.lvr.com/usb.htm.

Resources

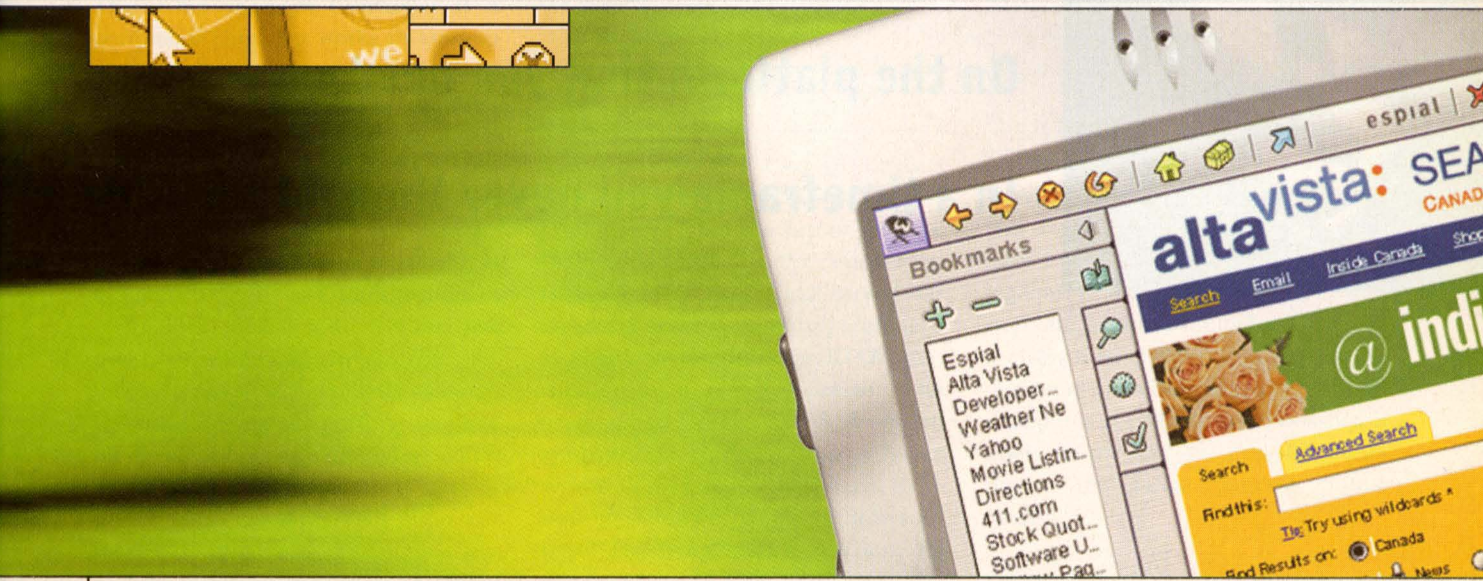
Ganssle, Jack. "An Introduction to USB Development," *Embedded Systems Programming*, March 2000, p. 79.

HID Page (*www.usb.org/developers/hidpage.html*). Links to the HID specification, Usage Tables documents, and the Descriptor Tool. Hosted by the USB Implementers Forum.

USBComp (*www.usb.org/developers/tools.html*). A free suite of tools for compliance testing.

DirectX Developers Center (*msdn.microsoft.com/directx/*). The DirectX DDK, newsgroups, articles, and more.

ESPIAL®



infrastucture software for the pervasive internet ...say again?

Our vision at Espial® is to move technology beyond the PC by harnessing the full potential of the internet. Access to information anytime, anywhere, for anyone – the industry is buzzing about it, but who can actually offer it?

Well, between our **service management software**, our **super-small internet applications** and our **development tools**, we meet the needs of service providers, OEMs and developers – in other words, everybody involved with getting **service-based smart devices** into the hands of consumers. Contact us to learn more about our complete software solution.

BUILD A SMART FUTURE™

espial.com

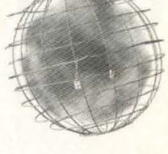
Capture the technologies that you want.

On the platform that you need.

In a timeframe you never thought possible.

Design the product of your dreams.

When you start with phoenix™ technologies anything is possible.



Embedded Internet Tools

Chip for wireless

The iChip Internet Controller is a peripheral chip that works in tandem with a device's host processor. It enables engineers to make Internet connectivity a feature of their products without designing in additional processors and memory to run the necessary protocols. It includes 512KB of onboard flash memory for storing and updating TCP/IP and other protocols. A few lines of code written in the host processor will invoke the Internet protocols stored in iChip's flash memory. In addition to supporting wireless modems, iChip now includes HTTP client software; binary MIME attachments; auto baud-rate detection for modem configuration; stay-online feature for sending and receiving multiple messages within one Internet session; and hardware flow control that frees the host CPU from managing communications. iChip is available now for $20 each in quantities of 10,000.

Connect One
Santa Clara, CA
(408) 986-9602
www.connectone.com

Processor

The BlueBird series of digital voice and digital music processors are designed to enable audio user interface capabilities in mobile devices. The BlueBirdV and BlueBirdVL allow MP3 devices to utilize voice record with background noise cancellation. They also enable PDAs to act like low-power audio devices and record or playback voice attachments to e-mail. On notebook computers, the chips can interface with CD-ROM drives to play back MP3 or audio CDs during sleep mode. The BlueBirdV is sampling now for $19 per unit in quantities of 10,000.

Silicon Motion
San Jose, CA
(408) 501-5300
www.siliconmotion.com

Linux driver for PCI boards

This Linux driver works with all PCI DAP boards from the top of the line DAP 5200a/526 to the entry-level DAP 840/103. Every DAP model has onboard intelligence implemented as DAPL, a multitasking RTOS that runs on an onboard processor. The user controls the RTOS from an application running under a Linux system that contains the board. Real-time functions in DAPL include software triggering, data reduction, filtering, FFT, and PID. The driver supports Linux kernel 2.2. You can download a free copy at the company's Web site.

Microstar Laboratories
Bellevue, WA
(425) 453-2345
www.mstarlabs.com

Wireless software

SMP-Softwire is a digital signal processor-pool technology that enables wireless equipment manufacturers to generate products that combine networked functions such as voice, data, video, and fax services. It runs on DSP devices and is IP-ready. Among the protocols SMP-Softwire features are GSM 3.45 and 3G TS 23.146 fax adaption protocols; customized cellular V.90 modem pools for high speed circuit switched data (HSCSD) and enhanced data rate EDGE/EHSCSD channels; and GSM AMR and FR voice coding. SMP-SoftWire is available now.

Surf Communication Solutions
Maynard, MA
(978) 897-4005
www.surf-com.com

"Swiss army knife" for Linux

BusyBox is a Linux development tool that combines tiny versions of common UNIX utilities into a single executable. It was designed for use with size- and resource-limited projects. Its architecture is modular, allowing designers to include or exclude specific features at compile time. Busybox is open source and freely distributable under the GNU general public license.

Lineo
Lindon, UT
(801) 426-5001
opensource.lineo.com

Software and RTOS package

The Advanced High Availability for Linux (HA-Linux) software is now being packaged with Red Hat Linux. The software tools and application enablers supported on Red Hat Linux are now available to telecom OEMs building wireless, wireline, and Internet infrastructure applications. The HA-Linux offering, together with Red Hat Linux, will be available on the CPX8000 line of computer platforms. Features of HA-Linux software include host CPU multi-stage switch over, hot-swap of all components, and network management. HA-Linux with Red Hat Linux 6.2 is available now. The package ranges from $449 to $748.

Motorola
Austin, TX
(512) 895-6217
www.motorola.com

• Best Practices • Software Development • Configuration Management • System Definition • System Testing • Best Practices •

Software Development • Configuration Management • System Definition • System Testing • Best Practices • Software Development

System Testing • System Definition • Configuration Management • Software Development • Best Practices • System Testing • System Definition • Configuration Management

Rational ClearCase. Change. Automated.

Simplify change management and make time for the things that really matter. When you spend all day keeping track of changes, you lose time for the important things. Like writing code and updating Web content. Rational ClearCase® can help by automating many of the time consuming tasks associated with developing software and building e-business applications, such as parallel development and change management. It even manages code and content so you're always sure the correct version of your Web site is posted. Plus, it integrates with other Rational analysis, design, development and testing tools to provide the industry's only complete development lifecycle solution. No wonder IDC ranked Rational ClearCase 1st in Software Configuration Management Market Leadership* for the 4th year in a row. Visit **www.rational.com/change3 for your free IDC report.**

Rational®
the e-development company™

*Application Life-Cycle Management Forecast and Analysis 2000-2004

Dan Saks

Character and String Literals

C and C++ provide several different kinds of literals: integer literals such as `10` and `0x1C`, floating literals such as `1.0` and `6.022e+23`, character literals such as `'a'` and `'\x10'`, and string literals such as `"ouch!"` and `"\n"`. Last month, I examined the integer and floating literals.[1] This month, I'll look at the character and string literals.

C and C++ differ with respect to numeric (integer and floating) literals in that C offers a couple of features not available in C++. In particular:

- Integer literals in C can have `long long` integer types. C++ does not have `long long` integer types
- C permits floating literals in hexadecimal form. C++ does not

Aside from these added capabilities of C, numeric literals are the same in C++ as they are in C.

Not surprisingly, character and string literals are also very similar in C and C++. Neither language offers forms unavailable in the other language. However, C++ changed the behavior of character and string literals in some subtle but interesting ways.

Character literals

In C, character literals such as `'a'` and `'\x10'` have type `int`, not `char`. At the risk of introducing a minor incompatibility with C, C++ changed the type of character literals to `char` so that overloaded functions and operators can have the expected behavior, as explained by the following example.

The Standard C++ library provides a set of functions named `operator<<` as output operators, declared more or less as:

In C++, character and string literals function just as C programmers would expect them to. Even more so.

```
ostream &operator<<
  (ostream &, char);
ostream &operator<<
  (ostream &, int);
ostream &operator<<
  (ostream &, long int);
...
```

The Standard C++ library defines `ostream` as the type of an output stream. Each function has a first parameter and return value of type `ostream` passed by reference.

An expression such as:

```
cout << x;
```

translates into the call:

```
operator<<(cout, x);
```

If `x` is an `int`, then from among the functions declared just above, the best match for this call is:

```
ostream &operator<<
  (ostream &, int);
```

which writes a character sequence representing the numeric value of `x` to output stream `cout`. On the other hand, if `x` is a `char`, then the best match for the call is:

```
ostream &
operator<<(ostream &,char);
```

which writes a single character representing the value of `x` to `cout`.

If character literals in C++ had type `int` (as they do in C), then an expression such as:

```
cout << 'a';
```

would call:

```
ostream &operator<<
  (ostream &, int);
```

which displays its right-hand operand as a number, not as a character. On a machine that uses the ASCII character set, this would display `'a'` as `97`.

C++ changed the type of character literals to `char`, so that:

```
ostream &operator<<
   (ostream &, char);
```

would be the better match for the call. This assures that overloaded operators such as `<<` have the expected behavior.

Bjarne Stroustrup, the inventor of C++, observed that:

"Surprisingly, giving `'a'` type `char` in C++ doesn't cause compatibility problems. Except for the pathological example `sizeof('a')`, every construct that can be expressed in both C and C++ gives the same result."[2]

In C++, `sizeof('a')` yields the same result as `sizeof(char)`, which is always 1. In C, `sizeof('a')` yields the same result as `sizeof(int)`, which may be 1, but is much more likely to be 2 or 4.

String literals in C

As I explained a few months ago, arrays in C and C++ are not just pointers.[3] Although C and C++ have a standard conversion from "array of `T`" to "pointer to `T`" which often makes arrays appear to be pointers, arrays really are arrays nonetheless.

In C, a string literal such as `"abcd"` has type "array of `char`." Since `"abcd"` consists of five characters—`'a'`, `'b'`, `'c'`, `'d'`, and `'\0'` (the null character)—its precise type is "array of 5 `char`." In any event, almost every time you use a string literal in an expression, the compiler treats it as if it were a "pointer to `char`".

Why isn't the type of a string literal "array of `const char`"? After all, programs shouldn't be overwriting the values of their string literals, should they?

A C compiler for embedded systems may place string literals into ROM. In that case, a program that attempts to write over a string literal will be trying to write into ROM. Not good.

Even in systems that don't have programmable ROM, C programs that overwrite string literals produce dire results. A C compiler is allowed to "pool" string literals. That is, a compiler can treat two or more string literals with the same value as if they are all the same object and allocate just one copy. If one part of a program overwrites the value of a string literal residing in RAM, it may affect the value of the literal as seen elsewhere in the program.

For example, suppose your program contains a function:

```
void rotate_left(char *);
```

which rotates a null-terminated character array left by one character. For example:

```
char digits[] = "9867530";
...
rotate_left(digits);
```

leaves the value `"8675309"` in array `digits`.

Now suppose the program contains the call:

```
rotate_left("Hello");
```

which changes the value of the string literal `"Hello"` to `"elloH"`. If the compiler is pooling literals, then the call:

```
printf("Hello");
```

elsewhere in the program might print `"elloH`." This is almost certainly not the correct output.

It seems that the simplest way to prevent such mishaps would be for C to give string literals type "array of `const char`." Then, the compiler would reject a call such as:

```
rotate_left("Hello");
```

for attempting an invalid pointer conversion. The compiler would transform `"Hello"` from "array of `const char`" to "pointer to `const char`". Function `rotate_left` expects a parameter of type "pointer to (non-`const`) `char`," but there's no standard conversion from "pointer to `const char`" to "pointer to (non-`const`) `char`." The compiler will issue a diagnostic message at this point.

So, why doesn't C treat string literals as "array of `const char`"? Because C didn't acquire the `const` qualifier until the language was into its second decade. By then, too much code had been written that treated string literals as arrays of non-`const` `char`. The C standards committee decided that the standard should sanction existing code rather than force programmers to rewrite it.

Even though string literals have type "array of `char`" in C, a program that attempts to modify a string literal has undefined behavior. That is, attempting to modify a string literal is an error, but compilers are not obligated to detect it.

String literals in C++

For compatibility with C, string literals in C++ originally had type "array of `char`." This allowed C code to be recompiled as C++ without change, but it produced surprising behavior during function overload resolution.

As I explained last year, a C++ program can overload functions whose parameter types differ only in their use of the `const` qualifier.[4] For example, a program can provide two distinct functions named `f` declared as:

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

Because string literals are supposed to be non-modifiable, you might reasonably expect a call such as `f("abc")` to call `f(char const *)`. In the past, it didn't. Rather, it called `f(char *)` because `"abc"` had type "array of (non-`const`) `char`."

A few years ago, the standards committee opted to change C++ so that string literals have type "array of `const char`." According to the C++ standard, `f("abc")` should call `f(char const *)`. Many, but not all, C++ compilers have since adopted this change.

Changing the type of string literals to "array of `const char`" introduces a problem for existing C++ programs containing code such as:

```
void g(char *);
...
g("abc");
```

Here, `g` is not overloaded. There's one function `g`, and it accepts an argument of type "pointer to (non-`const`) `char`." Calling `g("abc")` is valid C, and used to be valid C++. However, now that C++ regards `"abc"` as an "array of `const char`," this call requires a seemingly invalid pointer conversion.

For compatibility with existing code, C++ will still convert a string literal to a "pointer to (non-`const`) `char`" as a special case. Even though C++ will not convert any old "array of `const char`" to "pointer to `char`," it will do so only for string literals. This is so that code such as the prior call to `g("abc")` will continue to compile. However, the conversion from string literal to "pointer to `char`" is deprecated, which means that it may disappear from a future C++ standard.

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

References

1. Saks, Dan, "Numeric Literals," *Embedded Systems Programming*, September 2000, p. 113.
2. Stroustrup, Bjarne. *The Design and Evolution of C++*.Reading, MA: Addison-Wesley, 1994.
3. Saks, Dan, "Is an Array Really Just a Pointer?" *Embedded Systems Programming*, June 2000, p. 15.
4. Saks, Dan, "Using const and volatile in Parameter Types," *Embedded Systems Programming*, September 1999, p. 77.

Your OEM Project... Our Common Goal.

SBC8360VEA

The latest Socket 370 technology from the Axiom Embedded Group. It supports Intel® Celeron up to 533 MHz or Pentium III up to 700MHz. It features C&T 69000 AGP supporting CRT/LCD displays with integrated 2MB SDRAM. Also featured are onboard 10/100 Base-T Ethernet and 32-bit Audio capability.

LOW POWER CPU!

SBC8355VEA

- Supports Intel® Tillamook Low Power CPU, Pentium MMX, and AMD K6-2
- C&T 69000 AGP with 2MB SDRAM
- ESS Solo-1 Audio
- Realtek 8139 10/100 Base-T Ethernet
- One Axiom SODIMM Bus, One PCI and One PC/104

SBC8243VE

- PQFP AMD Am486DX5-133 CPU
- Ali 1487/1489 Core Logic Controller
- C&T 65550 PCI supporting CRT/LCD displays
- Realtek 8139B/C 10/100 Base-T Ethernet

SBC8441VE

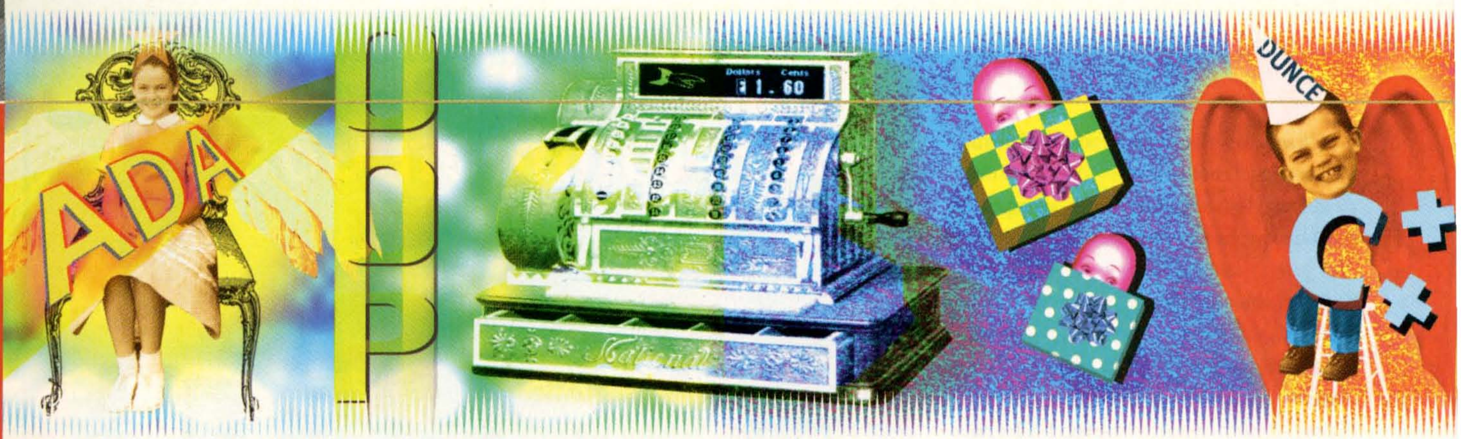
- Compact Size, fits in the space of 3.5" HDD
- Upgradeable with PC/104
- On board DX4-100 CPU, low cost for all embedded applications
- Built-in CRT/LCD Interface
- Realtek RTL8019 10 Base-T Ethernet

AXIOM™

888 - GO - AXIOM

www.axiomtek.com

CHAD BREMMON



From OOP to Nuts

Object-oriented programming techniques have been slow to become popular for embedded systems development. Maybe the problem isn't OOP, but C++. Here we see how Ada helps make OOP highly effective.

A key part of successful object-oriented programming is knowing when not to use it. The examples provided by Thomas Niemann in his article "Nuts to OOP!" (August 1999, p. 16) were not proof that object-oriented programming (OOP) is flawed. Rather, they were perfect examples of when not to use objects.

For those who may not have read the original piece, let me take you back to an article that suggested the increasing use of object-oriented programming by embedded developers is unwarranted. To support his position, the author described an example C++ program in which a `DEVICE` class provided the interface to an unnamed hardware device with two 16-bit registers. Each of these registers was abstracted with a template class, called `Register`, that allowed various operations (=, ^, &, |, <<, and >>) to be performed upon them. The `DEVICE` class did nothing but give these two registers names.

As Michael Barr wrote in his counterpoint to the original article: "Abstractions are only useful when they hide something." Mr. Niemann's example hid nothing of the underlying behavior and, in addition, took just as many lines of code to control the "abstracted" device as it would have taken to control the device directly. The real purpose of his `DEVICE` abstraction should have been to get away from reading and writing registers. It's the results of manipulating the registers that should be abstracted. In the process, the raw reads and writes will be hidden.

A proper example

For the last eight years, I have worked as an embedded software engineer, primarily developing software in Ada. Mr. Niemann says "Nuts to OOP!" but OOP isn't really his problem. What he ought to say is "Nuts to C++!" Ignoring the weaknesses and overcomplications of his example for a moment, most of the issues that he describes as flaws of OOP are really problems with the C++ language.

Before responding to Mr. Niemann's many criticisms of OOP, I'd like to show you a proper use of objects to abstract the functionality of a hardware device. We begin with the definition and implementation of a `Register` class in Ada95, which are shown together in Listing 1.

To create and use a register within your application code, you need only write:

```
My_Register : Register.Object;
Register.Create (My_Register,
  SomeAddress);--set the register
```

Over the years, I have found Ada code to be readable, reusable, and maintainable. Although these same code features can be achieved in other languages, Ada95 establishes a higher minimum standard in each of these areas.

```
   address
Register.Write (My_Register,
   Value);
```

Of course, the `Register` class doesn't really provide a very useful abstraction by itself. Any code that creates and uses a `Register` still has to know the actual address of the underlying register and the meaning of each value that can be written to or read from it. But the good thing is that Ada, unlike C++, allows the programmer to separate the OOP primitive of encapsulation from that of inheritance. So the overhead is at least minimized here.

In order to show the value of abstracting beyond this point, let's consider the simplest embedded program—what Mr. Barr described as the embedded version of "Hello, World" in the opening chapter of his book—one that flashes an LED. As you'll see, even this simple program can benefit from an object-oriented implementation.

To make a device like an LED more useful, we need to take the next step and hide the addresses of its registers and the specific values read from and written to them. Our goal is to separate the functionality of the device from the implementation details.

What is the function of an LED? It is a light, that is either on or off. As an application programmer, we would like to be able to turn it on and off, preferably without including any magic numbers (addresses or register values) in our code. Listing 2 shows the definition of a class that presents just such an abstract interface for an LED.

Of course, turning any LED on and off will probably involve a register. But no code outside the scope of the `Light` class need worry about such details. Nor should other code be affected if those details change on the next revision of the hardware. The resulting reduction in module coupling is a major advantage of OOP.

The end result of all this is an object-oriented program that is not overly complex, yet accomplishes the useful purpose of separating the function and use of the device from hardware details that may change in the future. One of the more interesting side effects is that we could easily extend this `Light` class, through inheritance, to support multi-color LEDs (say, red, green, and blue in a single package) or an eight-segment display.

If you look at Listing 3 you can see how the complexity is hidden within the implementation. Listing 4 shows how simple it becomes to use this class.

Benefits of Ada

Over the years, I have found Ada code to be readable, reusable, and maintainable. Although these same code features can be achieved in other languages, Ada95 establishes a higher minimum standard in each of these areas.

It's a fact that people maintain code and, for that reason, must read it many more times than it is written. Even without comments or careful coding, I think the previous examples show the readability inherent in the Ada95 language. Similarly, it should be clear that the `Light` class could be used in a variety of ways on a variety of hardware platforms. If platforms have a single memory-mapped register to control the LED, the only necessary changes are the address and values. By keeping the application software away from the direct manipulation of registers, we are able to ensure that the client code can be reused and only the implementation of our class may have to change.

The object-oriented principles of encapsulation, abstraction, modularity, and hierarchy lay the foundation for good programming. Ada95 takes advantage of these principles and enforces them:

- *Encapsulation.* Encapsulation is what keeps the application code from knowing or needing to know the implementation details of the class. This means that when you change an address or a way of doing things, you don't have to change every piece of source code that depends on the subprograms that are in your interface. Encapsulation in the above example is implemented by declaring the type `Light.Object` as private. No client can then access the values in that record directly
- *Abstraction.* When I woke up this morning, I tied my shoes. Long ago, I learned how to make my hands work together to tie my shoes. Before that, I learned how to make my hands work at all. The idea that I don't have to think about anything but tying my shoes is abstraction. In the above example, we can think of turning the LED on and off. We don't need to think of setting values in a register
- *Modularity.* Modularity is the ability to break a problem or its solution into smaller pieces. The classes in object-oriented programming languages allows us to do this. In Ada95, we specifically use packages and can go one step further and use child packages
- *Hierarchy.* Hierarchy is the idea of arranging elements of a problem into a treelike structure. For example, a digital display output may be built using several LEDs. In this case, you could combine them

BEN FISHMAN

LISTING 1 A Register class in Ada95

```
package Register is
   type Object is private;
   function Create (
         The_Address  :  in        System.Address )
     return Object;
   procedure Write (
         This         :  in out   Object;
         The_Value    :  in       Unsignedshort );
   procedure Read  (
         This         :  in out   Object;
         The_Value    :     out   Unsignedshort );
private
   type Object is
      record
         The_Location :  System.Address;
      end record;
end Register;


package body Register is

   function Create (
         The_Address  :  in       System.Address )
     return Object is
      This : Object;

   begin
      This.The_Location := The_Address;
      return This;
   end Create;

   procedure Write (
         This         :  in out   Object;
         The_Value    :  in       Unsignedshort ) is
      Register : Unsignedshort;
      for Register'Address use This.The_Location;

   begin
      Register := The_Value;
   end Write;

   procedure Read (
         This         :  in out   Object;
         The_Value    :     out   Unsignedshort ) is
      Register : Unsignedshort;
      for Register'Address use This.The_Location;

   begin
      The_Value := Register;
   end Read;

end Register;
```

together and pass characters to the digital display. This would in turn send the appropriate on/off commands to the LEDs on the display. These again could be combined to provide a multi-character display, to which you could pass a string as the argument. Hierarchy may also include inheritance, which involves adding functionality and state to a class and basing it on another class. For instance, we have an LED that can be turned on and off. We may want to manage an LED that can change colors. This would involve providing a method to change the color along with a value in the state to hold the color of the light. This colored light could simply inherit the rest of its behavior (on and off) from the `Light` class

Myths and facts

Finally, I would like to take an opportunity to discuss some of the OOP myths and facts presented by Mr. Niemann. First, what were called myths:

- *Objects are needed to protect data.* In order to truly protect data in C++, you must use the class mechanism. In Ada, however, the declaration of a private type allows for the protection of data without a class. Unfortunately, in C++ the encapsulation mechanism and the inheritance mechanism are one and the same. If you would like to declare data as private, you must put that data inside a class. In Ada 95, the encapsulation can take place regardless of whether or not you use inheritance. Encapsulation is implemented in Ada95 by declaring types in the private portion of a package. Inheritance is implemented using `tagged` types. These mechanisms may be used individually or together to provide an appropriate solution
- *Objects are needed to group data and procedures.* Again, if you are going to protect data and group it with pro-

ARC helps you keep pace with technology.

If your industry operates at a run, the ARC user configurable microprocessor core will let you sprint.

Proven in use, ARC is already bringing real products to market – fast. And in fast-moving product sectors such as mobile telecoms, networking and consumer electronics, it's speed to market which makes a product a winner.

ARC works quickly because it's simple to use, yet flexible.

Fully interoperable ARC software automatically generates the tools you need, so you can create your own application-specific processor without any extra specialist knowledge. You're in complete control of your design from start to finish – and the two have never been closer together.

Download a free demo of our ARChitect processor creation tool at www.arccores.com

And don't just keep up. Get ahead.

microprocessors by

ARC

Start thinking.

NORTH AMERICA: ARC Cores Inc, 6862 Santa Teresa Blvd, San Jose, CA 95119 Tel: 408 360 2120 Fax: 408 360 2139
EUROPE: ARC Cores, ARC House, Waterfront Business Park, Elstree Road, Elstree, Herts. WD6 3BS, UK Tel: +44 (0) 20 8236 2800 Fax: +44 (0) 20 8236 2801
www.arccores.com info@arccores.com ARC Cores is a trading name of ARC International (UK) Ltd.

LISTING 2 The definition of the Light class

```
with System;
with Opsys;
package Light is
    type Object is tagged private;
    function Create (
          Theaddress  : System.Address;
          Themask     : Opsys.Unsignedshort)
       return Object;
    procedure Turn_On (
       This : in out Object);
    procedure Turn_Off (
       This : in out Object);
private
    type Object is tagged
       record
          Address     : System.Address;
          Mask        : Opsys.Unsignedshort;
          Is_On       : Boolean := False;
        end record;
end Light;
```

cedures in C++ you must use a class. You can, of course, put some procedures together with some data that they manipulate, but the only way to ensure that the data manipulation is always happening within the proper code is to use a class. In Ada95, you have several options. Data and procedures can be grouped with or without using the inheritance mechanism

- *Objects are needed when implementing large programs.* The idea of implementing a very large system (in today's terms) without taking advantage of encapsulation, abstraction, hierarchy, and modularity is absurd. The advantages you gain from using object-oriented technology are strong enough to consider their use
- *Objects are easier to maintain.* I agree that if I were developing a small piece of code that had no other code depending on it, then making it a class would complicate matters. But if I have any code (other modules) that depends on the functionality I am producing, the entire effort is much easier to manage with object-oriented technology. When speaking of maintaining an individual class, yes the work is harder. But when it comes to maintaining an entire system, the work becomes easier
- *The switch statement is bad.* Here, I agree with Mr. Niemann, there is nothing wrong with C++'s switch statement (or the corresponding "case" statement in Ada95). However, I've never even heard of such an OOP myth before. In fact, it should be clear that when developing real-time systems, you can always know how long a switch or a case statement will take in the worst case. If you are using dynamic run-time dispatching through inheritance, it is much harder to determine what will happen ahead of time

According to Mr. Niemann, these are the facts about OOP:

- *Object-oriented programming requires more time, up front, doing design work.* We plan before we program. But it's not OOP that requires more planning. It's the whole idea of writing software correctly. Analysis and design are also supposed to be used in other programming paradigms, they just often aren't
- *Object-oriented programming requires more time to code a solution.* In describing why this is an issue, Mr. Niemann points out that it takes time to understand polymorphism and inheritance. Neither polymorphism nor inheritance are requirements for writing object-oriented programs. If a programmer designs an effective solution, the idea of polymorphism can be built in as appropriate. If you make every solution use inheritance and polymorphism it may become more complex. But why do that? I suggest using these features of the languages only when appropriate
- *Object-oriented programming results in a more complex solution.* When Mr. Niemann recommends this, I need to ask what is meant by a more complex solution. Harder to use? Harder to understand? I've been around procedural programs enough to know that it would certainly become a much simpler world when changes that I make do not impact the entire application. Coding in an object-oriented language, whether it be Ada95 or any other, limits the impact that changes made to one part of the code will have on the rest of the system
- *Object-oriented programming results in code that is more prone to error.* Mr. Niemann makes this claim based on the fact that he has proven object-oriented programming to be more complex with his example. If you are only talking about a 100-line program, it's more complex to develop using classes in an object-oriented language. But on a grand scale, object-oriented programming reduces complexity and localizes and contains errors

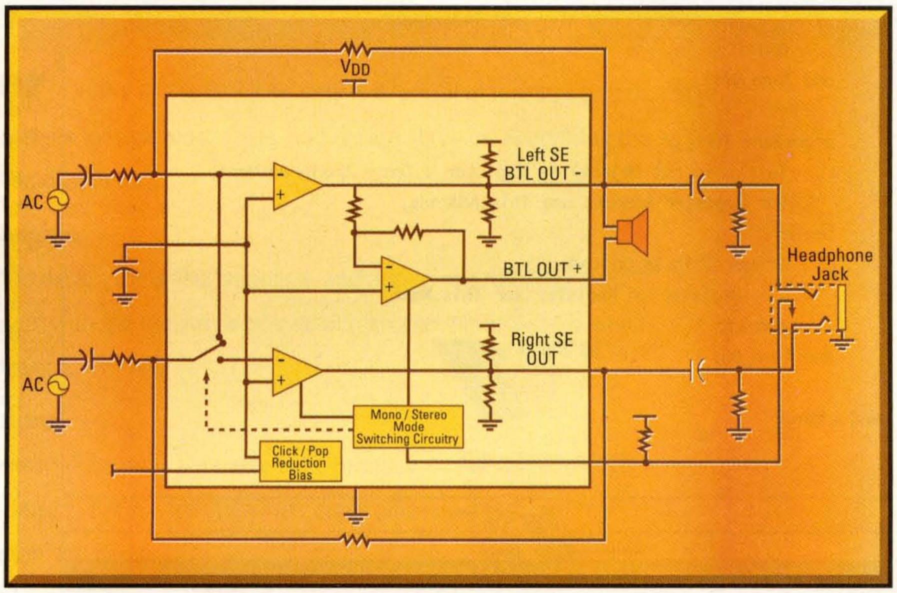
Now Available!
10-Pin MSOP, 1.5W Mono + 300 mW Stereo Boomer® Amplifier

National's LM4850 Available in 10-Pin MSOP and 14-Pin TSSOP Packages

- Mono 1.5W Differential at 4Ω or Stereo 300 mW Output at 8Ω
- Headphone Sense Pin Automatically Allows Use of Headphones @ 95 mW
- "Click and Pop" Suppression Circuitry
- No Bootstrap Capacitors Required
- Thermal Shutdown Protection
- Unity-gain Stable
- Low Shutdown Current

Ideal for Use in Portable Computers, Desktop Computers, PDAs, Handheld Games, and Cell Phones

For More Information on the LM4850:
www.national.com
1-800-272-9959
Free CD-ROM Data Catalog Available at:
freecd.national.com



V_{DD}
AC
AC
Left SE BTL OUT -
BTL OUT +
Right SE OUT
Headphone Jack
Mono / Stereo Mode Switching Circuitry
Click / Pop Reduction Bias

National Semiconductor

LISTING 3 The implementation of a Light class

```
with Opsys;
use type Opsys.Unsignedshort;

package body Light is

    function Create (
        Theaddress : System.Address;
        Themask : Opsys.Unsignedshort)
      return Object is This : Object;
    begin
        This.Address := Theaddress;
        This.Mask := Themask;
        This.Is_On := False;
        return This;
    end Create;

    procedure Turn_Off (
        This : in out Object) is Register : Opsys.Unsignedshort;
        for Register'Address use This.Address;
    begin
        if This.Is_On then
            Register := Register xor This.Mask;
        end if;
    end Turn_Off;

    procedure Turn_On (
        This : in out Object) is Register : Opsys.Unsignedshort;
        for Register'Address use This.Address;
    begin
        if not This.Is_On then
            Register := Register xor This.Mask;
        end if;
    end Turn_On;

end Light;
```

LISTING 4 An example of using a Light class

```
with Light;
with System;
procedure Lightdriver is

    My_Light : Light.Object;

begin
    My_Light := Light.Create (16#FF5E#, 16#40#);
    Light.Turn_On (My_Light);
    Light.Turn_Off (My_Light);
end Lightdriver;
```

- *Object-oriented programming results in increased maintenance costs.* Software is always hard to maintain. It is especially hard to understand and see the impacts of the code you are changing. At least in the object-oriented paradigm, you can reduce the coupling between modules, so that changes do not have such a dramatic impact on the entire application. Those who maintain legacy systems often take six months or more to come to a good understanding of even a small system. That's a high maintenance cost that is independent of the programming paradigm. If you have built your system using object-oriented principles, at least the maintainers will be able to see the scope and potential impact of the changes they need to make before they make them

C++ may be flawed and an object-oriented programming language. But does that mean we ought to abandon object-oriented programming altogether? Certainly not. There are many object-oriented programming languages to choose from. In particular, Ada95 does an excellent job of addressing the concerns of the embedded systems programming world, while maintaining an object-oriented capability. Let's not give up on object oriented technology simply because we can't easily encapsulate registers in C++. **esp**

Chad Bremmon works for Armstrong Consulting in the areas of object-oriented e-development and business process engineering. Chad was previously an Air Force Officer at the Pentagon for five years, after which he worked as a software engineering specialist for Rational Software. You can e-mail him at chadb@armstrongconsulting.com.

Resources

For more information about Ada, go to *www.acm.org/sigada* or *www.adapower.com*.

IP that clicks for everyone.

IP Center™ Xilinx IP Center is the only web portal that brings together cores, reference designs, design reuse tools, and design services. Everything the system designer could wish for.

SmartSearch™ engine

Get to the cores and design tools you want fast, and access our AllianceCORE™ and XPERTs Partners with the industry's best search engine. SmartSearch means easy access, and easy downloads.

VIRTEX-E

Complete solutions

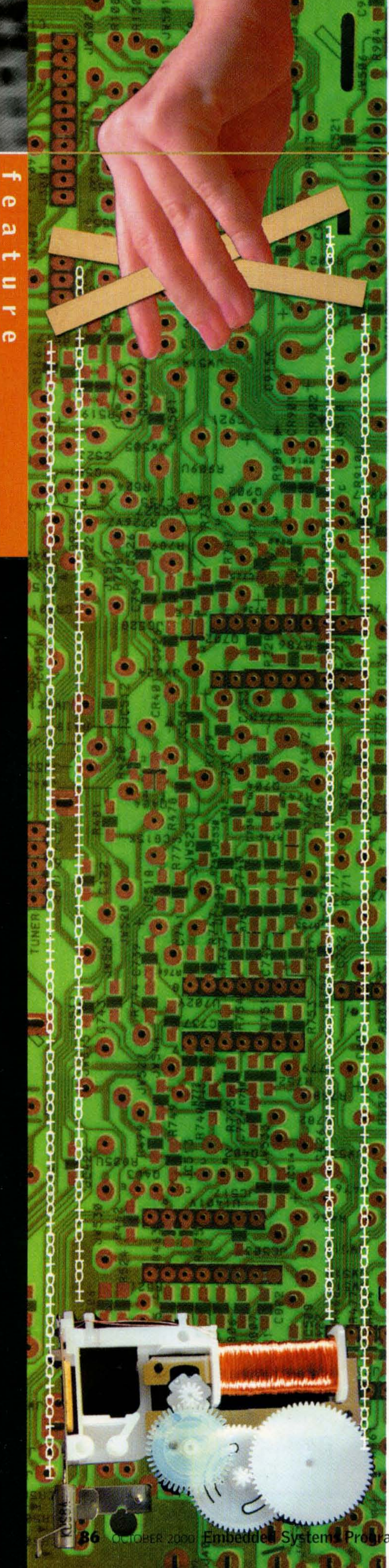
Supporting our powerhouse programmable logic products like Virtex™ and Spartan™ FPGAs, the Xilinx IP Center offers you the total solution you need.

SPARTAN-II FPGA

Try before you buy

At the Xilinx IP Center, you can try the most efficient IP solutions available today before you buy. It's just one more reason why Xilinx is the industry leader in IP. Visit ***www.xilinx.com/ipcenter*** today and get the IP that clicks for you.

XILINX®
The Programmable Logic Company℠
www.xilinx.com



TIM WESCOTT

PID Without a PhD

PID (proportional, integral, derivative) control is not as complicated as it sounds. Follow these simple implementation steps for quick results.

At work, I am one of three designated "servo guys," and the only one who implements control loops in software. As a result, I often have occasion to design digital control loops for various projects. I have found that while there certainly are control problems that require all the expertise I can bring to bear, a great number of control problems can be solved with simple controllers, without resorting to any control theory at all.

This article will tell you how to implement and tune a simple controller without getting into heavy mathematics and without requiring you to learn any control theory. The technique used to tune the controller is a tried and true method that can be applied to almost any control problem with success.

PID control

The PID controller has been in use for over a century in various forms. It has enjoyed popularity as a purely mechanical device, as a pneumatic device, and as an electronic device. The digital PID controller using a microprocessor has recently come into its own in industry. As you will see, it is a straightforward task to embed a PID controller into your code.

PID stands for "proportional, integral, derivative." These three terms describe the basic elements of a PID controller. Each of these elements performs a different task and has a different effect on the functioning of a system.

In a typical PID controller these elements are driven by a combination of the system command and the feedback signal from the object that is being controlled (usually referred to as the "plant"). Their outputs are added together to form the system output.

PID stands for "proportional, integral, derivative." These three terms describe the basic elements of a PID controller.

Figure 1 shows a block diagram of a basic PID controller. In this case the derivative element is being driven only from plant feedback. The plant feedback is subtracted from the command signal to generate an error. This error signal drives the proportional and integral elements. The resulting signals are added together and used to drive the plant. I haven't described what these elements do yet—we'll get to that later. I've included an alternate placement for the proportional element (dotted lines)—this can be a better location for the proportional element, depending on how you want the system to respond to commands.

Sample plants

In order to discuss this subject with any sense of reality we need some example systems. I'll use three example plants throughout this article, and show the effects of applying the various controllers to them:

- A motor driving a gear train
- A precision positioning system
- A thermal system

Each of these systems has different characteristics and each one requires a different control strategy to get the best performance.

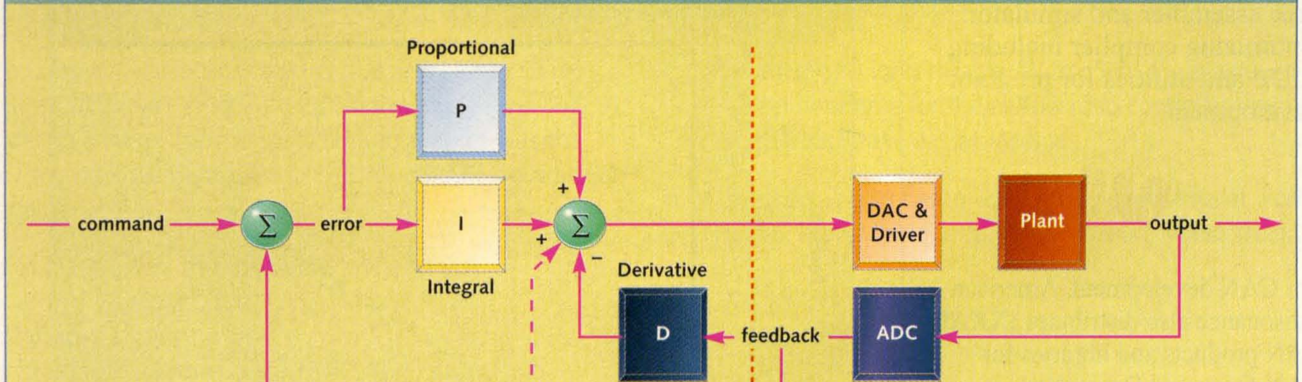
Motor and gear

The first example plant is a motor driving a gear train, with the output position of the gear train being monitored by a potentiometer or some other position reading device. You might see this kind of mechanism driving a carriage on a printer, or a throttle mechanism in an automobile cruise control system, or almost any other moderately precise position controller. Figure 2 shows a diagram of such a system. The motor is driven by a voltage that is commanded by software. The motor output is geared down to drive the actual mechanism. The position of this final drive is measured by the potentiometer.

A DC motor driven by a voltage wants to go at a constant speed that is proportional to the applied voltage. Usually the motor armature has some resistance that limits its ability to accelerate, so the motor will have some delay between the change in input voltage and the resulting change in speed. The gear train takes the movement of the motor and multiplies it by a constant. Finally, the potentiometer measures the position of the output shaft.

Figure 3 shows the step response of the motor and gear combination. I'm using a time constant value of $\tau_0 = 0.2$s. The step response of a system is just the behavior of the output in response to an input that goes from zero to some constant value at time $t = 0$. Since we're dealing with fairly generic examples here I've shown the step response as a fraction of full scale, so it goes to 1. Figure 3 shows the step input and the motor response. The

FIGURE 1 A basic PID controller



response of the motor starts out slowly due to the time constant, but once that is out of the way the motor position ramps at a constant velocity.

Precision actuator

It is sometimes necessary to control the position of something very precisely. A precise positioning system can be built using a freely moving mechanical stage, a speaker coil (a coil and magnet arrangement), and a non-contact position transducer.

You might expect to see this sort of mechanism stabilizing an element of an optical system, or locating some other piece of equipment or sensor. Figure 4 shows such a system. Software commands the current in the coil. This current sets up a magnetic field that exerts a force on the magnet. The magnet is attached to the stage, which moves with an acceleration proportional to the coil current. Finally, the stage position is monitored by a non-contact position transducer.

With this arrangement, the force on the magnet is independent of the stage motion. Fortunately this isolates the stage from external effects.

FIGURE 2 A voltage driven motor and gear train

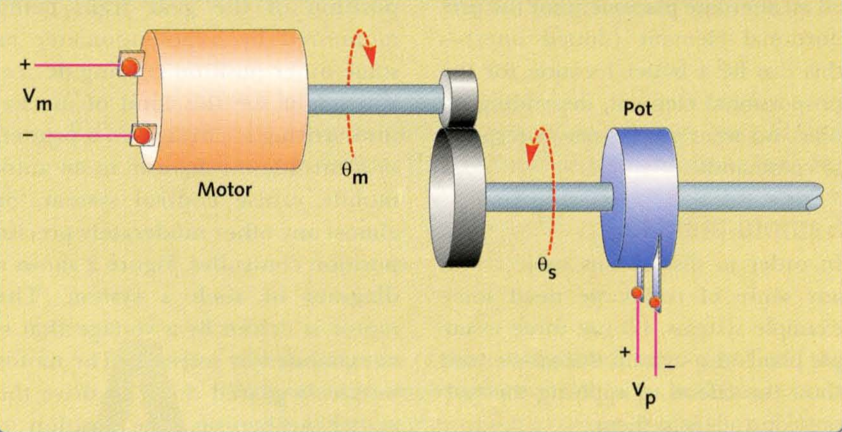
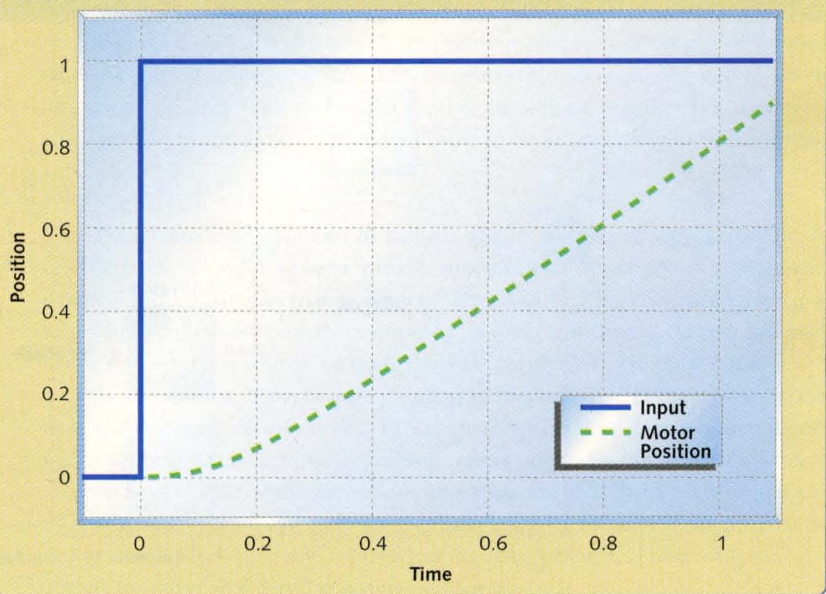
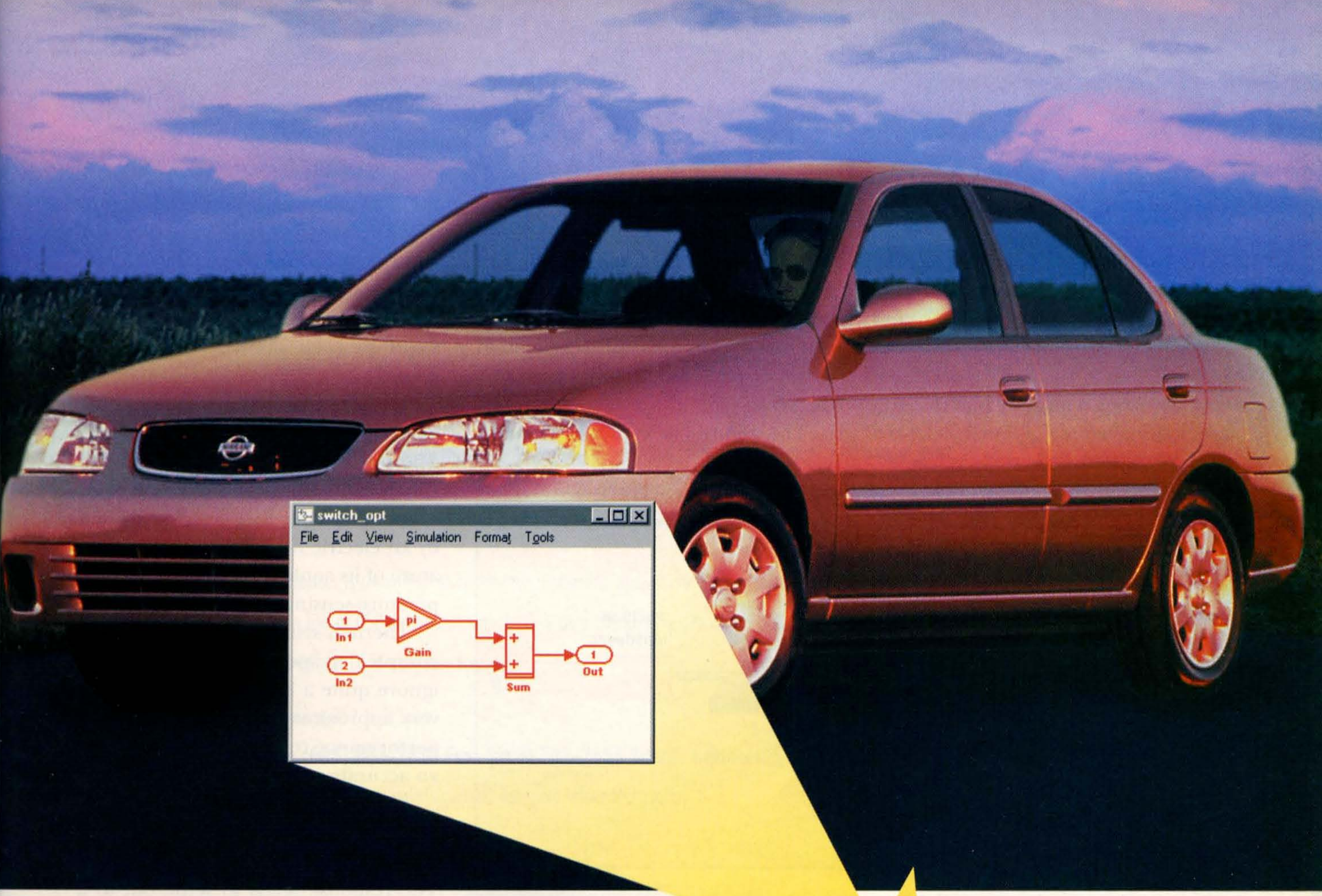
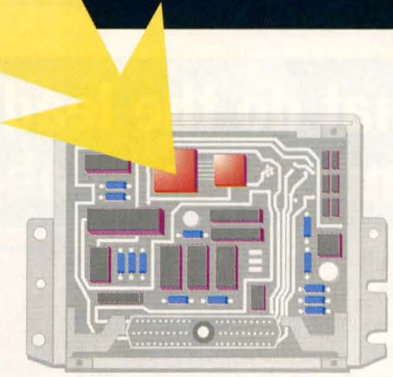


FIGURE 3 Motor and gear position vs. time







dSPACE Code Hits the Road

The world's lowest emission car runs with TargetLink code

Featuring a novel fuel mix control, the new Nissan Sentra CA fulfills California's strict Zero Emission Vehicle laws (P-ZEV). A decisive factor for the swift time to market was the development team's trust in TargetLink, dSPACE's production code generator.

TargetLink offers the following advantages:

- Highly efficient code
- Intuitive user interface
- Complete integration in MATLAB®/Simulink®

"TargetLink is highly reliable. With dSPACE we reduced our development time for the air/fuel ratio controller to 60%."
Shigeaki Kakizaki, Nissan Motor

TargetLink is also being used by DaimlerChrysler, Ford, Honda, TRW, Unisia Jecs, VDO, Visteon, Volvo Truck.

dSPACE Inc.
22260 Haggerty Road · Suite 120
Northville MI 48 167
Tel: (248) 344-0096 · Fax: (248) 344-2060
info@dspaceinc.com · www.dspaceinc.com

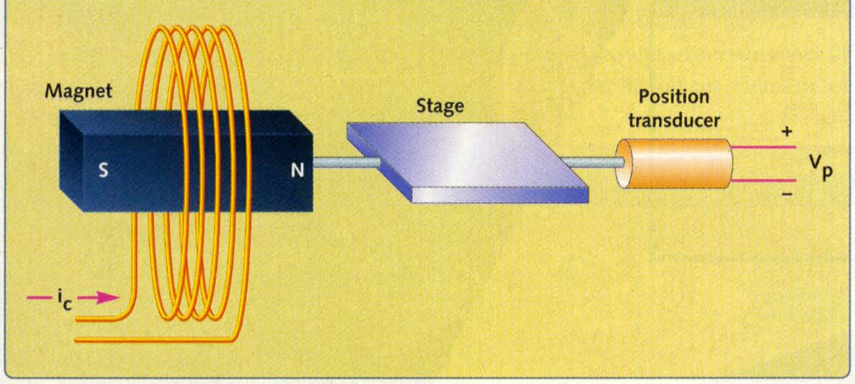
dSPACE

Unfortunately the resulting system is very "slippery," and can be a challenge to control. In addition, the electrical requirements to build a good current-output amplifier and non-contact transducer interface can be challenging. You can expect that if you are doing a project like this you are a member of a fairly talented team (or you're working on a short-lived project).

The equations of motion for this system are fairly simple. The force on the stage is proportional to the drive command alone, so the acceleration of the system is exactly proportional to the drive.

The step response of this system by itself is a parabola, as shown in Figure 5. As we will see later this makes the control problem more challenging because of the sluggishness with which the stage starts moving, and its enthusiasm to keep moving once it gets going.

FIGURE 4 A precision actuator



Temperature control

The third example plant I'll use is a heater. Figure 6 shows a diagram of an example system. The vessel is heated by an electric heater, and the temperature of its contents is sensed by a temperature-sensing device.

Thermal systems tend to have very complex responses. I'm going to ignore quite a bit of detail and give a very approximate model. Unless your performance requirements are severe, an accurate model isn't necessary.

Figure 7 shows the step response of the system to a change in V_d. I've used time constants of $\tau_1 = 0.1$s and $\tau_2 = 0.3$s. The response tends to settle out to a constant temperature for a given drive, but it can take a great deal of time doing it. Also, without lots of insulation, thermal systems tend to be very sensitive to outside effects. This effect is not shown in the figure, but we'll be investigating it later in the article.

Controllers

The elements of a PID controller presented here either take their input from the measured plant output or from the error signal, which is the difference between the plant output and the system command. I'm going to write the control code using floating point to keep implementation details out of the discussion. It's up to you to adapt this if you are going to implement your controller with integer or other fixed-point arithmetic.

I'm going to assume a function call as shown below. As the discussion evolves, you'll see how the data structure and the internals of the function shapes up.

```
double UpdatePID(SPid * pid,
  double error, double position)
```

What do the leading silicon vendors know about BIOS?

They know that the right BIOS is key to the success of embedded designs—and that configurability is key to the right BIOS.

That's why **AMD**, **Intel**, and **STMicro** ship General Software's Embedded BIOS pre-installed on their embedded platform evaluation boards.

With over 400 configuration options, Embedded BIOS offers the advanced configurability you need to run your custom target environment without editing the core BIOS source code.

Contact us today for detailed information and a free sample BIOS binary for your standard reference design.

Embedded BIOS™

ADAPTATION KIT:
Full source code automatically configured with over 400 parameters using BIOStart™ expert system

CORE BIOS FEATURES:
ROM/RAM/Flash disks, Setup system, console re-direction, manufacturing mode, WinCE loader, configurable PCI, integrated debugger, modular callouts to chipset, board, and CPU-level modules

CHIPSETS:
ALI—Aladdin V, Finali
AMD—186, SC300, SC400, SC520
INTEL—386EX, 430HX/TX, 440BX, 810, 840
NSC—Geode GXm, GXlv
STMicro—STPC family

IDEAL FOR:
Windows 95/98/CE/NT Embedded, DOS, Linux, and all x86-based operating systems

GENERAL™ SOFTWARE

www.gensw.com • sales@gensw.com • 800-850-5755 • 425-454-5755

VisualDSP – The Best DSP Development Environment For The Best DSPs Delivered.

You want to build the most advanced digital signal processing applications – that's why you are working with DSPs from Analog Devices. To get the most out of them, you need the best development environment – VisualDSP is that product!

With VisualDSP, you have a completely integrated, easy-to-use project management environment that will allow you to control and manage your projects efficiently from end-to-end, and make the most of the superior advantages only Analog Devices DSPs can offer.

Analog Devices is committed to continuous innovation and superior DSP performance via the White Mountain line of DSP development tools. Give us a call and we'll fetch the best development environment for DSPs available today.

Analog Devices DSP Tools Sales

Phone: (603) 883-2430 Fax: (603) 882-2655 Email: aditools@analog.com

Web: http://www.analog.com for more information today.

WHITE MOUNTAIN DSP™

The Design Center for Analog Devices DSP Tools

```
{
  .
  .
  .
}
```

The reason I pass the error to the PID update routine instead of passing the command is that sometimes you want to play tricks with the error. Leaving out the error calculation in the main code makes the application of the PID more universal. This function will get used like this:

```
.
.
position = ReadPlantADC();
drive = UpdatePID(&plantPID,
  plantCommand - position,
  position);
DrivePlantDAC(drive);
.
.
```

Proportional

Proportional control is the easiest feedback control to implement, and simple proportional control is probably the most common kind of control loop. A proportional controller is just the error signal multiplied by a constant and fed out to the drive. The proportional term gets calculated with the following code:

```
double pTerm;
.
.
.
pTerm = pid->pGain * error;
.
.
.
return pTerm;
```

Figure 8 shows what happens when you add proportional feedback to the motor and gear system. For small gains ($k_p = 1$) the motor goes to the correct target, but it does so quite slowly. Increasing the gain ($k_p = 2$) speeds up the response to a point. Beyond that point ($k_p = 5$, $k_p = 10$) the motor starts out faster, but it overshoots the target. In the end the system doesn't settle out any quicker than it would have with lower gain, but there is more overshoot. If we kept increasing the gain we would eventually reach a point where the system just oscillated around the target and never settled out—the system would be unstable.

The motor and gear start to overshoot with high gains because of the delay in the motor response. If you look back at Figure 2, you can see that the motor position doesn't start ramping up immediately. This delay, plus high feedback gain, is what causes the overshoot seen in Figure 8.

Figure 9 shows the response of the precision actuator with proportional feedback only. Proportional control alone obviously doesn't help this system. There is so much delay in the plant that no matter how low the gain is, the system will oscillate. As the gain is increased, the frequency of the output will increase but the system just won't settle.

Figure 10 shows what happens when you use pure proportional feed-

FIGURE 5 Precision actuator position vs. time

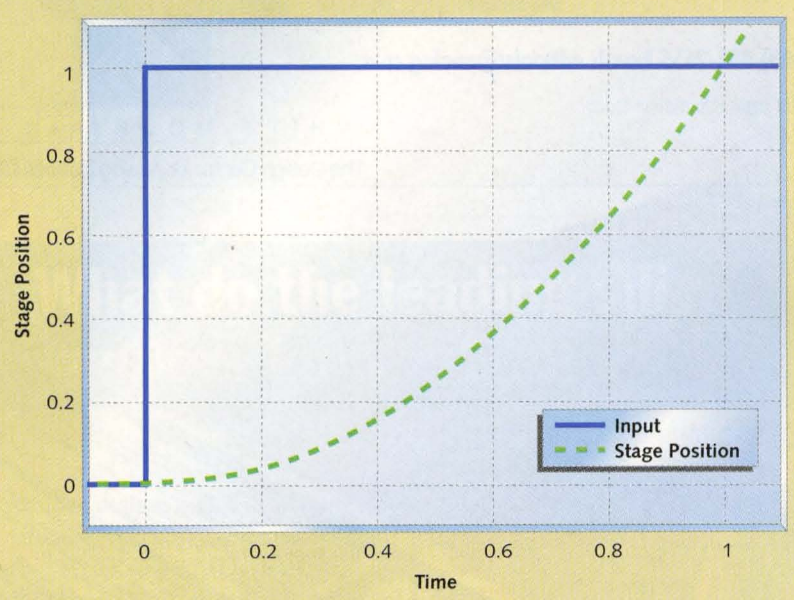
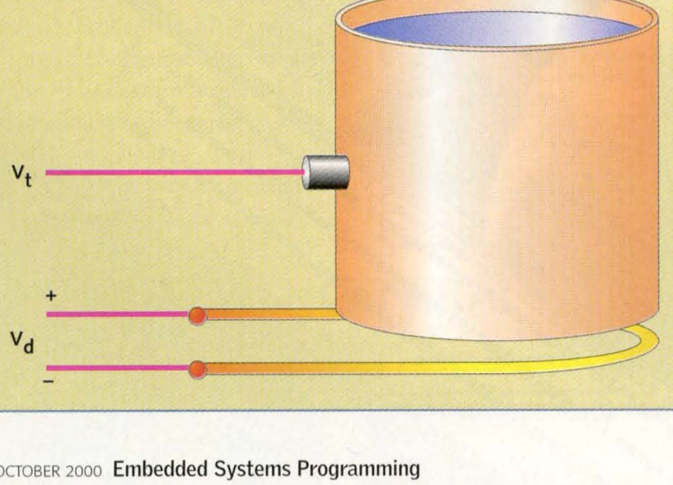


FIGURE 6 A heater



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

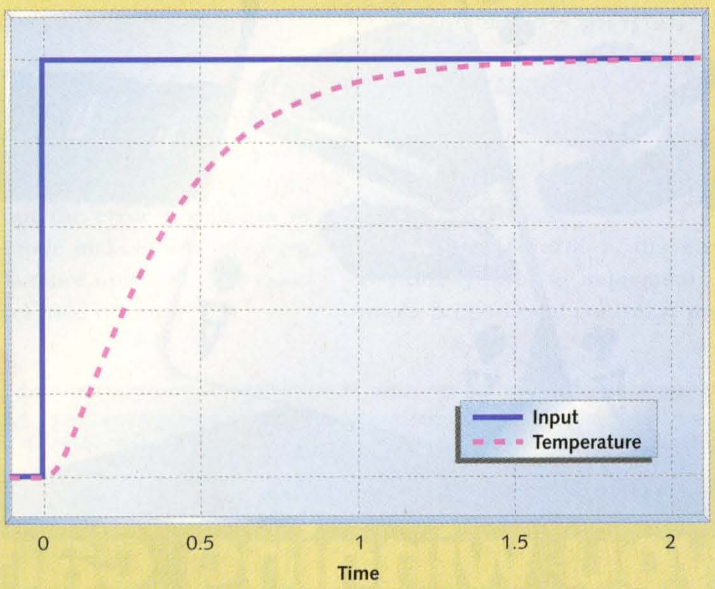
Advanced technology that brings results...

"The best emulator that I ever used!"

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: +90-312-4472700 PBX

FIGURE 7 Heater temperature vs. time



back with the temperature controller. I'm showing the system response with a disturbance due to a change in ambient temperature at $t = 2$s. Even without the disturbance you can see that proportional control doesn't get the temperature to the desired setting. Increasing the gain helps, but even with $k_p = 10$ the output is still below target, and you are starting to see a strong overshoot that continues to travel back and forth (this is called *ringing*).

As the previous examples show, a proportional controller alone can be useful for some things, but it doesn't always help. Plants that have too much delay, like the precision actuator, can't be stabilized with proportional control. Some plants, like the temperature controller, cannot be brought to the desired set point. Plants like the motor and gear combination may work, but they may need to be driven faster than is possible with proportional control alone. To solve these control problems you need to add integral or differential control or both.

Integral

Integral control is used to add long-term precision to a control loop. It is almost always used in conjunction with proportional control.

The code to implement an integrator is shown below. The integrator state, `iState` is the sum of all the preceding inputs. The parameters `iMin` and `iMax` are the minimum and maximum allowable integrator state values.

```
  double iTerm;
  .
  .
  .
// calculate the integral state
// with appropriate limiting
  pid->iState += error;
  if (pid->iState > pid->iMax)
    pid->iState = pid->iMax;
  else if (pid->iState < pid->
    iMin)
    pid->iState = pid->iMin;
  iTerm = pid->iGain * iState;
```

You always believed there were more intelligent embedded tools out there.

You were right.

People doubted their existence – yet you continued to search – and now you've found them.

COSMIC C compilers are fast, efficient, reliable, and produce the tightest object code available. Cosmic Software's embedded development tools offer portability for a complete line of microcontrollers. All toolkits include IDEA, our intuitive IDE that provides everything you need in a single, seamless Windows framework.

Add ZAP, our non-intrusive source-level debuggers and minimize your test cycle too. Want proof of their existence?

Download a free evaluation copy of our development tools at *www.cosmic-software.com* or call Cosmic today.

Cosmic supports the Motorola family of microcontrollers:
68HC05, 68HC08, 68HC11, 68HC12, 68HC16, 68300 and STMicroelectronics' ST7 Family.

COSMIC Software

E-mail: c-tools@cosmic-us.com
www.cosmic-software.com

Phone:	US	781 932-2556
	France	33 1 4399 5390
	UK	44 01256 843400
	Germany	49 0711 4204062
	Sweden	46 31704 3920

```
// calculate the integral term
.
.
.
```

Integral control by itself usually decreases stability, or destroys it altogether. Figure 11 shows the motor and gear with pure integral control (pGain = 0). The system doesn't settle. Like the precision actuator with proportional control, the motor and gear system with integral control alone will oscillate with bigger and bigger swings until something hits a limit. (Hopefully the limit isn't breakable.)

Figure 12 shows the temperature control system with pure integral control. This system takes a lot longer to settle out than the same plant with proportional control (see Figure 10), but notice that when it does settle out, it settles out to the target value—even with the disturbance added in. If your problem at hand doesn't require fast settling, this might be a workable system.

Figure 12 shows why we use an integral term. The integrator state "remembers" all that has gone on before, which is what allows the controller to cancel out any long term errors in the output. This same memory also contributes to instability—the controller is always responding too late, after the plant has gotten up speed. To stabilize the two previous systems, you need a little bit of their present value, which you get from a proportional term.

Figure 13 shows the motor and gear with proportional and integral (PI) control. Compare this with Figures 8 and 11. The position takes longer to settle out than the system with pure proportional control, but it will not settle to the wrong spot.

Figure 14 shows what happens when you use PI control on the heater system. The heater still settles out to the exact target temperature, as with pure integral control (see Figure 12), but with PI control, it settles out two to three times faster. This figure shows operation pretty close to the limit of the speed attainable using PI control with this plant.

Before we leave the discussion of integrators, there are two more things I need to point out. First, since you are adding up the error over time, the sampling time that you are running becomes important. Second, you need to pay attention to the range of your integrator to avoid windup.

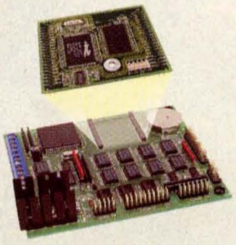
The rate that the integrator state changes is equal to the average error multiplied by the integrator gain multiplied by the sampling rate. Because the integrator tends to

A MORE POWERFUL ALTERNATIVE TO MICROCONTROLLERS

1.90" x 2.30" In Size

$29 qty 100
$39 qty one

Design your custom microprocessor-based board around the Rabbit 2000™ Core Module. The core module is based on the new Rabbit 2000 microprocessor, and can handle programs of up to 50,000 C statements. Benefit from easy and rapid development with this low-cost solution!



Make the Rabbit 2000™ Core Module the heart of your custom board design. A low-cost manufacturing license is also available.

RABBIT 2000™ CORE MODULE FEATURES

- 40 I/O pins
- 4 serial ports
- 7 timers
- Battery-backed time/date clock
- 128K-512K SRAM
- 256K flash
- Clock up to 25.8MHz
- Slave port
- Remote cold boot
- Excellent math processing

Development Kit only $169

Includes core module, prototyping board, complete Dynamic C® software development system (not a trial version), PC serial cable for real-time debugging, and documentation on CD-ROM

RABBIT Semiconductor

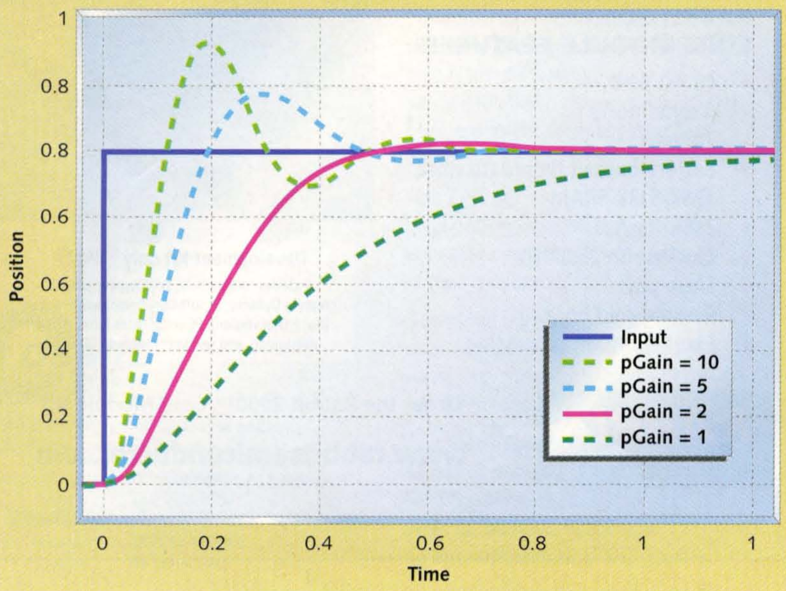
Order the Rabbit 2000™ Core Module Kit online at
www.rabbitsemiconductor.com
or call 530.757.8400

2932 Spafford Street, Davis, CA 95616 • Fax 530.757.8402

LISTING 1 PID controller code

```
typedef struct
{
  double dState;          // Last position input
  double iState;          // Integrator state
  double iMax, iMin;      // Maximum and minimum allowable integrator state

  double   iGain,         // integral gain
           pGain,         // proportional gain
           dGain;         // derivative gain
} SPid;

double UpdatePID(SPid * pid, double error, double position)
{
  double pTerm, dTerm, iTerm;

  pTerm = pid->pGain * error;    // calculate the proportional term

// calculate the integral state with appropriate limiting
  pid->iState += error;
  if (pid->iState > pid->iMax) pid->iState = pid->iMax;
  else if (pid->iState < pid->iMin) pid->iState = pid->iMin;

  iTerm = pid->iGain * iState;  // calculate the integral term

  dTerm = pid->dGain * (pid->dState - position);
  pid->dState = position;

  return pTerm + dTerm + iTerm;
}
```

FIGURE 8 Motor and gear with proportional feedback position vs. time

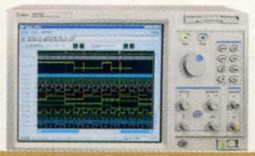


smooth things out over the long term you can get away with a somewhat uneven sampling rate, but it needs to average out to a constant value. At worst, your sampling rate should vary by no more than ±20% over any 10-sample interval. You can even get away with missing a few samples as long as your average sample rate stays within bounds. Nonetheless, for a PI controller I prefer to have a system where each sample falls within ±1% to ±5% of the correct sample time, and a long-term average rate that is right on the button.

If you have a controller that needs to push the plant hard, your controller output will spend significant amounts of time outside the bounds of what your drive can actually accept. This condition is called saturation. If you use a PI controller, then all the time spent in saturation can cause the integrator state to grow (wind up) to very large values. When the plant reaches the target, the integrator value is still very large, so the plant drives beyond the target while the integrator unwinds and the process reverses. This situation can get so bad that the system never settles out, but just slowly oscillates around the target position.

Figure 15 illustrates the effect of integrator windup. I used the motor/controller of Figure 13, and limited the motor drive to ±0.2. Not only is controller output much greater than the drive available to the motor, but the motor shows severe overshoot. The motor actually reaches its target at around five seconds, but it doesn't reverse direction until eight seconds, and doesn't settle out until 15 seconds have gone by.

The easiest and most direct way to deal with integrator windup is to limit the integrator state, as I showed in my previous code example. Figure 16 shows what happens when you take the system in Figure 15 and limit the integrator term to the available drive output. The controller output is still large (because of the proportional term), but the integrator doesn't wind



16702B Logic Analysis System

- Large touch screen flat panel display
- Knobs and dedicated hot keys
- Built-in 40X CD-ROM drive
- 100BaseT/10BaseT Ethernet
- Supports up to 400 MHz state speed/32M depth

We've given the learning curve a new direction. Our job is to make your job easier. That's why our latest logic analyzer employs a huge 12.1 inch touch screen that's easy to see and use, transforming your task into much less of one. It also utilizes hot keys and front panel knobs that allow you to instantly access frequently used windows, and break many functions down into just a couple steps. Add to that Eye Finder technology that automatically adjusts for the optimal set-up and hold conditions on high speed buses.

And modules from your 16500 can be used in the Agilent Technologies 16702B Logic Analyzer, which supports future generation modules as well. Meaning you'll be performing effortless logic analysis for years to come.

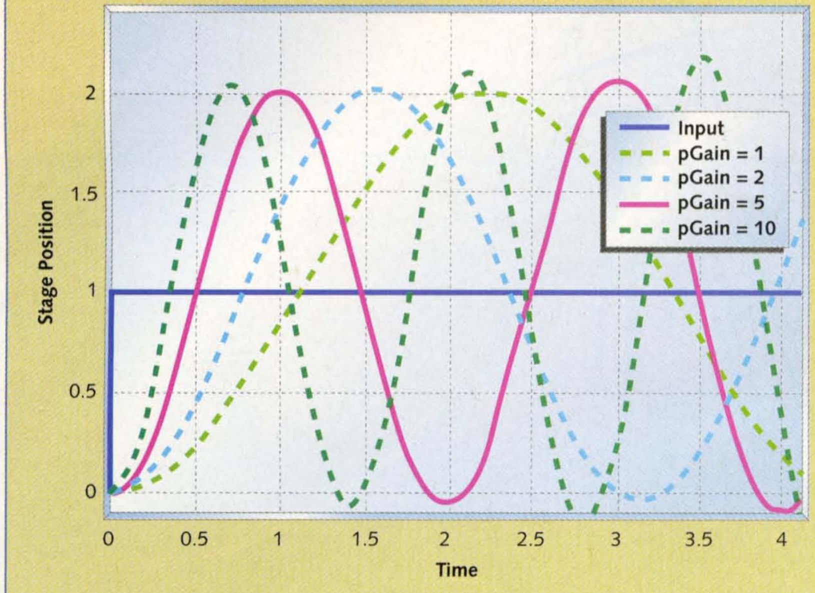
To find out more, contact us. We'll send you a free video that will show you just how simple it really is.

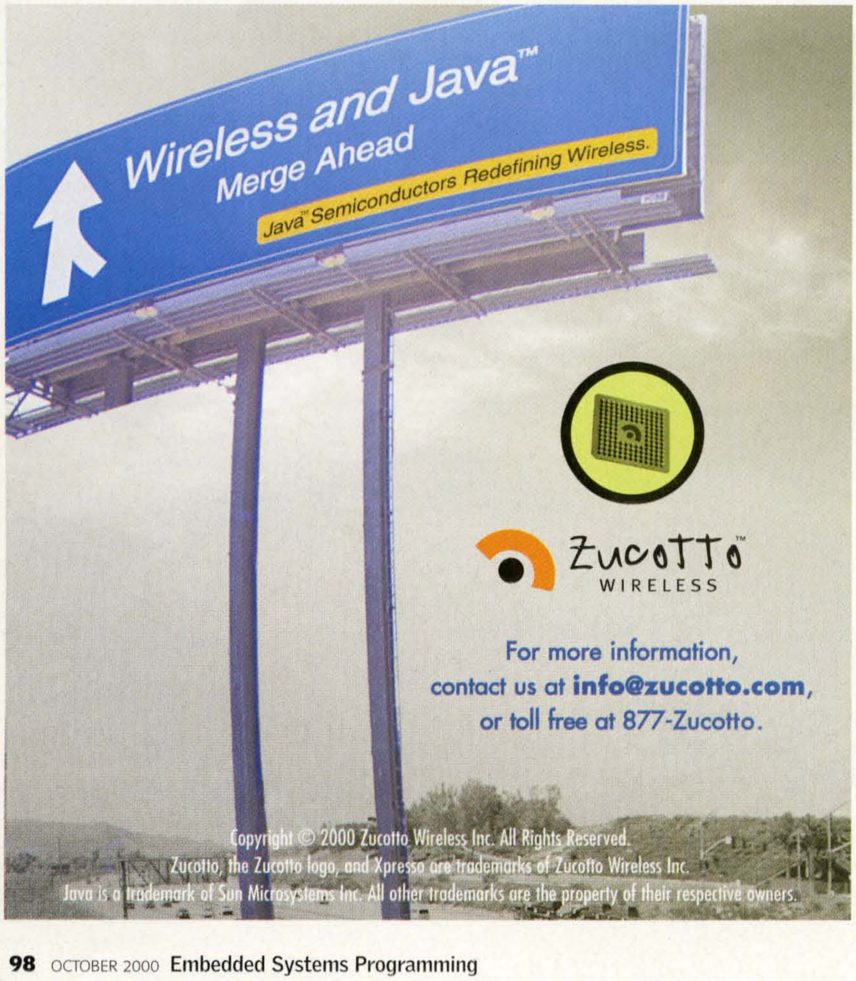
www.agilent.com/find/LAS-DSD

1-800-452-4844*, Ext. 7076

Agilent Technologies
Innovating the HP Way

FIGURE 9 Precision actuator with proportional feedback vs. time





For more information, contact us at **info@zucotto.com**, or toll free at 877-Zucotto.

up very far and the system starts settling out at five seconds, and finishes at around six seconds.

Note that with the code example above you must scale `iMin` and `iMax` whenever you change the integrator gain. Usually you can just set the integrator minimum and maximum so that the integrator output matches the drive minimum and maximum. If you know your disturbances will be small and you want quicker settling, you can limit the integrator further.

Differential

I didn't even show the precision actuator in the previous section. This is because the precision actuator cannot be stabilized with PI control. In general, if you can't stabilize a plant with proportional control, you can't stabilize it with PI control.

We know that proportional control deals with the present behavior of the plant, and that integral control deals with the past behavior of the plant. If we had some element that predicts the plant behavior then this might be used to stabilize the plant. A differentiator will do the trick.

The code below shows the differential term of a PID controller. I prefer to use the actual plant position rather than the error because this makes for smoother transitions when the command value changes. The differential term itself is the last value of the position minus the current value of the position. This gives you a rough estimate of the velocity (delta position/sample time), which predicts where the position will be in a while.

```
double dTerm;
.
.
.
dTerm = pid->dGain *
   (pid->dState - position);
pid->dState = position;
.
.
.
```

A 500 ps logic analyzer for $7,000?
Where's the glitch?

COMPUTING
TELECOM
VIDEO

TLA600 Logic Analyzer At $7,000,* you can't afford to miss this one. Here's powerful logic analysis — through a Windows interface — that provides 500 ps timing with up to 200 MHz state on every channel through the same probe. Look for comparable timing analysis elsewhere, and you could spend nearly four times as much. See this and believe. Get your free XYZ's of Logic Analyzers by calling 800-426-2200 x3059 or visit **www.tektronix.com/TLA**

Tektronix®

*TLA600 Series starts at $7,000 (manufacturer's suggested retail price).

With differential control you can stabilize the precision actuator system. Figure 17 shows the response of the precision actuator system with proportional and derivative (PD) control. This system settles in less than 1/2 of a second, compared to multiple seconds for the other systems.

Figure 18 shows the heating system with PID control. You can see the performance improvement to be had by using full PID control with this plant.

Differential control is very powerful, but it is also the most problematic of the control types presented here. The three problems that you are most likely going to experience are sampling irregularities, noise, and high frequency oscillations.

FIGURE 10 Temperature controller with proportional feedback. The kink at time = 2 is from the external disturbance

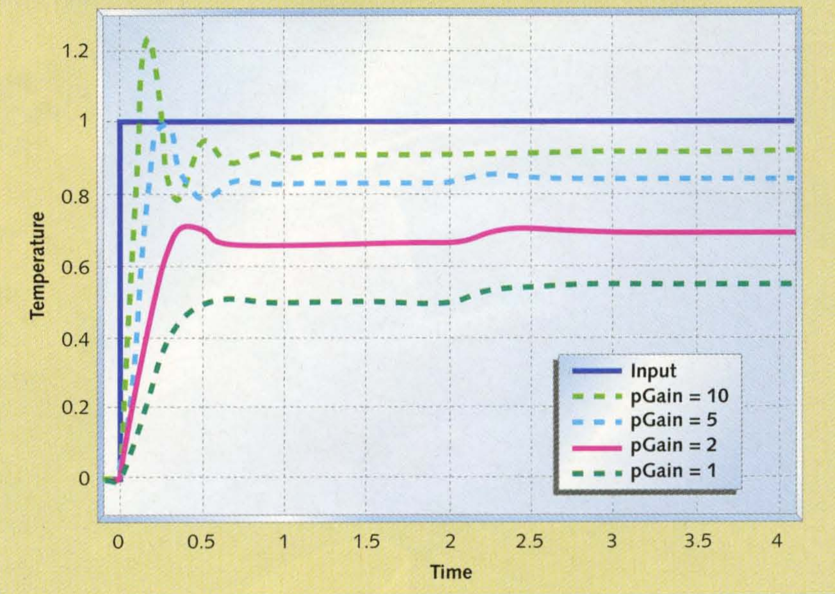
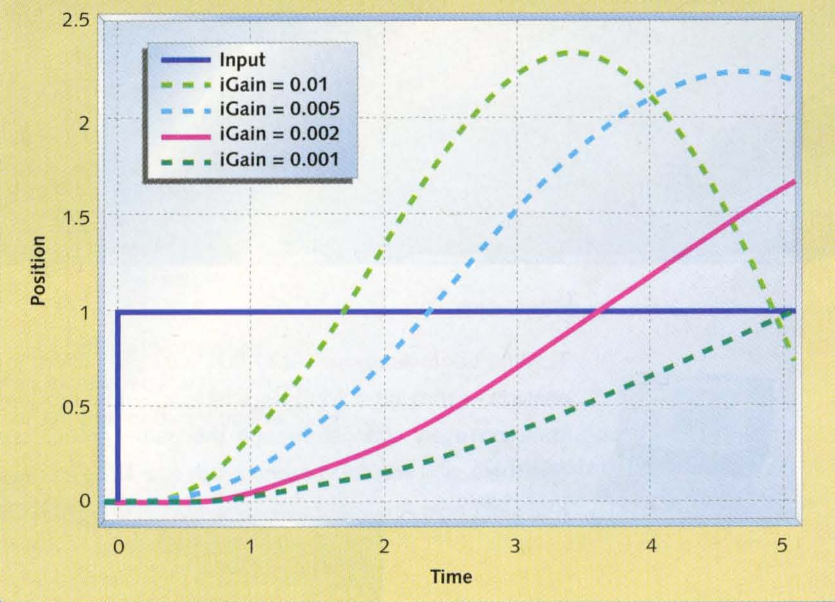


FIGURE 11 Motor and gear with pure integral control



American RAISONANCE

ARI

CAN Tools

Introducing American Raisonance. Your new supplier of IXXAT CAN development tools. Hardware and software to simplify your CAN implementation!

Analyzers

We now offer an extensive line of CAN analyzers, providing you with complete monitoring of your CAN network.

Interface Cards

We have the CAN interface card you need, whether it's ISA, PCMCIA, parallel port or USB. Utilizing your PC, 80320 or a C165 every card comes with:

- CAN BUS Monitor
- User manual
- Download and flash programs
- VCI Virtual CAN Interface

Protocols

CANopen Master/Slave software. Delivered in C source code and comes with example programs and technical support. DeviceNet Master/Slave software. Delivered in MASM/C source code and comes with example programs and technical support.

877-315-0792

www.amrai.com

INTEL CPU PLATFORMS

WIDE AREA NETWORKING

SYSTEM PLATFORMS

DSP TECHNOLOGIES

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.

When I presented the code for a differential element I mentioned that the output is proportional to the position change divided by the sample time. If the position is changing at a constant rate but your sample time varies from sample to sample, you will get noise on your differential term. Since the differential gain is usually high, this noise will be amplified a great deal.

When you use differential control you need to pay close attention to even sampling. I'd say that you want the sampling interval to be consistent to within 1% of the total at all times—the closer the better. If you can't set the hardware up to enforce the sampling interval, design your software to sample with very high priority. You don't have to actually execute the controller with such rigid precision—just make sure the actual ADC conversion happens at the right time. It may be best to put all your sampling in an ISR or very high-priority task, then execute the control code in a more relaxed manner.

FIGURE 12 Temperature control system with integral control. The kink at time = 2 is from the external disturbance

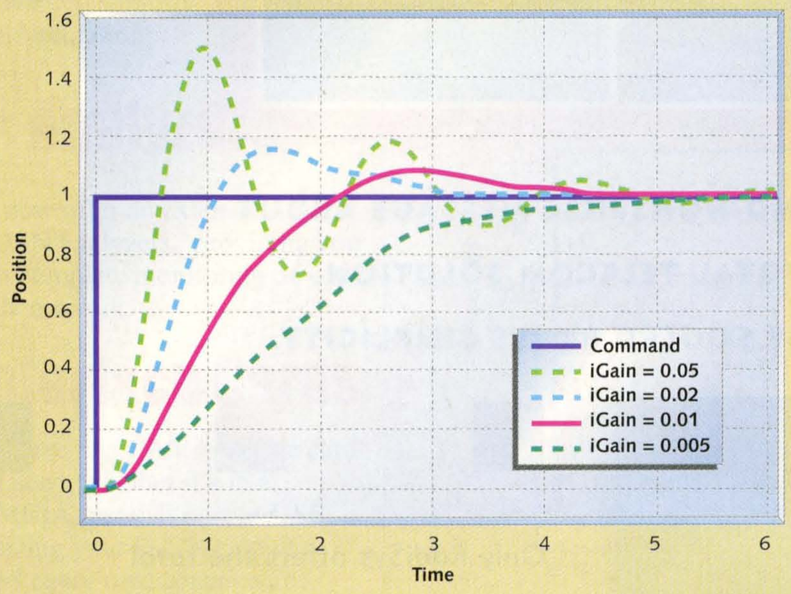
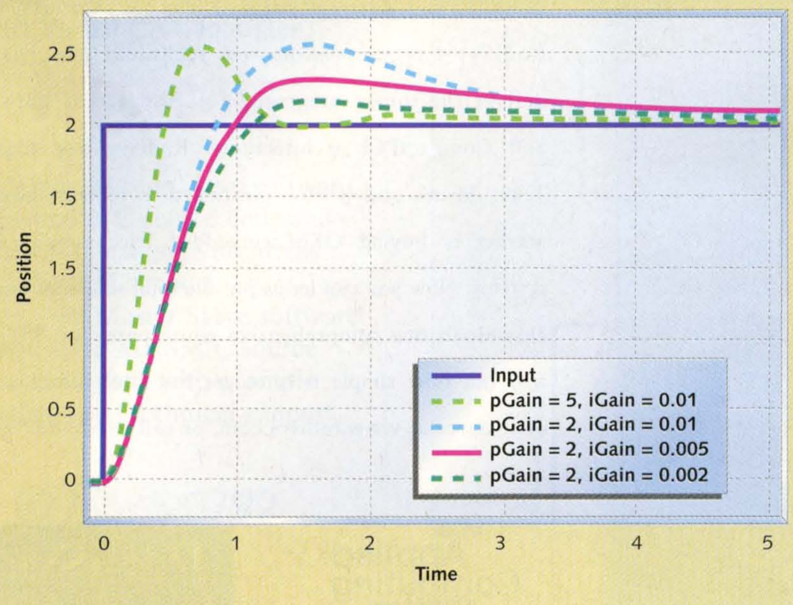


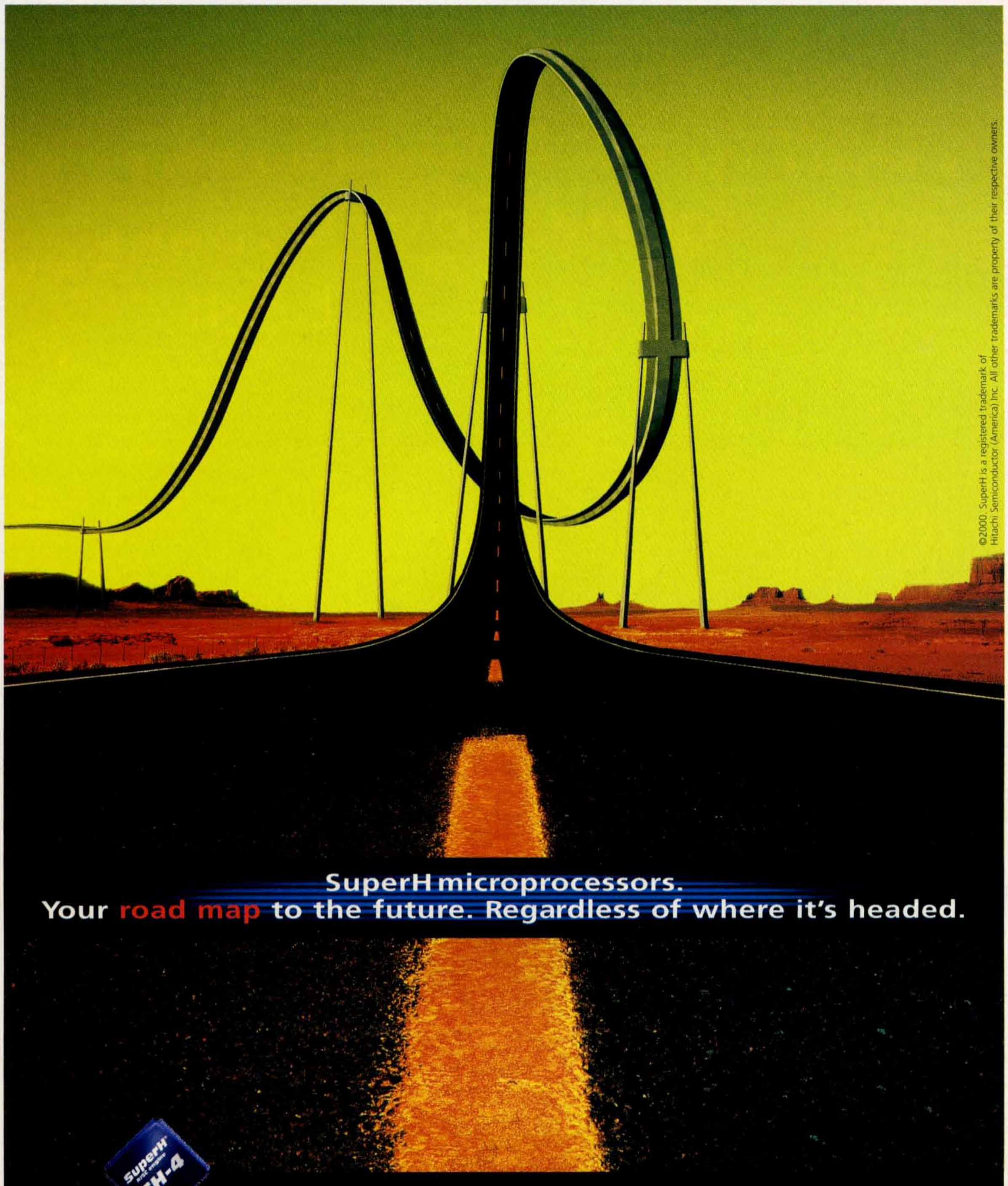
FIGURE 13 Motor and gear with PI control



Differential control suffers from noise problems because noise is usually spread relatively evenly across the frequency spectrum. Control commands and plant outputs, however, usually have most of their content at lower frequencies. Proportional control passes noise through unmolested. Integral control averages its input signal, which tends to kill noise. Differential control enhances high frequency signals, so it enhances noise. Look at the differential gains that I've set on the plants above, and think of what will happen if you have noise that makes each sample a little bit different. Multiply that little bit by a differential gain of 2,000 and think of what it means.

You can low-pass filter your differential output to reduce the noise, but this can severely affect its usefulness. The theory behind how to do this and how to determine if it will work is beyond the scope of this article. Probably the best that you can do about this problem is to look at how likely you are to see any noise, how much it will cost to get quiet inputs, and how badly you need the high performance that you get from differential control. Once you've worked this out, you can avoid differential control altogether, talk your hardware folks into getting you a lower noise input, or look for a control systems expert.

The full text of the PID controller code is shown in Listing 1 and is available at *www.embedded.com/code.html.*


SuperH microprocessors.
Your road map to the future. Regardless of where it's headed.
SuperH RISC engine SH-4

What's the fastest way to market? On board a SuperH® RISC SH-4, 32-bit MPU core. Everything you need to design your Internet appliance is mapped out and ready to drive off the lot. With SH-4-based chips from Hitachi and ST you get voice, graphics, data, and video in a high-performance, low power consumption environment. Then we throw in a comprehensive IP toolkit, unparalleled upward compatibility, and a price that's unbeatable. So no matter what the future has in store, the SH-4 lets you hit the road running. Check out www.SuperH.com today.

Voice Video Data

HITACHI
Semiconductor

ST

Tuning

The nice thing about tuning a PID controller is that you don't need to have a good understanding of formal control theory to do a fairly good job of it. About 90% of the closed-loop controller applications in the world do very well indeed with a controller that is only tuned fairly well.

If you can, hook your system up to some test equipment, or write in some debug code to allow you to look at the appropriate variables. If your system is slow enough you can spit the appropriate variables out on a serial port and graph them with a spreadsheet. You want to be able to look at the drive output and the plant output. In addition, you want to be able to apply some sort of a square-wave signal to the command input of your system. It is fairly easy to write some test code that will generate a suitable test command.

FIGURE 14 Heater with PI control. The kink at time = 2 is from the external disturbance

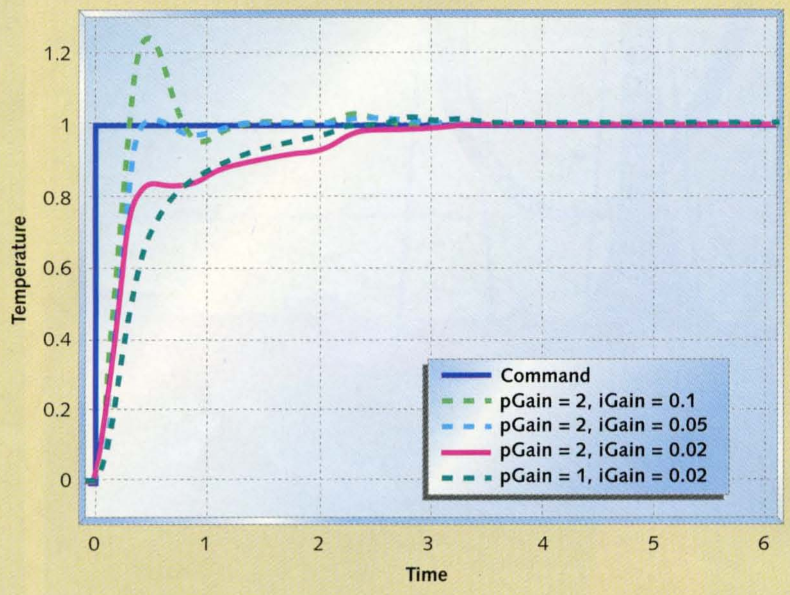
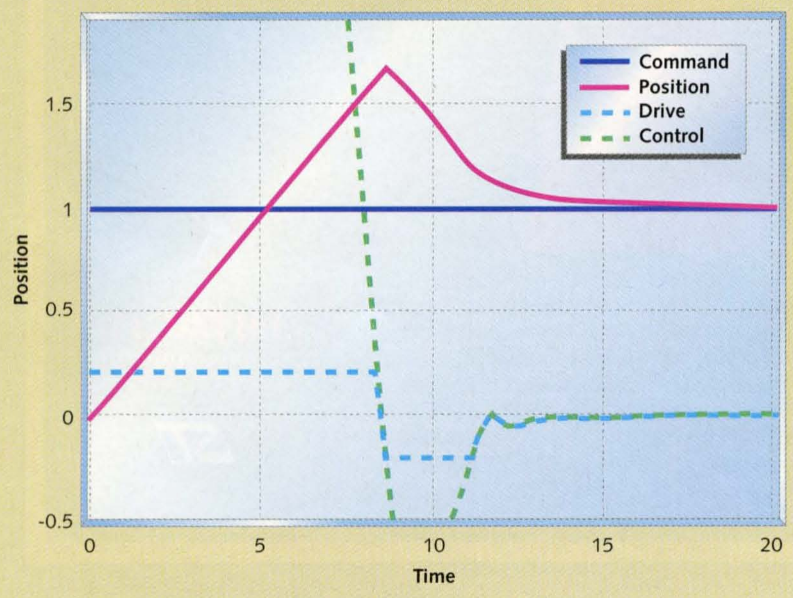


FIGURE 15 Motor and gear with PI control and windup

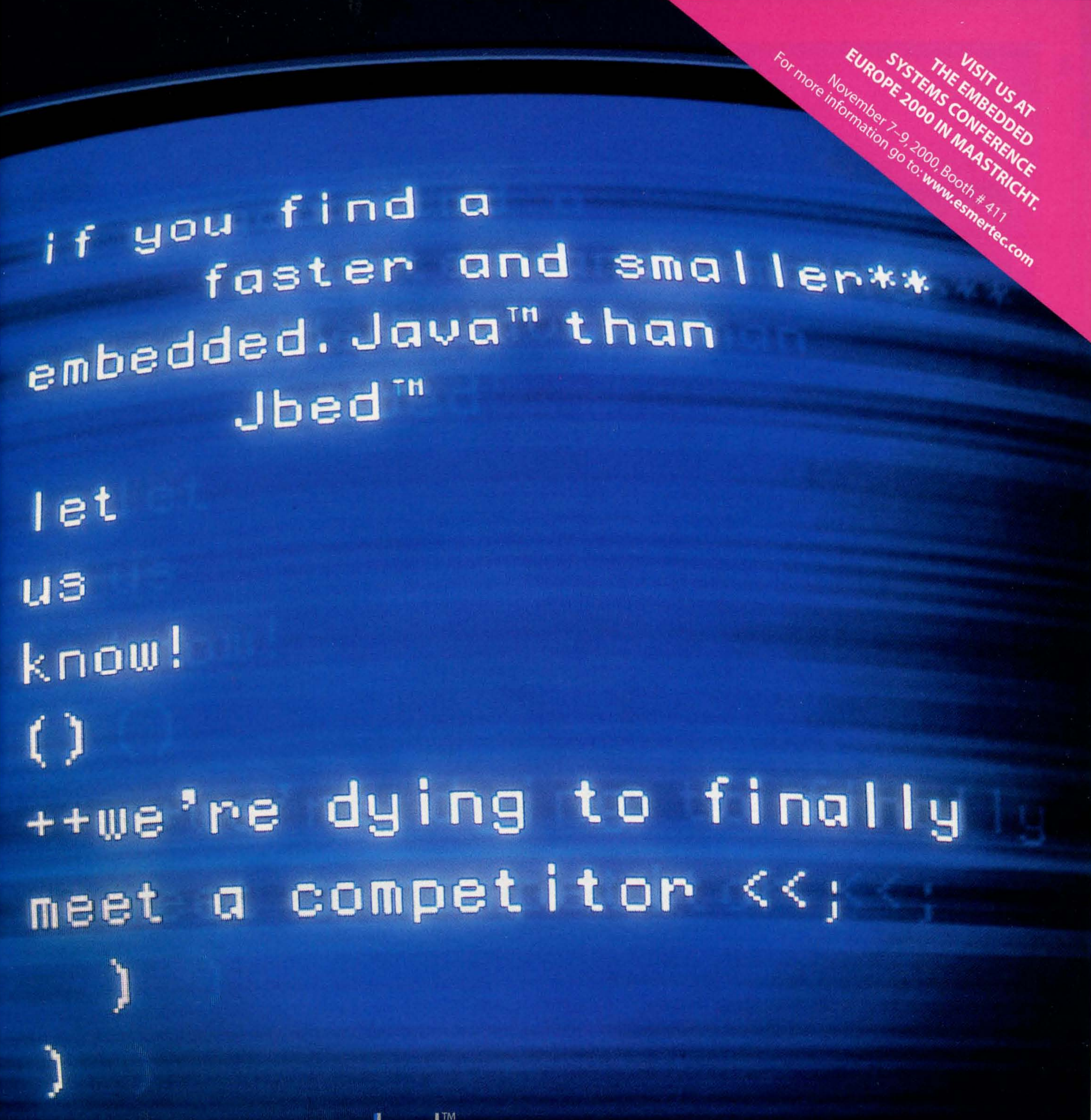


Once you get the setup ready, set all gains to zero. If you suspect that you will not need differential control (like the motor and gear example or the thermal system) then skip down to the section that discusses tuning the proportional gain. Otherwise start by adjusting your differential gain.

The way the controller is coded you cannot use differential control alone. Set your proportional gain to some small value (one or less). Check to see how the system works. If it oscillates with proportional gain you should be able to cure it with differential gain. Start with about 100 times more differential gain than proportional gain. Watch your drive signal. Now start increasing the differential gain until you see oscillation, excessive noise, or excessive (more than 50%) overshoot on the drive or plant output. Note that the oscillation from too much differential gain is much faster than the oscillation from not enough. I like to push the gain up until the system is on the verge of oscillation then back the gain off by a factor of two or four. Make sure the drive signal still looks good. At this point your system will probably be responding very sluggishly, so it's time to tune the proportional and integral gains.

If it isn't set already, set the proportional gain to a starting value between 1 and 100. Your system will probably either show terribly slow performance or it will oscillate. If you see oscillation, drop the proportional gain by factors of eight or 10 until the oscillation stops. If you don't see oscillation, increase the proportional gain by factors of eight or 10 until you start seeing oscillation or excessive overshoot. As with the differential controller, I usually tune right up to the point of

VISIT US AT THE EMBEDDED SYSTEMS CONFERENCE EUROPE 2000 IN MAASTRICHT.
November 7–9, 2000, Booth # 411
For more information go to: **www.esmertec.com**



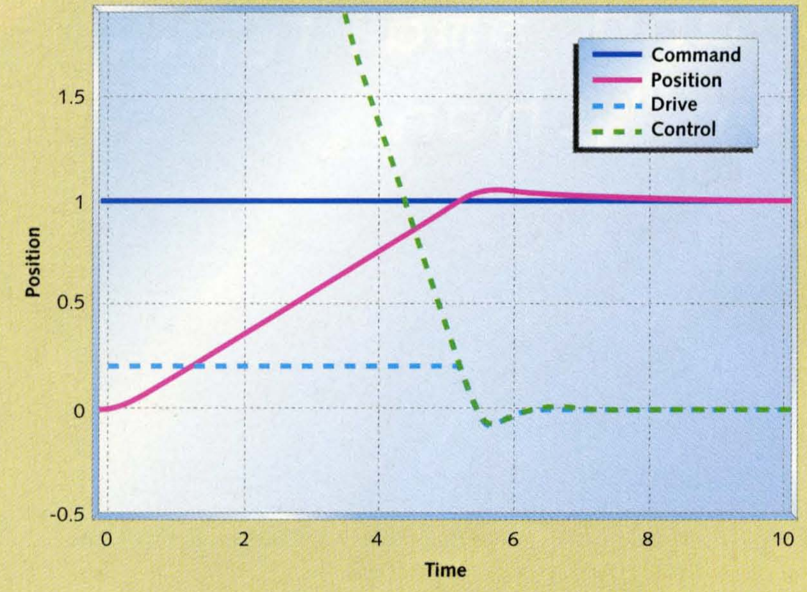
```
if you find a
    faster and smaller**
embedded.Java™ than
      Jbed™

let
us
know!
()
++we're dying to finally
meet a competitor <<;
   }
 }
```

Then again, we are very pleased to see that Jbed™ is alone at the top. After all there's a good reason for this: the Jbed Combined RTOS/Java™ Virtual Machine eliminates your concerns about the size, performance and cost of running a JVM in an embedded, let alone real-time, demanding system. Why? Firstly, Jbed IDE allows you to prototype your tasks, so you know you can meet your scheduling deadlines and unused processor load percentage. Secondly, you can download new or modified Java classes to your fielded, running systems without needing to reset, while the Jbed Admission-Facility continues to ensure that your deadlines are met. And last, but certainly not least, Java byte-code is complied on your PC or on your running target by our Target Bytecode Compiler into optimised machine code – never interpreted. By the way: if you are concerned about running hand-held applications on the KVM™, use our new Jbed Micro Edition CLDC™. **info@esmertec.com, www.esmertec.com**

esmertec

your partner for embedded real-time Java™

FIGURE 16 Motor and gear with PI control and integrator limiting



too much overshoot then reduce the gain by a factor of two or four. Once you are close, fine tune the proportional gain by factors of two until you like what you see.

Once you have your proportional gain set, start increasing integral gain. Your starting values will probably be from 0.0001 to 0.01. Here again, you want to find the range of integral gain that gives you reasonably fast performance without too much overshoot and without being too close to oscillation.

Other issues

Unless you are working on a project with very critical performance parameters you can often get by with control gains that are within a factor of two of the "correct" value. This means that you can do all your "multiplies" with shifts. This can be very handy when you're working with a slow processor.

Your DSP development is in a superior orbit with Blackhawk™!
DSP Enabling Technologies™
• Bundled Toolsets • Design Services
• Operating Systems • Consulting
• Development Hardware
WIN THESE PRIZES: Digital Camcorder
Digital Camera or DVD Player
www.blackhawk-dsp.com/contest107.shtml
Call: 1-877-983-4514
Blackhawk™
by EWA Inc.
© 2000 EWA

Run in the path of a Leader.

COPPER MOUNTAIN
High Performance DSL Networking

Life moves fast at Copper Mountain. In a short period of time, we've grown from an exciting start-up company to the world's leading provider of DSL networking systems with an international client list of global giants.

We've realized our vision to take high-speed data networking and carrier class transmission technologies and turn them into THE technologies of choice. Today, we're the fastest growing Internet-related company in the world. And with each passing day, we're becoming stronger, expanding our technologies, creating new applications and winning new clients.

If you're looking for the kind of powerful career that comes from being with a leader, look to us. You'll find a career filled with intriguing challenges, lucrative rewards and exciting advancement potential. To learn more about Copper Mountain and the many opportunities we have available, please visit our Web site at:

www.jobs.coppermountain.com

FIGURE 17 Precision actuator with PID control

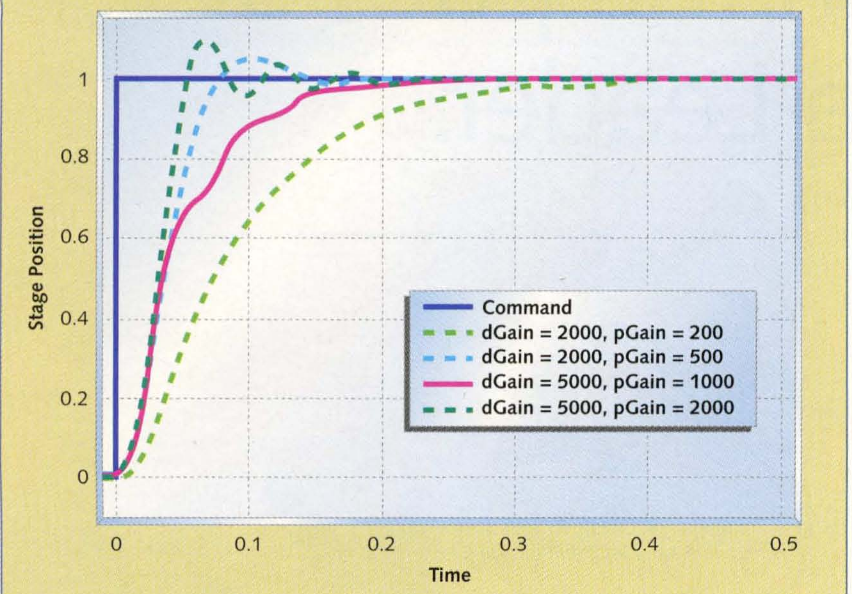
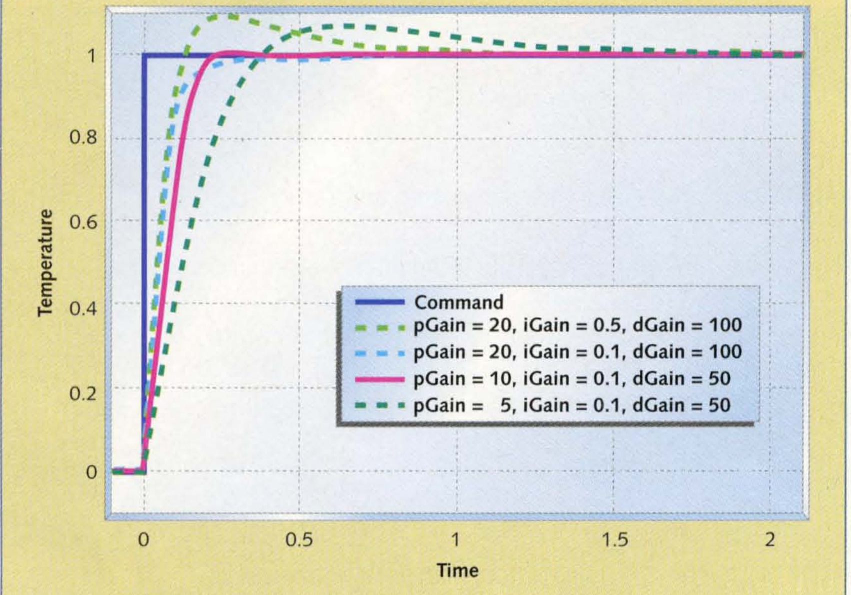


FIGURE 18 Heater with PID control



Sampling rate

So far I've only talked about sample rates in terms of how consistent they need to be, but I haven't told you how to decide ahead of time what the sample rate needs to be. If your sampling rate is too low you may not be able to achieve the performance you want, because of the added delay of the sampling. If your sampling rate is too high you will create problems with noise in your differentiator and overflow in your integrator.

The rule of thumb for digital control systems is that the sample time should be between 1/10th and 1/100th of the desired system settling time. System settling time is the amount of time from the moment the drive comes out of saturation until the control system has effectively settled out. If you look at Figure 16, the controller comes out of saturation at about 5.2s, and has settled out at around 6.2s. If you can live with the one second settling time you could get away with a sampling rate as low as 10Hz.

You should treat the sampling rate as a flexible quantity. Anything that might make the control problem more difficult would indicate that you should raise the sampling rate. Factors such as having a difficult plant to control, or needing differential control, or needing very precise control would all indicate raising the sampling rate. If you have a very easy control problem you could get away with lowering the sampling rate somewhat (I would hesitate to lengthen the sample time to more than one-fifth of the desired settling time). If you aren't using a differentiator and you are careful about using enough bits in your integrator you can get away with sampling rates 1,000 times faster than the intended settling time.

Exert control

This covers the basics of implementing and tuning PID controllers. With this information, you should be able to attack the next control problem that comes your way and get it under control. **esp**

Tim Wescott has a master's degree in electrical engineering and has been working in industry for more than a decade. His experience has included a number of control loops closed in software using 8- to 32-bit microprocessors, DSPs, assembly language, C, and C++. He is currently involved in control systems design at FLIR Systems where he specifies mechanical, electrical, and software requirements and does electrical and software design. You can contact him at robintim@teleport.com.

Personal Best

There's a feeling about achieving a goal, or simply reaching a new level of performance that's hard to put into words. It's not about winning or beating the next guy necessarily, just about knowing that you've done more than you've ever done before.

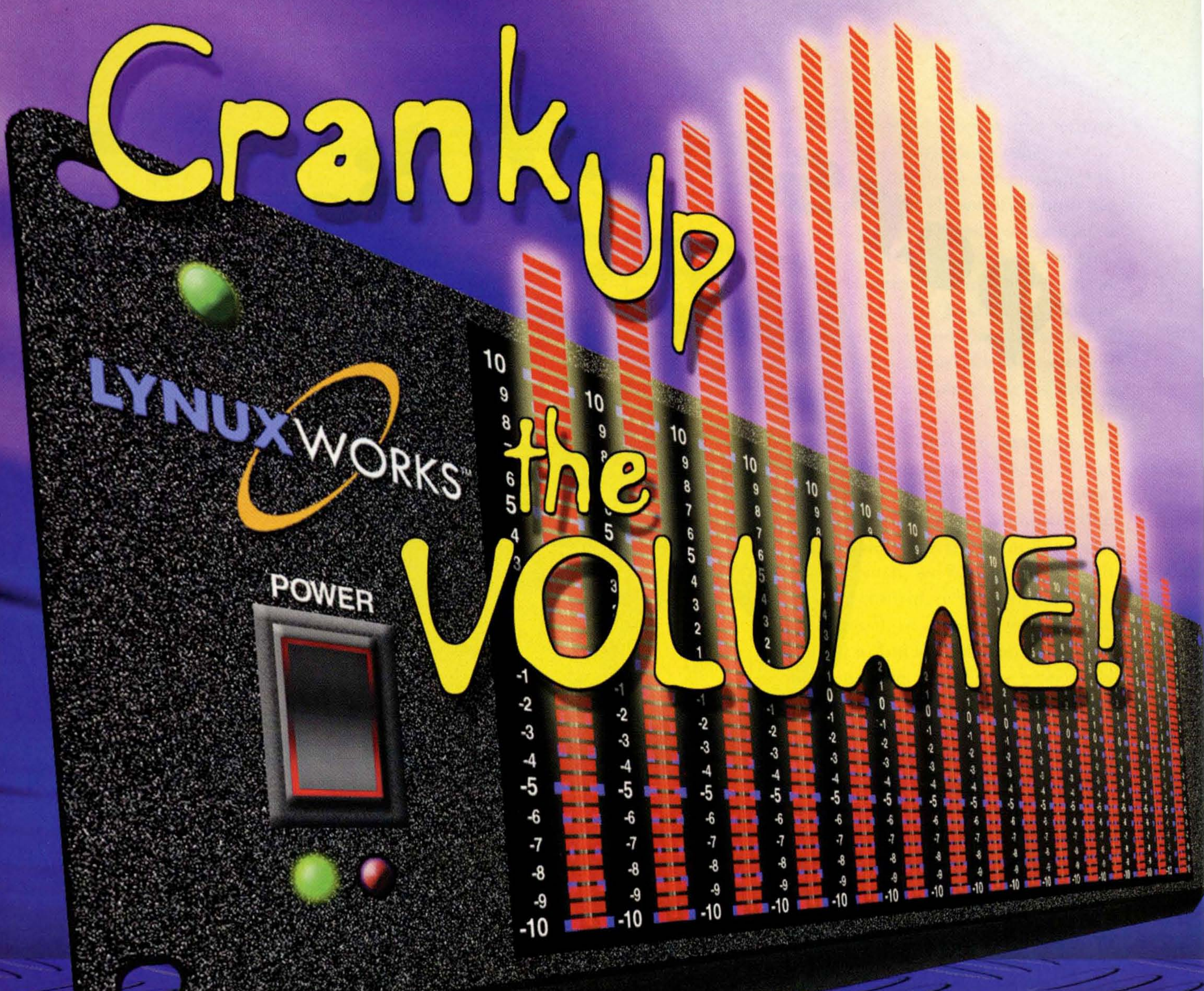
HI-TECH C users get that feeling often - the feeling of being "in the groove", the realization that not only did you meet your deadline, but you did it easily, without stress.

If you program embedded microcontrollers, and you want to raise your performance level, get HI-TECH C. World-beating performance from a tool that gets the job done without getting in your way, and backed by support our competitors can't believe - that's what *you* need to achieve your personal best. Call us today, or visit our web site for demos and information.

www.htsoft.com

HI-TECH Software
7830 Ellis Rd.
Melbourne FL 32904
Ph 800 735 5715
Fax 407 722 2902

HI-TECH
S O F T W A R E



LynuxWorks – Orchestrating Embedded Linux Solutions

LynuxWorks has powered-up to help you crank up volume on your embedded computing projects. Our recent merger with ISDCorp, a premier provider of quality Linux ports to embedded customers, has enhanced our ability to deliver proven results and tangible solutions.

Our reliable, scalable products – the Linux-compatible LynxOS® real-time operating system and the open-source BlueCat™ Linux operating system – along with our professional services and network of value-added partners can help you develop better embedded computers with a faster time to market – starting today. This is what LynuxWorks orchestrates and we bring it all to you in perfect harmony. Our decade of experience and ISO9001 certification makes us the low risk, proven partner you need to make yourself be heard in the embedded Linux market.

If you're tired of the noise other companies are making, but you're ready to crank up the volume with a proven solution,

you're ready for LynuxWorks.

For more information on LynuxWorks, our product and services visit our web site www.lynuxworks.com.

ALEXANDER WOLFE

HP and STMicroelectronics Launch "Lx"

A new VLIW architecture promises to shake up the system-on-chip design world.

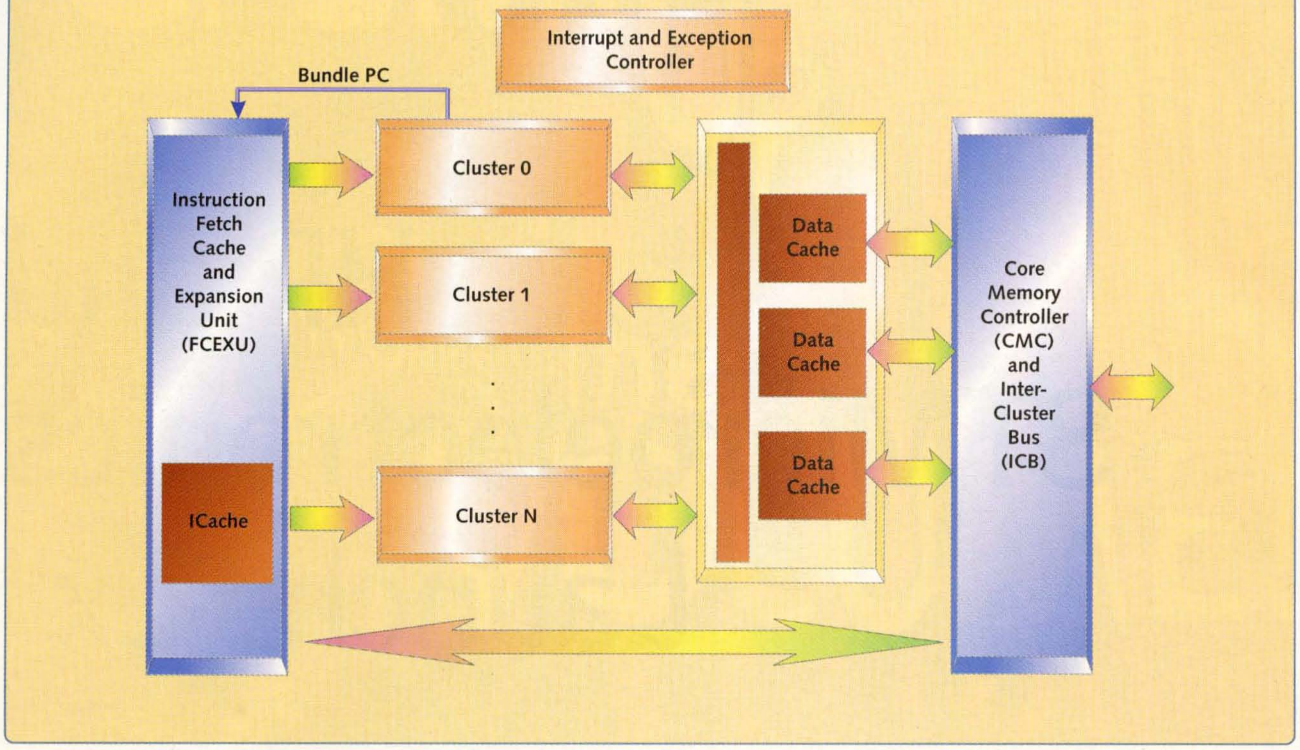
As buzz builds around the new approach toward embedded chip architectures called system-on-chip, two companies have teamed together to come up with what they think is a better mousetrap. This new family of embedded processors, which is designed to be more flexible and provide better performance at lower power dissipation and lower cost than traditional SOC approaches, will make its debut this month at the Microprocessor Forum in San Jose, CA.

The ground-breaking architecture, known as "Lx", evolved from a collaboration between Hewlett-Packard Labs, in Cambridge, MA, and France-based STMicroelectronics.

"The fundamental thing we believe is that, at any level of technology, we can produce a VLIW processor that's two to 10 times faster than an equivalent RISC processor," said Roger Shepherd, manager of advanced architectures at STMicroelectronics' Bristol, England labs. Shepherd also serves as ST's lead representative on the project, which is centered at HP's labs in Cambridge, MA.

System-on-chip, in general, refers to hardware runs where time-to-market and cost are critical factors. The chips themselves are often ASIC-based solutions, consisting of a central microprocessor core or a DSP core and multiple peripheral functions. However, reining in the engineering costs for small-volume designs—even when off-the-shelf cores are used—has remained an industrywide challenge.

FIGURE 1 The structure of a multi-cluster Lx architecture


Interrupt and Exception Controller
Bundle PC
Instruction Fetch Cache and Expansion Unit (FCEXU)
ICache
Cluster 0
Cluster 1
Cluster N
Data Cache
Data Cache
Data Cache
Core Memory Controller (CMC) and Inter-Cluster Bus (ICB)

New twist

Enter Lx. In fact, the Lx team has not eschewed SOC; they've just gone about it in a slightly different manner, one they believe gives the concept a little more utility.

The Lx architecture consists of a family of cores, according to Josh Fisher, head of HP Labs. Fisher is often referred to as "the father of VLIW" for his cutting-edge work on the technology at Multiflow Computer in the 1980s.

"It's a core for a SOC, but it's a family which has been designed for your domain of application, and typically designed with a large customer in mind," said Fisher. High performance comes from its VLIW heritage, which gives it the ability to exploit instruction-level parallelism (ILP). It is also customizable for different application domains.

"Lx is also enhanced by the fact that the customized cores all 'feel' to the user like they're in the same family," said Fisher—something that's previously been difficult to achieve with VLIW designs. "So, we were able to do customization without a lot of incremental engineering and cost to the customer."

Lx's creators reiterate that it is intended to be used in system-on-chip configurations, where it can be married up with appropriate additional processor capabilities and desired peripherals. It's certainly not being touted as a general-purpose microprocessor, which requires a notoriously expensive infrastructure of development tools. One possible application would be in a set-top box, where the multiple Lx cores could be configured on a single chip to handle, for example, general-purpose processing, MPEG decoding, and streaming media data types.

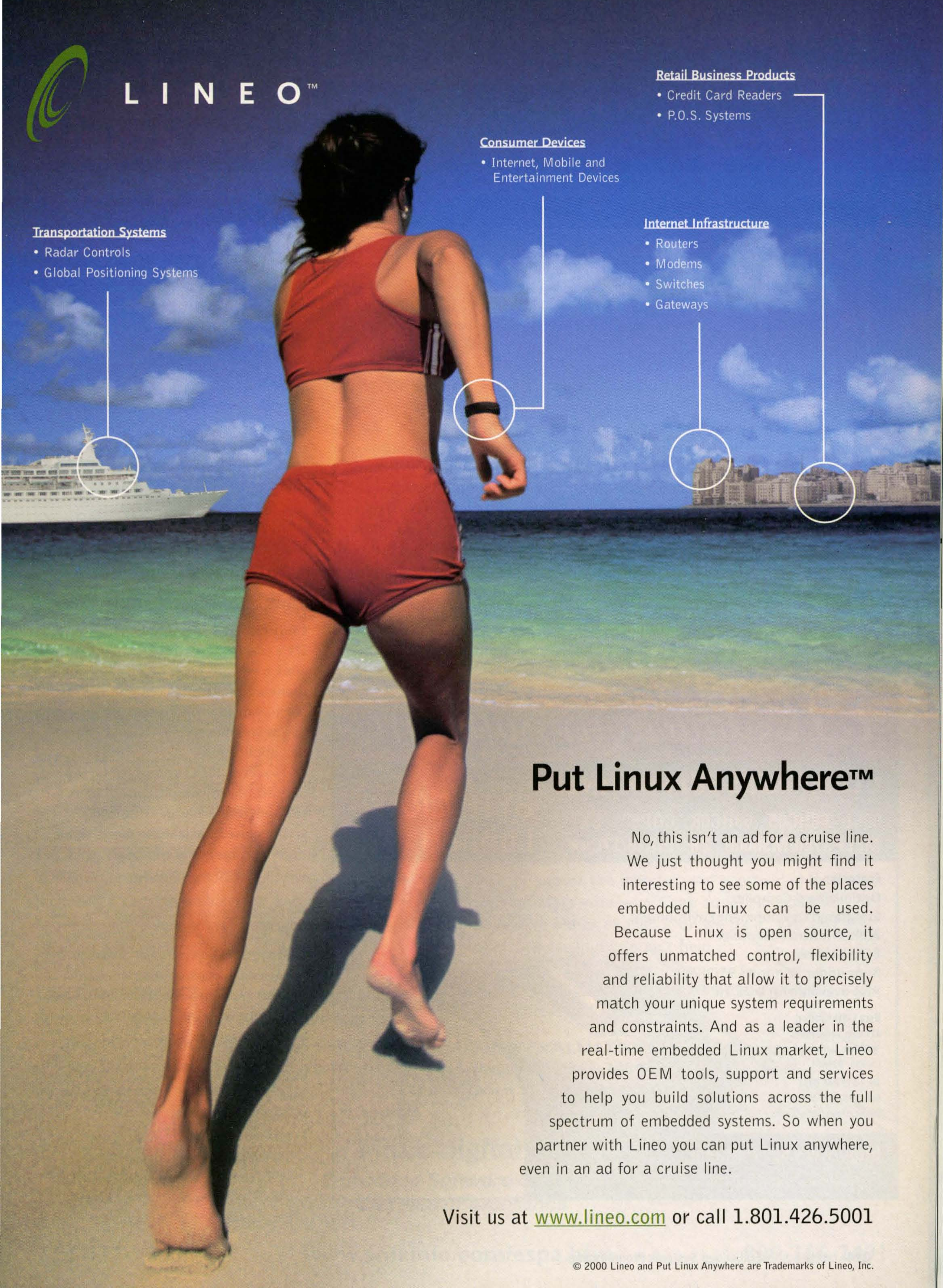
Although HP and ST do not like to use DSP in regards to Lx—"DSP is too specialized"—they note, perhaps the Lx is more DSP-like than anything else. (Strictly speaking, the companies see Lx as a convergence of DSP and microcontroller.)

Indeed, the Lx prototypes, which are due by the end of the year, will enter a DSP market that's been flooded by new devices and turned into a testbed for everything from reconfigurable computing DSPs to VLIW design. Competitors will likely run the gamut from high-flying upstarts such as the reconfigurable chips from Tensilica, through to VLIW DSPs such as Motorola's StarCore, TI's TMS320C62xx family, and Phillips' Trimedia. Also sure to remain in the mix are tried-and-true general-purpose platforms such as MIPS and ARM.

Under the hood

For each potential customer, the Lx custom processors would be architected by an automated hardware/software co-design process. Such chips would be churned out in very low volumes for specific, deeply embedded applications.

A major impetus for the venture is a perceived need to supply the burgeoning demands of smart embedded



Put Linux Anywhere™

No, this isn't an ad for a cruise line. We just thought you might find it interesting to see some of the places embedded Linux can be used. Because Linux is open source, it offers unmatched control, flexibility and reliability that allow it to precisely match your unique system requirements and constraints. And as a leader in the real-time embedded Linux market, Lineo provides OEM tools, support and services to help you build solutions across the full spectrum of embedded systems. So when you partner with Lineo you can put Linux anywhere, even in an ad for a cruise line.

Visit us at www.lineo.com or call 1.801.426.5001 devices such as Web processors, car navigation systems, and other new-age consumer-electronics devices being suggested by Internet designers.

The approach rests on two bedrock concepts, according to the two companies. First, there is Lx's new clustered VLIW core architecture (see Figure 1) and microarchitecture specialized to an application domain that ensures scaleability and customizability.

Next, HP and ST have developed a toolchain they say is based on "aggressive ILP technology." This is intended to give the user a uniform view of the platform at the programming language level.

The first major peek at the Lx project came in June, when engineers from the two companies delivered a joint paper on the effort at the 27th Annual International Symposium on Computer Architectures in Vancouver, B.C.

In terms of design, the basic Lx is laid out as a cluster of four VLIW execution units. That is, the low-end design would be a four-issue machine containing four 32-bit integer ALUs, two 16-by-32 multipliers, one load/store unit and one branch unit.

One level up in the family, an eight-issue device could be constructed by ganging together two clusters. The added cluster would bring along its own set of registers, along with other detritus to minimize the difficulties inherent in scaling up a processor.

Lx also relies on a two-level code compression scheme. The instruction cache is compressed so that unused slots don't consume space during encoding, the companies' engineers write. More ambitious is an effort in the works to compress binaries with Huffman-like techniques and then decompress blocks of instructions during I-cache refills.

To clarify each company's role in the overall effort, so far as is known publicly, HP developed the instruction set architecture and much of the software behind Lx, while ST implemented the hardware.

"The Lx project has concentrated on producing a highly automated way of exploring architectures," said Shepherd. "Not on implementing them, but exploring them."

Though Shepherd's statement sounds cryptic, he really means that the process is not yet entirely hands off. Right now, the tools output a compiled description of the device that must be tweaked. However, he sees things moving in the direction of eventual full automation.

In a separate effort that delves into some of the same concepts as Lx, HP is also working on a project called the PICO Architecture Synthesis System. (PICO itself stands for "Program In, Chip Out"). Led by Bob Rau, out of HP's labs in Cupertino, CA, PICO also aims to create a set of software tools that could make it economically feasible to quickly roll custom processors in low volumes for specialized embedded applications.

However, PICO appears to be working with more complex cores, akin to HP and Intel's highly advanced IA-64 architecture. Lx takes a more downsized tack, both in terms of power consumption and complexity. **esp**

Alexander Wolfe is editor-at-large for ESP. *He holds a BE in electrical engineering from Cooper Union. He has written assembly language code for embedded systems. He is co-author of* From Chips to Systems: An Introduction to Microcomputers, 2nd Edition *(Sybex, 1987). He can be reached at awolfe@cmp.com*

Resources

Faraboschi P., G. Brown, et al. "Lx: A Technology Platform for Customizable VLIW Embedded Processing," is available at *www.hpl.hp.com/cambridge/projects/cfp/docs/ISCA00_paper.pdf.*

PROductivity Software

C COMPILERS SUPPORTING:

8051 68HC05 68HC08 COP8™ SX MELPS740/M38000 Z8 PIC16/17Cxx

BYTE CRAFT Limited

Features:
Optimizing C Compiler
Integrated Development Environment
Built-in Macro Assembler
BClink Linker
Runs under: Windows 95/98/NT.
Also available for DOS, HP-UX and Sun Solaris.

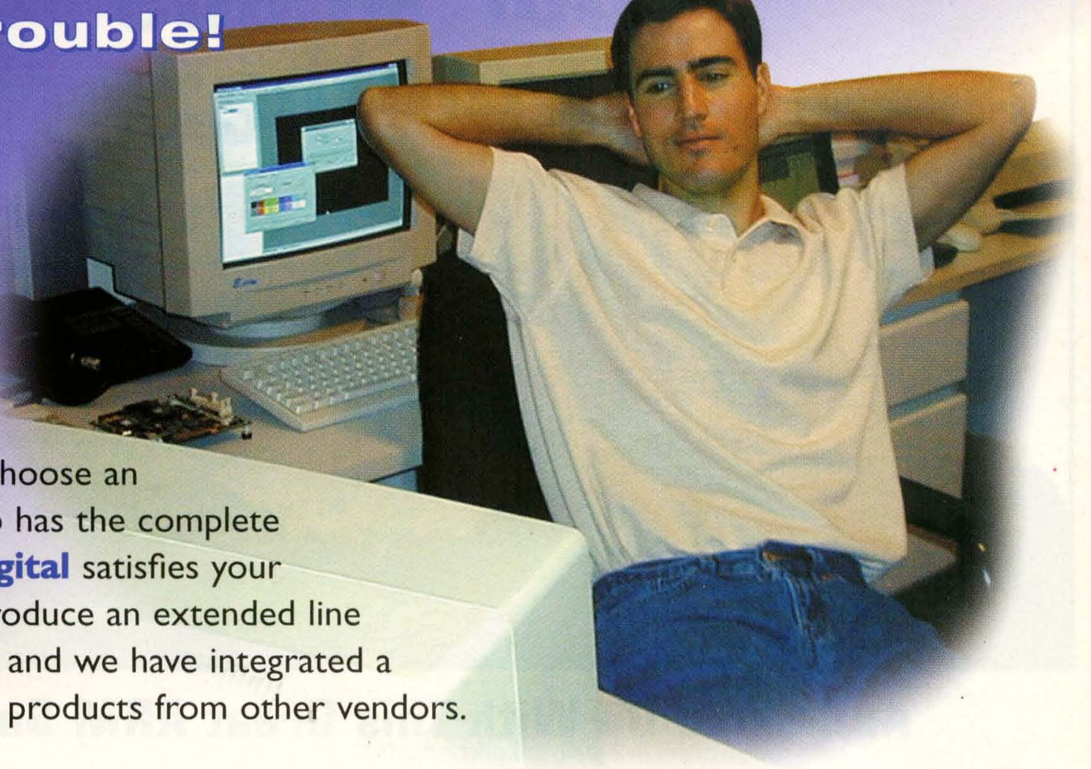
Byte Craft Limited
421 King Street North
Waterloo, Ontario
Canada N2J 4E4
Tel: 1-519-888-6911
Fax: 1-519-746-6751
Email: info@bytecraft.com

http://www.bytecraft.com

COP8 is a trademark of National Semiconductor Corporation

Most Programmers Want ...

Integrated tools that work together and don't cause trouble!



It makes sense to choose an **RTOS** vendor who has the complete solution. **Micro Digital** satisfies your requirement. We produce an extended line of embedded tools, and we have integrated a broad range of fine products from other vendors.

KERNEL - Low latency, hard-real-time, preemptive, multitasking, powerful API.

FILE - High-performance (up to 14 MB/sec) FAT 12/16/32 file system with all the drivers you are likely to need. Contiguous files supported.

NETWORK - Fully compliant TCP/IP stack with over 20 Ethernet drivers and 10 application-level protocols, including a capable web server and an HTML 4.0 browser with a very small footprint (200 KB). We also offer a USB host stack with 11 drivers.

USER INTERFACE - Text windows; graphics library; GUI with window builder, font capture, and image capture utilities, Unicode, and over a dozen drivers; 3D animation library with high-speed drivers for all common graphics boards.

TOOLS - Full support for popular compilers, linkers, locators, and debuggers with kernel awareness.

SPECIALTY - Dynamic load module support, x86 protected mode environment, NT console application loader, DOS emulator, C++ support package, boot loader and more.

SUPPORT - Excellent support is as close as your phone or the Internet and is provided by the programmers who wrote the code. No quibbling policy: we either explain it or we fix it. We also offer training and contract help.

TERMS - No royalties, source code included, 30-day free trial.

FREE EVALUATION KIT - Call today for details, **800-366-2491** or visit our website, **www.smxinfo.com/espa.htm**

μd

Micro Digital Inc

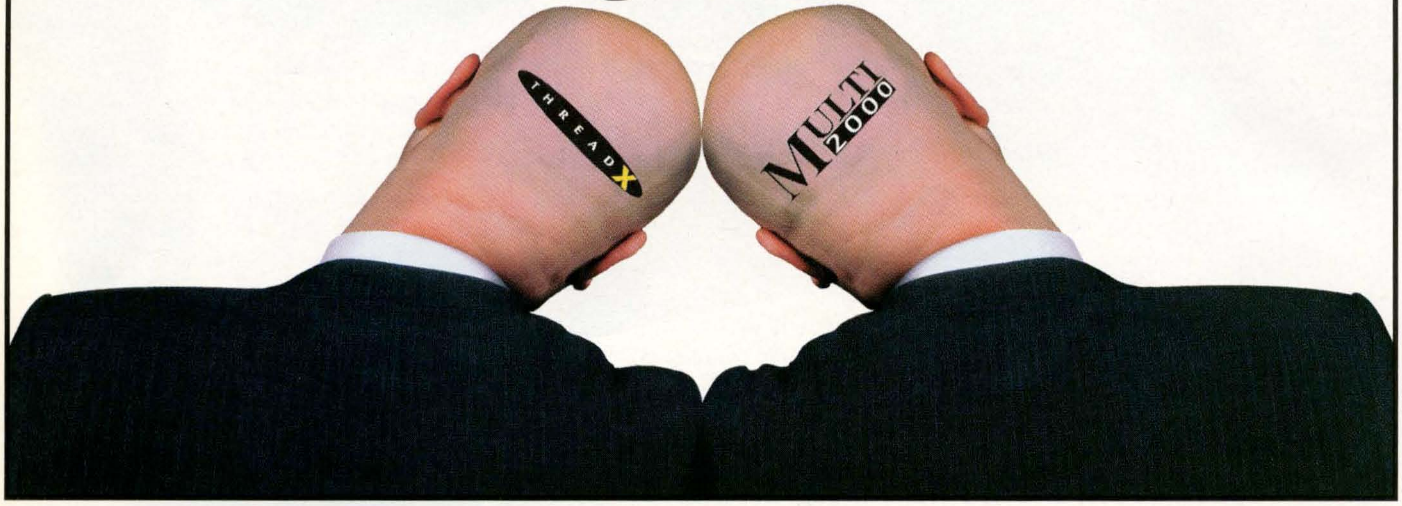
Embedded Software Outfitters

25 years of excellence

smx®

714-437-7333 www.smxinfo.com/espa.htm 800-366-2491

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

JOSEPH LEMIEUX

Abstracting System Hardware for Maximum Reuse

Abstracting hardware is difficult at times, but necessary. If you do it right, the resulting software will be much easier to reuse.

There are many reasons to adopt hardware abstraction. The foremost is to enable code reuse. The term *code reuse* has been defined in many different ways throughout industry, from code that is borrowed and slightly modified for each project, to libraries of features that are released as object code and linked together during development. For the purpose of this article, code reuse will be defined as any code, either at a source level or in library form, that is reused completely, with only changes in the definition of constants in a header. Abstraction that requires functions to be written or macros to be defined that change executable code is considered "borrowed" and not "reused."

Another justification for hardware abstraction is maintainability of the application code. If a standard method of accessing hardware exists, all software engineers in a department can maintain the code. When hardware changes, the abstraction changes in one location, not in every source code module. For example, changing the port for a digital input from Port 1 to Port 2 would require a change in the constant fed to the I/O layer, and each module will then benefit from this change after linking.

Finally, quality of the end product increases. By reusing code and abstracting hardware, the number of new or modified lines of code is reduced for each project. Since the probability of creating a defect increases with the number of lines of code written, by reducing this number, the number of defects will also decrease.

This article describes three methods of abstracting hardware from software applications. The first method, which I call the *function name* method, is a direct I/O access method using a different function for each input and output. This method is the fastest and easiest to use, but is also the hardest to change when the hardware changes. The second method uses an API for direct I/O. This method is more flexible and ports more easily to different hardware, but requires a slight addition in resources. Finally, the *message* method is an example of indirect I/O, where the I/O is obtained using a messaging system such as is found in OSEK/VDX. This message method works well in a complex environment where data is obtained both from hardware and other devices. The cost is additional resources.

The system model that will be used is shown in Figure 1. The operating system, utilities, and physical layer chosen do not impact the I/O abstraction directly and are not discussed here. The I/O layer is the basis of this article, and the impact to the application tasks is discussed. In the case of the message method, the I/O layer will be separated from the application tasks by a messaging system.

Function name

The function name method takes its name from the fact that a specific function or set of functions is defined for each input or output. These functions provide the value of an input and set the value of an output. In addition, sampling routines need to be written for the inputs, and driver routines need to be written for the outputs. The steps required to use this method are:

- Define all inputs and outputs in the application
- Define an access function for each input and output. This may optionally be a macro
- Create the input sampling routines required for all types of inputs. Each routine should ideally sample many inputs
- Create the output driver routines required for all types of outputs. Again, each routine should ideally drive many outputs
- Create support routines such as debounce routines, filtering, scaling, and so on. These may already exist in a library

This method will allow reuse of application source code. However, the I/O layers will use borrowed code instead of reused code, with the borrowed code being re-written for every application.

Define inputs and outputs

The first step in applying this method is to define all the inputs and outputs to the system. These may include digital inputs and outputs such as switches and lamps, analog inputs and outputs

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

NSPIRATION

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

FINAL CODE

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

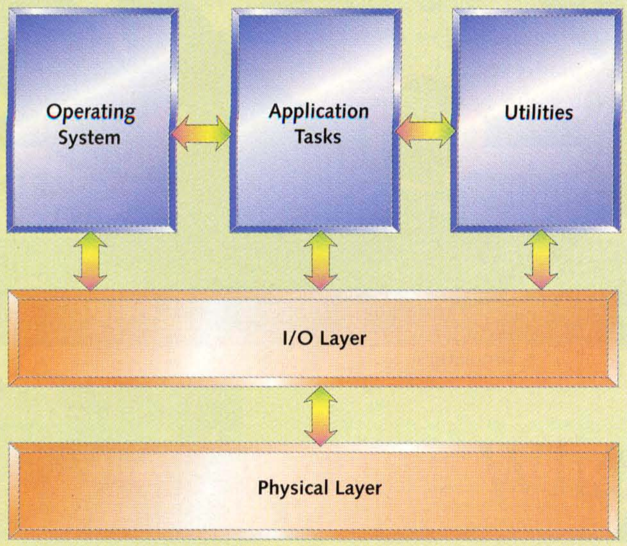
AMD: x86xx Family, ELAN 3/4/520C, K6
ARM: 6, 7/7T, 710, 9T, StrongArm
Hitachi: H8/300 Family
IBM: PPC 4/6
Infineon: 80C16x Family
Intel: 80x86/386/486 Real Mode,
80x86/386/486/Pentium Family Protected Mode
Mitsubishi: M16C
Motorola: DSP, StarCore, M.CORE, MPC500,
ColdFire 52xx/53xx, 68K Family, CPU 32 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

FIGURE 1 System model



LISTING 1 Obtaining a descrete input state

```
/*************************************
* Define an access of a discrete input that is packed in a byte with 7 other
* discrete inputs. The macro or routine returns 0 if the input is inactive or
* 1 if the input is active.
*************************************/
#define CheckNAMEState() \
  ((NAMEvariable & NAMEMASK ==  0) \
  ?INACTIVE:ACTIVE)
BOOLEAN CheckNAMEState(void)
{
  if(NAMEvariable & NAMEMASK == 0)
    return INACTIVE;
  else
    return ACTIVE;
}
```

such as voltages and currents, time-based inputs and outputs such as frequencies and PWM signals, and so on. These should be listed in a table along with any requirements such as range, granularity, debouncing, filtering, default values, and so on.

Define an access function for each input and output

This function or macro will always take on the function prototype of: `type inputfunction(void);` or `void outputfunction(type);`, where type is the type that contains the complete range and granularity of the input. For digital inputs, this is usually BOOLEAN, which most applications equate to an unsigned character. It may be any type of integer or float available in C, and may also be a structure. The return value is always the value of the input or output. Outputs may have two functions if other application tasks need to read the current output value. Following are examples of access function names, with `NAME` replaced by a unique name for each input or output:

```
BOOLEAN CheckNAMEState(void);
void SetNAMEState(BOOLEAN);
U8 GetNAMEValue(void);
void SetNAMEValue(U16);
```

After the function is defined, it needs to be created. In the case of a macro, the definition and creation are simultaneous. The listings presented here provide examples of routines for each of the previous examples. The macros are included. However, only one or the other will actually exist in the final product.

In Listing 1, the input is a discrete input that can have the value either ACTIVE or INACTIVE. Each input is packed into a byte that contains eight discrete inputs. The function must unpack the bit and return the proper value. Using a macro, the variable and mask must be defined globally so all routines can access them. The function provides some level of encapsulation of the variable.

For a discrete output, Listing 2 presents an example where the output port is written directly. Again with this macro, the port must be defined globally.

For analog inputs, Listing 3 describes the access function. In this case, a structure is defined for each analog input as follows:

```
typedef struct AnalogInputTypetag{
  U16 rawvalue;
  U16 filteredvalue;
  U16 defaultvalue;
  BOOLEAN validvalue;
} AnalogInputType;
```

If an error occurs, the default is used; otherwise the filtered value is used. It is up to the sampling routine to set the `validvalue` status.

Finally, for a timer-based output (PWM or variable frequency), Listing 4 shows how the value is buffered for

HEY!

We're in here!

We've stayed out of sight for far too long.

It's time everyone knew ...

that KADAK has been

quietly setting real-time standards since 1978.

That's right, from Sony to Hewlett Packard, Philips to Hughes Aircraft, over 1,000 of the most demanding companies in the world have chosen our AMX™ kernels to provide the processing power for all manner of smart devices.

From hidden embedded devices to the market-leading 3Com PalmPilot™ connected organizer, they've come to count on KADAK for superior real-time multitasking kernels, crystal-clear documentation, highly-responsive technical support and the ultimate in work and time-saving tools and utilities.

So, next time you need a real-time multitasking kernel remember ... You can count on KADAK!

AMX™

Real-Time Multitasking Kernels

KADAK

Visit us at www.kadak.com

Celebrating our 20th Anniversary

KADAK Products Ltd. 206-1847 W. Broadway, Vancouver, BC, CANADA V6J 1Y5 Tel: (604) 734-2796 Fax: (604) 734-8114 E-mail: amxsales@kadak.com output using an output driver. The value is first checked for validity and limited at the high end.

LISTING 2 Setting a descrete output

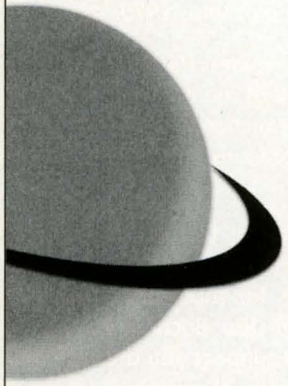
```
/*************************************
* Define an access of a discrete output that is connected to a port on the
* micro. The port is written to directly.
*************************************/

#define SetNAMEState(A) \
  ((A ==  0) \
  ?PORT1&=~NAMEMASK: PORT1 |= NAMEMASK)
void SetNAMEState(BOOLEAN state)
{
  if(state == 0)
    PORT1&=~NAMEMASK;
  else
    PORT1 |= NAMEMASK;
}
```

Create the input sampling routines
Next, create the input sampling routines and interrupt handling routines that are used to determine the value of the inputs. Each routine should sample as many inputs as possible in one function call. The following listings describe examples of some types of inputs and how they can be handled. Listing 5 shows an example where a single digital input port is sampled as a byte and debounced together. If the hardware is designed to optimize this algorithm, eight inputs are debounced at one time. The `DebounceDigital` routine will take the information from the sample port and the structure to determine the debounced value. This structure includes eight timers, one for each input port.

To sample analog inputs, Listing 6 shows a system that navigates a null terminated list of analog input definitions and filters the inputs. First, all analog channels are sampled. Then, for each entry in the input list, the raw value is passed to the filter function

The most widely deployed embedded database on any planet

SLEEPYCAT SOFTWARE



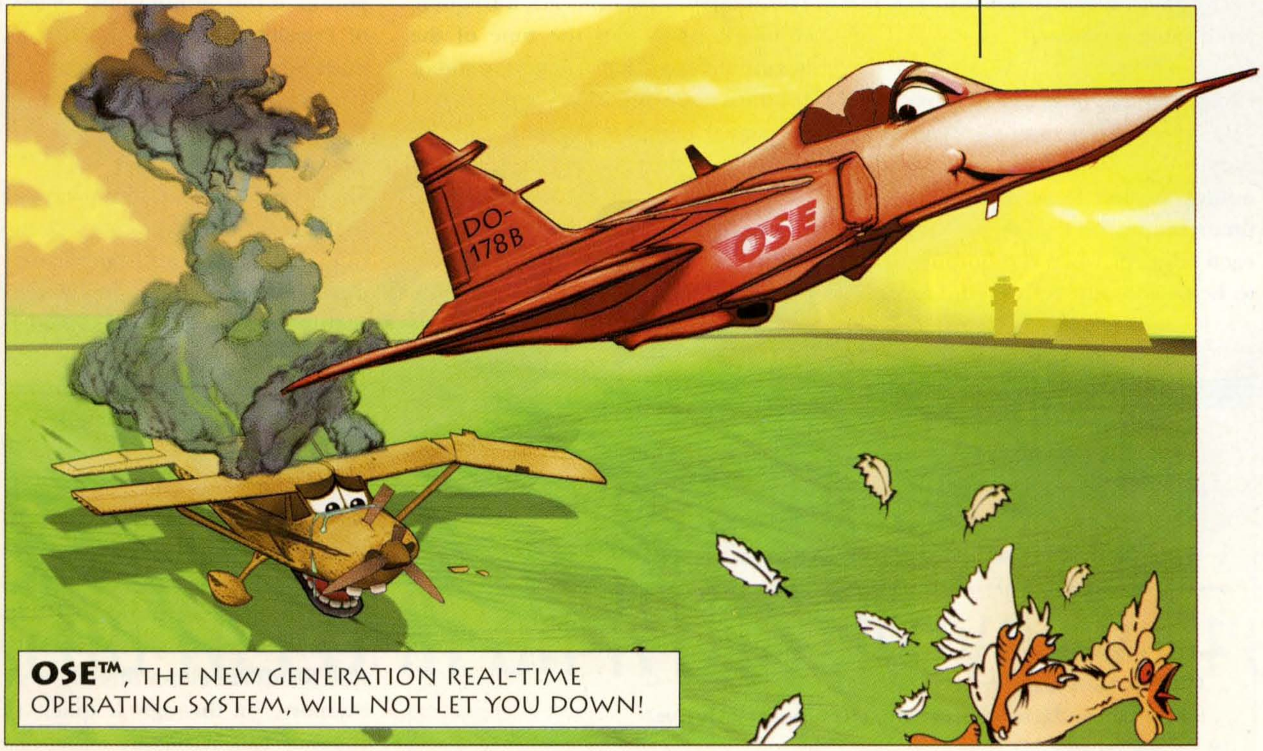
- Runs on Linux, UNIX, Windows, and VxWorks
- Supports thousands of concurrent users
- Scales to terabytes of data
- Survives power loss, application crashes, and hardware failures without losing data
- Provides easy-to-use APIs available from Java, C++, C, Perl, Python, and Tcl
- Links directly into the application: No separate server, no database administrator
- Small footprint
- Distributed with complete source code

Berkeley DB from Sleepycat Software

Phone: 1-877-SLEEPYCAT (toll-free in the US) or +1-510-526-3972 • Fax: +1-510-526-2537

www.sleepycat.com

– Conventional RTOS? …sorry!


DO-178B
OSE
OSE™, THE NEW GENERATION REAL-TIME OPERATING SYSTEM, WILL NOT LET YOU DOWN!

OSE THE NEW GENERATION RTOS!

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.

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.

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

with the filtered value and filter constant, and a simple first-order filter is performed on the system and the filtered value is updated.

Create the output driver routines
The output driver routines are used to output data when direct writes to the output are not feasible, such as in a timer output that must interrupt on each edge, or when the outputs need to be periodically refreshed. Listing 7 shows an output driver for a frequency output in an Intel 87C196 processor. This interrupt routine calls a frequency handler that sets the time of the next interrupt. The structure member *output->time_register is a pointer to the actual compare register, and the member output->period is the time between toggles of the output. EPA0 has been setup to toggle the output whenever a compare occurs, which presumes that the output is a 50% duty cycle frequency output. Therefore, output->period is _ the period of the output signal. The value of `FreqOutput.period` has been previously set using `SetFreqOutputValue()` function.

LISTING 3 Obtaining an analog input value

```
/*************************************
* Define an access of an analog input that is packed in a structure with the
* raw value and the filtered value. This input has the capability to determine
* if the input is valid and use a default value if invalid.
*************************************/

#define GetNAMEValue() \
  ((NAMEstructure.validvalue == TRUE)?\
  NAMEstructure.filteredvalue:NAMEstructure.defaultvalue)
U8 GetNAMEValue(void)
{
  if(NAMEstructure.validvalue = TRUE)
    return NAMEstucture.filteredvalue;
  else
    return NAMEstructure.defaultvalue;
}
```

LISTING 4 Setting a buffered output

```
/*************************************
* Define an access of a timer port output that is
* connected to a port on the micro. A buffer is written
* that is accessed on the next sample of the driver.
*************************************/

#define SetNAMEValue(A) \
  ((NAMEvariable > NAMEMAX)?\
  (NAMEvariable = NAMEMAX):(NAMEvariable = A))

void SetNAMEValue(U16 value)
{
  if(value > NAMEMAX)
    NAMEvariable = NAMEMAX;
  else
    NAMEvariable = value;
}
```

Create support routines
Finally, create the support routines that will be used to debounce inputs, filter inputs, and scale inputs or outputs. These routines are general in nature and will not be covered here. The focus, however, should be on generic routines that do not use any global variables. All information that is needed is passed to the routine in the function call.

Function name advantages and disadvantages
The advantage of using the function name method is that access to data is fast. Reads are either directly to global memory, or through a quick single function call. Writes are either directly to a port, to a global memory location, or through a quick single function call. The read and write functions that are written are straightforward and simple.

The disadvantages include difficulty to set up, poor code reuse, error prone, and extensive use of global variables. Whenever the source of an input changes, multiple sections of the code may need to be re-written or new macros defined. Defining new macros also requires re-compiling. Debugging of macros is very difficult.

API calls

The API calls method requires development of a generic API that is called by the application program using structures that contain the I/O information. The amount of actual code written for each new application is minimal or non-existent. The steps to implement the API calls method can be split into two sections—one time code development and integration into the application.

NUCLEUS

All You NEED in an RTOS. Royalty Free.

Prototyping

Development Tools

Kernels

C++

Networking

Graphics Products

Web Products

Accelerated Technology INCORPORATED

www.atinucleus.com

800.468.6853

334.661.5770

info@atinucleus.com

LISTING 5 Sampling digital input ports

```
/*************************************
* Define structure to hold digital inputs
*************************************/

struct debounceddigitalstruct {
  U8 timer[8];
  U8 value;
  }DebouncedDigital;

/*************************************
* Sample Digital Input port   sample complete port and debounce all 8 inputs
* Assumes that all 8 pins are inputs
*************************************/

void SampleDigitalInputs(void)
{
  U8 RawDigital = Port1;
  DebounceDigital(RawDigital, &DebouncedDigital);
}
```

One time development

- Classify the types of I/O that your organization uses
- Define type definitions for configurations of each type. Define a RAM type for data that changes and a ROM type for constant data
- Define macros to create ROM and RAM variables for each I/O
- Create routines to access, sample, and drive I/O

Integration into application

- Create each instance of the input or output in a configuration file
- Include the I/O files in your build scripts/makefiles
- Include the `io.h` file in any application module that accesses I/O
- Use the names defined in the configuration file to access the I/O in each application module
- Schedule the sampling and driving tasks in your system. This is the only

You Choose the Operating System.

You Choose the Single Board Computer.

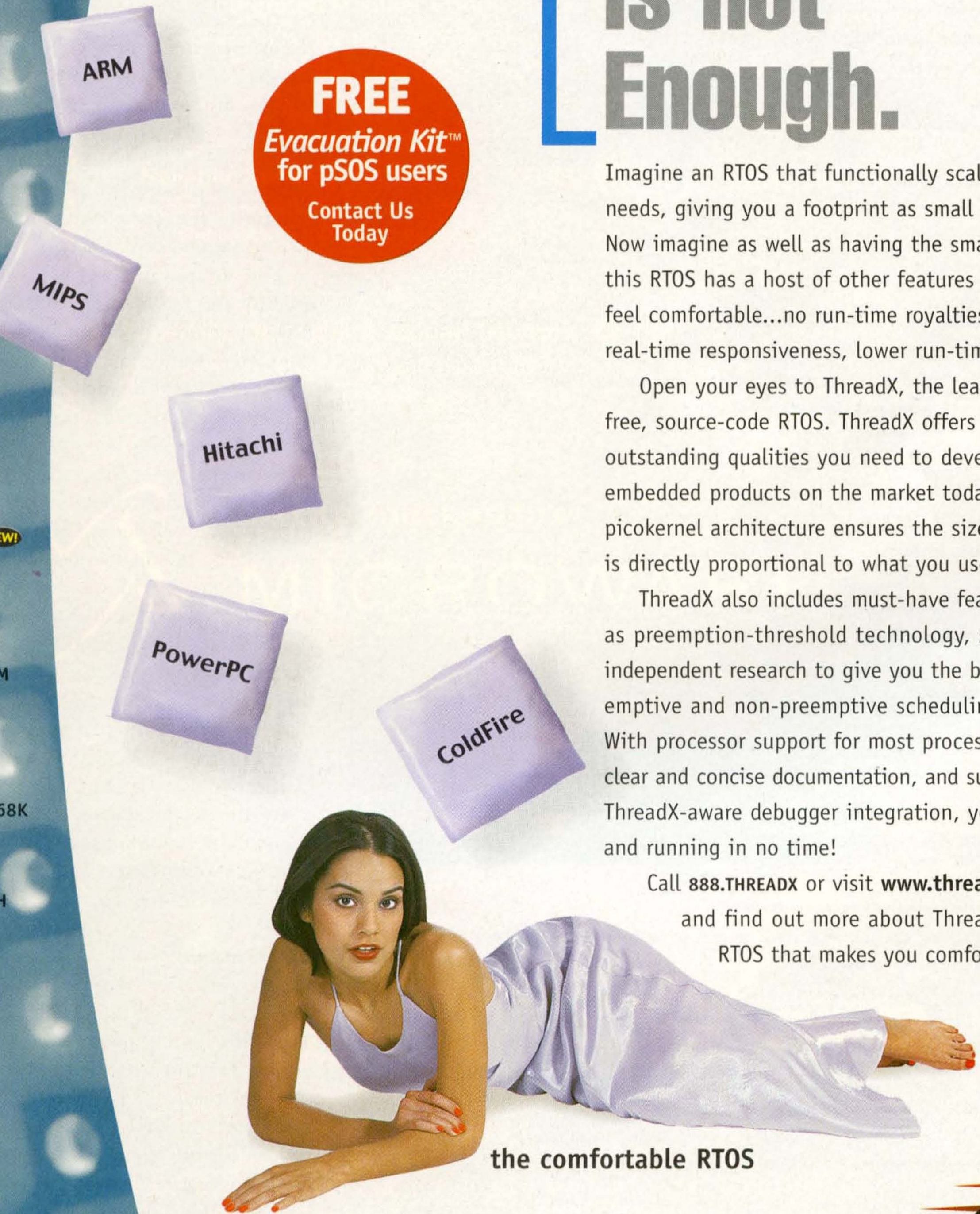
We Do the Rest.

Could this get any easier?

Hardware and software integration provided by Arcom Controls

Arcom CONTROL SYSTEMS

1.888.941.2224 • www.arcomcontrols.com



THREADX

FREE
Evacuation Kit™
for pSOS users
Contact Us
Today

Being The Smallest Is not Enough.

Imagine an RTOS that functionally scales to your needs, giving you a footprint as small as 3 Kbytes. Now imagine as well as having the smallest kernel, this RTOS has a host of other features to make you feel comfortable...no run-time royalties, better real-time responsiveness, lower run-time costs.

Open your eyes to ThreadX, the leading royalty-free, source-code RTOS. ThreadX offers you all the outstanding qualities you need to develop the best embedded products on the market today. Its unique picokernel architecture ensures the size of the RTOS is directly proportional to what you use.

ThreadX also includes must-have features, such as preemption-threshold technology, shown by independent research to give you the best of pre-emptive and non-preemptive scheduling models. With processor support for most processor families, clear and concise documentation, and such extensive ThreadX-aware debugger integration, you will be up and running in no time!

Call **888.THREADX** or visit **www.threadx.com** and find out more about ThreadX, the RTOS that makes you comfortable.

Processors Supported

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

the comfortable RTOS

eL
Express Logic, Inc.

tf **888.THREADX** • *fx* **858.613.6646** • **www.expresslogic.com** section where code may need to be written if the I/O functions can't be called directly by the RTOS or to encapsulate in an ISR

Classify I/O types
In this step, each type of I/O that is used by the organization is defined and the parameters are classified. For example, discrete inputs may be defined as being from either a digital or an analog port, will be debounced, and have a default value upon power-up. Other types of I/O may include discrete outputs, analog inputs/outputs, PWM inputs/outputs, frequency inputs/outputs, multi-state inputs/outputs, and so on.

LISTING 6 Sampling analog input ports

```
/*************************************
* Define structure to hold analog inputs
*************************************/

struct analoginputstruct {
  U8 constant;
  BOOLEAN validvalue;
  U8 channel;
  U8 defaultvalue;
  U8 filteredvalue;
  }AnalogInput;
/*************************************
* Sample Analog Input ports   sample and filter a list of analog inputs.
* Routine is passed a null terminated list of analog inputs.
*************************************/

void SampleAnalogInputs(AnalogInput **list)
{
  ConvertAllAnalogs();
  while(*list != NULL)
  {
    *list->filteredvalue = filter(rawanalog[*list->channel],*list->filteredvalue,
    *list->constant);
  list++;
  }
}
```

LISTING 7 Output driver for frequency outputs

```
/*************************************
* Define an interrupt service routine for an output and a routine for
* frequency outputs in general
*************************************/

interrupt void EPA0_ISR(void)
{
  UpdateFrequencyOutput(&FreqOutput);
}
void UpdateFrequencyOutput(FreqOutputStruct *output)
{
  *output->time_register += output->period;
}
```

Define type definitions
For each type above, define a variable (RAM) and a constant (ROM) structure that will be used by each instance of that type of input. For the previous example of discrete inputs from a digital port, the structures shown in Listing 8 would be defined.

The definition `IOHANDLE` will be used for the API function calls. In the RAM structure, `rawvalue` is the last sampled value of the input; `debouncedvalue` is the debounced state of the input; and `time` is the time that each `rawvalue` has been constant in order to debounce. Each RAM structure can hold eight digital inputs. The ROM structure includes `*digitalport`, a pointer to the digital port; `mask`, the mask of the bit in the port corresponding to the input; `debouncetime`, the time to debounce the input in samples; `ramlocation`, the offset into the array of `DigitalInputRam` variables that holds the particular input; `rammask`, the mask of the RAM location to extract the digital input; and `defaultstate`, the default state of the input.

Similarly, definitions of each of the input and output types defined previously would be created.

Define macros
Define macros that create the instances of each input or output. These macros encapsulate the definition of the structure, thereby allowing the structure to be modified to adapt to changes in the API. Examples of macros for our digital input example are in Listing 9. These macros will all appear in the configuration file.

The first macro reserves RAM memory for the packed digital inputs

The Embedded Software Solution That Wins.

SUPERIOR INTEGRATED DEVELOPMENT ENVIRONMENT

Internet and Multimedia Applications 1
High Performance Java™ VM, Internet Browsers, E-mail, MPEG, GUI, MP3, Third-Party Technology

Integrated Modular Connectivity 2
IEEE 1394, 802.11, USB, Fast Ethernet, Bluetooth, IXP 1200, ATM, WAP

Reliable, Secure, Proven OS–9® RTOS 3
Protected, process-based with threads support
Outstanding, integrated development tools

Expert Consulting and Support Services 4
Leading design, development and implementation
Total commitment to customer satisfaction

If you're in a race to market with a new product and you want to finish first, make sure you get the embedded software solution that goes the distance.

Get to market first™

MICROWARE®

To order a FREE evaluation CD go online to www.microware.com/first
Or call: 888–642–7609 (U.S. and Canada) • 515–223–8000 (International)

JAVA

in an array named `digitalin`. This array will be used later in the access and sample routines. Only one instance of this macro should exist in the configuration file.

The ROM macro defines the constant structure for each digital input. Each digital input will have one instance of this macro and it will be referenced by the unique identifier name. The member `port` is a global name that has been defined elsewhere and is specific to the microcontroller being used. The `debouncetime` argument is the number of samples to debounce the input. `position` and `rammask` is the offset position in the RAM array and the mask of the digital input in the packed RAM. The combination of `position` and `rammask` must exist only once in the configuration file. Finally, `default` is the default value that will be used in the RAM array for that input.

Create access, sample, and driver routines

The access functions should be generic and allow multiple inputs or outputs to be accessed using the same interface, with a different argument. For instance, `GetDiscreteState`, shown in Listing 10, is used to unpack all discrete inputs from a digital input. To expand to analog-based digital inputs, the routine would have to determine the type of argument that has been sent in the parameter list. This would require an additional parameter or structure that contains the type of the input. For simplicity, assume that `GetDiscreteState` will only return data on a digital input.

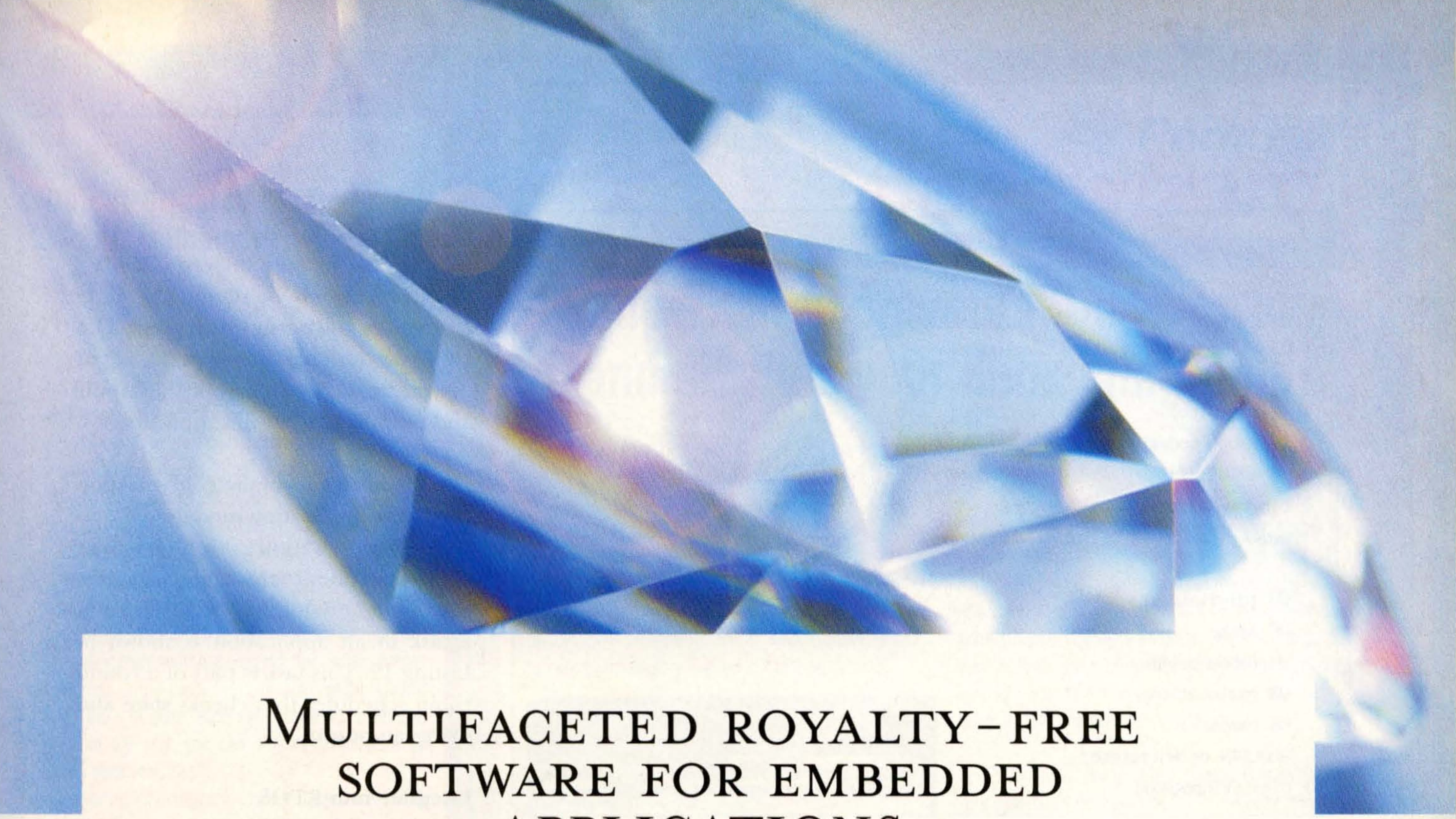
This function is called by passing the address of the ROM constant structure to the function. In this interface, the debounced value of the digital input is extracted from the RAM packed array. `INACTIVE` is returned if the bit is 0, otherwise `ACTIVE` is returned.

The sample routine is quite different from the one in the function name method, since it is more generic and needs to pack the inputs differently. The generic routine is shown in Listing 11.

In this function, the address of the ROM constant structure is passed and used to sample and extract the data from the digital port. The result is either 0 if low or non-zero if high. The routine, `DebounceDiscrete`, will use the raw data to debounce the input.

Create each instance in configuration file

A configuration file with a name such as `iocfg.h` needs to be created for the application. This file is included in the



Multifaceted royalty-free software for embedded applications.

Precise – What It Takes to Be a Leader

Precise has polished its products and customer-friendly licensing policy into a leading standard for embedded applications.

The Precise Way

Precise provides fully equipped and integrated application development platforms for some of today's most important markets – routers, subscriber equipment, VoIP and Internet appliances. Our products are top of the line, right off the shelf. But if you have unique requirements, we're there to give you the detailed attention needed for sparkling performance.

Precise Products and Services

Precise offers a dazzling selection of royalty-free software tools and systems:

- Precise/MQX™ Real-Time Operating System
- Precise/RTCS™ embedded Internet stack
- RTOS-aware and stack-aware debugging extensions
- Simulation tools for Precise/MQX and Precise/RTCS
- An extensive set of communication protocols
- Support for a wide range of general-purpose and DSP processors
- Numerous board support packages
- Training, application development services and consultation

PRECISE™
Software Technologies Inc.

Contact us:
www.psti.com
800-265-9833
info@psti.com

LISTING 8 Discrete input structure

```
typedef void * IOHANDLE;
typedef struct DigitalInputRamtag
{
  U8 rawvalue;
  U8 debouncedvalue;
  U8 time[8];
} DigitalInputRam;
typedef struct DigitalInputRomtag
{
  U8 *digitalport;
  U8 mask;
  U8 debouncetime;
  U8 ramlocation;
  U8 rammask;
  BOOLEAN defaultstate;
} DigitalInputRom;
```

LISTING 9 Macros for creating discrete inputs

```
#ifdef CREATE_MEMORY
#define CreateDigitalInRAM(number) \
  DigitalInputRam digitalin[number/8 + 1];
#define\
  CreateDigitalIn(name,port,mask,debouncetime,position,rammask,default)\
  DigitalInputRom const name = {\
  &port,mask,debouncetime,position,rammask,default};
#else
#define CreateDigitalInRAM(number) \
  extern DigitalInputRam digitalin[number/8 + 1];
#define\
  CreateDigitalIn(name,port,mask,debouncetime,position,rammask,default)\
  extern DigitalInputRom const name;\
#endif
```

LISTING 10 Obtaining a discrete input state

```
BOOLEAN GetDiscreteState(IOHANDLE name)
{
  if(digitalin[((DigitalInputRom *)name)->ramlocation].debouncedvalue &
     ((DigtialInputRom *)name)->rammask == 0)
  {
    return INACTIVE;
  }
  else
  {
    return ACTIVE;
  }
}
```

io.h file used by all application modules. However, the macro CREATE_MEMORY must only be defined in io.c, or an error will occur during link. The configuration file will contain all of the macros to define all of the inputs and outputs used in the application.

Use the names from the configuration file in the application module

In each application, use the names created in the iocfg.h file to access the inputs and outputs. An example of a task in an application is shown in Listing 12. This task is part of a round robin scheduler that checks state and drives an output.

Integrate into RTOS

The final step is to integrate the sample and driver routines into the RTOS. Each sample routine that periodically samples inputs needs to be scheduled. The advantage of these routines is that individual inputs can be sampled at different frequencies. The example in the function name method requires all inputs on a port to be sampled at the same frequency. This may require the writing of a shell routine that calls the function multiple times, once for each input. This could be made more efficient by using a null terminated list of inputs to be sampled and a small while() loop. ISRs that are specific to the application and call API routines must also be developed.

API calls: advantages and disadvantages

The advantage of this method is that little or no code is written when I/O is added, deleted, or modified. Only constants using the macros in the configuration file need to be changed. This method has high reusability and high portability since each function receives a void pointer, and changes in the structure to which the pointer is referenced do not affect the application code generated.

The disadvantages of this method are slight increases in ROM and CPU usage. However, since the difference has not been benchmarked, there may

Object-Oriented Meets Web Development

by Adam Kolawa

We continue our discussion by taking a look at object-oriented techniques. In light of the current trend towards Web development, it is only natural that developers will try to compare object-oriented development to the Web paradigm. But a closer examination reveals that with a few mental adjustments, object-oriented developers can use both approaches in tandem for their various projects.

The main mental adjustment that object-oriented developers must make is to understand how state is recorded. Web pages do not have memory, so each interaction with a Web page has to be handled independently without regard for what events occurred in the past. Even dynamic pages cannot keep track of state; each time you travel from one page to the next, you can only carry the state by using state variables.

Storing information in state variables breaks the object-oriented paradigm. A fundamental principle of object-oriented programming is that all necessary information is encapsulated inside the objects. Objects have to carry state information from one page to another page. But in Web development, state dies each time a user leaves a particular page, and each page is a new instance.

Developers who are trained in object-oriented programming often enter Web development with the false assumption that if they have the variables stored in their objects, the variables will be ready to apply to the new page. Ironically, these developers can gain a better understanding of Web development if they just think back to the old rules of FORTRAN programming, in which external variables carried information between functions and subroutines. The functions and subroutines executed tasks on their own, out of the developer's sight.

Dynamic Web development most certainly breaks the object-oriented paradigm, and it keeps object-oriented developers on their toes. Object-oriented developers can take Web development as a new challenge to their mental versatility, and can use its best points to enhance their applications. One example of how the two techniques can complement each other is Java Server Pages (JSP) technology, which encapsulates business logic in JavaBeans but allows developers to build HTML pages to control the display of the application.

Adam Kolawa, Ph.D., is Chairman and CEO of ParaSoft. You can reach him at ak@parasoft.com

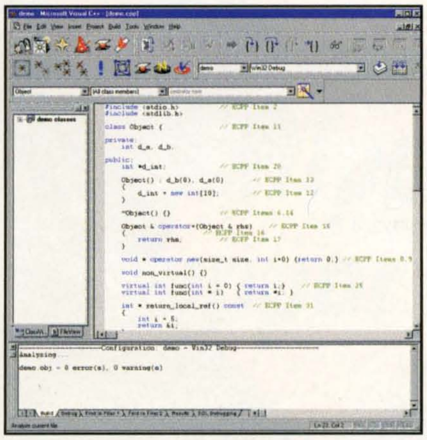
ParaSoft®

ParaSoft®
codewizard®

CodeWizard® prevents C/C++ errors and slashes your debugging time

"CodeWizard is awesome. I am recommending our management to purchase at least 5 copies.... ParaSoft has a lot to offer to the world of C++ development." ***-Piyush Shaw, Arrowsmith Technologies, Inc.***

CodeWizard® is a tool that helps you prevent errors by automatically performing static analysis on C/C++ code. The first thing you'll notice about CodeWizard is that it cuts down on the thing you hate the most: debugging. Preventing errors with CodeWizard is painless and saves you time in the long run.



```
#include <stdio.h>              // ECPP Item 2
#include <stdlib.h>

class Object {                  // ECPP Item 11

private:
    int d_a, d_b;

public:
    int *d_int;                 // ECPP Item 20

    Object() : d_b(0), d_a(0)        // ECPP Item 13
    {
        d_int = new int[10];         // ECPP Item 12
    }

    ~Object() {}                // ECPP Items 6,14

    Object & operator=(Object & rhs)      // ECPP Item 15
    {                              // ECPP Item 16
        return rhs;             // ECPP Item 17
    }

    void * operator new(size_t size, int i=0) {return 0;} // ECPP Items 8,9

    void non_virtual() {}

    virtual int func(int i = 0) { return i;}    // ECPP Item 25
    virtual int func(int * i)   { return *i; }

    int * return_local_ref() const   // ECPP Item 31
    {
        int i = 5;
        return &i;
```

```
------------------Configuration: demo - Win32 Debug--------------------
Analyzing...
demo.obj - 0 error(s), 0 warning(s)
```

CodeWizard enforces over 120 sophisticated C/C+ coding standards.

CodeWizard automatically enforces a sophisticated set of C/C++ coding standards. These language-specific rules are designed by industry experts such as Scott Meyers to help you avoid dangerous coding constructs. CodeWizard parses your code, clearly displays violations of the coding standards, and gives you all the information you need to make repairs, including suggestions on better coding constructs. Follow CodeWizard's coding suggestions, and you'll immediately begin to write cleaner, more reliable code.

The key to CodeWizard is that you can customize the program to fit your development schedule. With a few clicks, you can tell CodeWizard to enforce only the standards that are most important to you at a particular stage of your project. Using the RuleWizard feature, you can point and click to build your own sophisticated coding standards. RuleWizard guides you through the coding standard building process and lets you know when your coding standard is ready for enforcement by CodeWizard.

You won't have any problem adding CodeWizard to your arsenal of development tools, because it installs directly into Windows and UNIX development environments. In Microsoft Developer Studio, you'll begin testing with a single click of the CodeWizard icon. On UNIX platforms, CodeWizard is a wrapper around the compiler. Learn a few simple commands, and you'll be on your way to cleaner code.

If you would like to prevent errors with a customizable tool that conforms to your development cycle, download CodeWizard today at www.parasoft.com/best_oct.htm, or call (888)305-0041 for more information.

Download your
FREE DEMO at:
www.parasoft.com/best_oct.htm

be no increase or a slight decrease in total ROM usage due to the efficiencies of integrating functions into the API instead of creating multiple functions.

LISTING 11 Generic routine for sampling discrete inputs

```
void SampleDiscreteInput(IOHANDLE name)
{
  U8 RawDigital = *((DigitalInputRom *)name->port &
  ((DigitalInputRom *)name->mask;
  DebounceDiscrete(RawDigital, name);
}
```

LISTING 12 Example application task

```
void CheckRelay(void)
{
  if((GetDiscreteState(&switch1)==INACTIVE) ||
    (GetDiscreteState(&switch2) == INACTIVE))
  {
    SetDiscreteState(&relay,INACTIVE);
  }
  else
  {
    SetDiscreteState(&relay,ACTIVE);
  }
  TerminateTask();
}
```

LISTING 13 Discrete input structure with message field

```
typedef void * IOHANDLE;

typedef struct DigitalInputRamtag
{
  U8 rawvalue;
  U8 debouncedvalue;
  U8 time[8];
} DigitalInputRam;

typedef struct DigitalInputRomtag
{
  U8 *digitalport;
  U8 mask;
  U8 debouncetime;
  U8 ramlocation;
  U8 rammask;
  MESSAGE message;
  BOOLEAN defaultstate;
} DigitalInputRom;
```

Message

The message method is almost identical to the API calls method, except that it uses the RTOS's interprocess communication (IPC) system to transfer the status of the data. A major assumption is that the RTOS IPC provides messages that are not consumed when they are received. An example is the OSEK/VDX unqueued message in the COM specification. (See my previous OSEK/VDX articles in *ESP*: "OSEK/VDX Standard: Operating System and Communication," March 2000, p. 90; and "OSEK/VDX Network Manager and Implementation Language," April 2000, p. 96.) Unqueued messages retain the last value, and can be "received" multiple times without losing the value. The differences between message and API calls methods are:

- Type definitions change slightly
- `ReceiveMessage` instead of the API function `Get`
- `SendMessage` instead of the API function `Set`
- Definition of messages for all inputs required

Type definitions

From the example in API calls of the digital input, the RAM type variable does not change. However, the ROM type variable adds a reference to the message that contains the input as shown in Listing 13.

Macros

Since the ROM structure changed, a change needs to be made to the macro that creates the structure instances to include the name of the message, as shown in Listing 14.

Routines

The access routines have been replaced by the IPC routines, and the input sampling routine changes slightly as shown in Listing 15. The addition of the `SendMessage` function places the result in the system message for use by the algorithms. The `DebounceDiscrete` function returns the debounced state of the discrete.

IF YOU USE:

ATI Nucleus™

Embedded Power RTXC™

Express Logic ThreadX™

Kadak AMX™

Precise/MQX™

Your own custom RTOS

ON:

ARM®

StrongARM

ARC

PowerPC™

Contact MetaWare for a free evaluation www.metaware.com/eval

Thanks to its object-oriented architecture, MetaWare's SeeCode™ debugger can be set to fit your project requirements. Through documented APIs, SeeCode™ interfaces with system-level tools, software/hardware execution tools, or other key elements of your own environment. The RTOS API can make SeeCode aware of your RTOS, even if it's home-grown.

MetaWare® INCORPORATED

2161 Delaware Avenue, Santa Cruz, CA 95060 USA • Toll-free: 1.877.TOOLSET (866.5738) • Fax: 1.831.429.9273

12 Rue Saulnier, 92800 Puteaux, France • Tel: 33 (0)1 47 28 86 31 • Fax: 33 (0)1 47 28 46 99

LISTING 14 Macros with message field added

```
#ifdef CREATE_MEMORY
#define CreateDigitalInRAM(number) \
  DigitalInputRam digitalin[number/8 + 1];
#define\
CreateDigitalIn(name,port,mask,debouncetime,position,rammask,message,default)\
DigitalInputRom const name = {\
&port,mask,debouncetime,position,rammask,message,default};
#else
#define CreateDigitalInRAM(number) \
  extern DigitalInputRam digitalin[number/8 + 1];\
#define\
CreateDigitalIn(name,port,mask,debouncetime,position,rammask,message,default)\
extern DigitalInputRom const name;\
#endif
```

LISTING 15 Sampling discrete inputs using IPC routines

```
void SampleDiscreteInput(IOHANDLE name)
{
  BOOLEAN result ;
  U8 RawDigital = *(((DigitalInputRom *)name)->port &
 (((DigitalInputRom *)name)->mask;
  result = DebounceDiscrete(RawDigital, name);
  SendMessage(((DigitalInputRom *)name)->message,&result;
}
```

LISTING 16 Discrete output driver routine

```
void WriteDiscreteOutput(IOHANDLE* names)
{
  BOOLEAN result;
while(*names != NULL)
  {

    ReceiveMessage(((DigitalOutputRom *)*names)->message,&result) ;
    Result ^= (((DigitalOutputRom *)*names->activelow);
    If(result == INACTIVE)
      ((DigitalOutputRom *)*names)->port &=
        ~((DigitalOutputRom *)*names)->mask;
    else
      ((DigitalOutputRom *)*names)->port |=
        ((DigitalOutputRom *)*names)->mask;
    names++;
  }
}
```

An additional routine must be created to enable output. Since `SendMessage` only enters the information into the message RAM, a driver routine must be created to actually output the data to a port. This can be seen in Listing 16. With OSEK/VDX, this routine can be called whenever an output changes. This routine navigates a null terminated list of outputs that are passed in names. It receives the current message and sets the output at the port based on the state after inverting the message value for active low outputs.

Message advantages and disadvantages

The message method clearly abstracts the source of the data, whether it is a true hardware input or output, or data received over a network. Complex messages such as structures or strings can be received easily, whereas the API calls work best if all data is of a simple type (that is, U8, U16, float, and so on).

The disadvantage with this method is that more RAM, ROM, and throughput is utilized by adding another level of abstraction: the message.

Software reuse is a noble goal that has been chased by many firms for years. In reality, attaining a large level of true reuse is usually difficult and seldom achieved. Major impediments to reuse include different microcontrollers, multiple microcontrollers on a single project, hardware changes, and networks. Changing the source of data usually forces extensive changes in code. Following the tips just outlined will help increase your reuse opportunities. **esp**

Joe Lemieux is a 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. He can be reached by e-mail at joe.lemieux@eds.com.

This little PDA went to market.

This one stayed home.

Why do some platforms and devices succeed where others fail? **Applications.** Palm™ used CodeWarrior to create the Palm OS®. Now more than **50,000 developers use CodeWarrior** to build a full range of applications for business and personal uses. Result? The Palm OS is the standard for handheld computing, and devices built on the platform are market leaders.

Want to make your platform a success? **Build it with CodeWarrior.** Get leading-edge tools and a world-wide community of more than 200,000 developers supporting your platform.

Learn how CodeWarrior can help your platform dominate the market.

Visit www.metrowerks.com.

CodeWarrior®

metrowerks®

Software Starts Here

Any other way of developing software is MANUAL LABOR.

Why slave over writing, validating, debugging and documenting your code...when you can actually enjoy designing it?

We feel your pain. That's why we created Rhapsody®. It's the only visual UML compliant real-time application software development environment that simultaneously integrates and automates analysis, design, implementation and test.

Our exclusive Model-Code Associativity guarantees that your model and code are always in sync. Change your model, your code changes. Or–you won't believe this–change your code, your model changes. And you won't even break a sweat.

Frankly, Rhapsody is amazing. It automatically generates readable, deployable, production-quality C, C++ and Java®. Not just code frames, but ALL the code. You can say goodbye to the tedium of manually developing makefiles, state machines, communication infrastructures and the like. You can also move from one RTOS to another with the push of a button. Yes!

With Rhapsody's unique design-level debugging you can visualize, easily detect and instantly correct logic and programming errors. In fact, you can actually test your application as you build it. And, wonder of wonders, you can import and reuse your legacy code. How's that for flexible?

Rhapsody users are already reducing their development cycle by at least 30%. C'mon, what are you waiting for?

Rhapsody will change the way you work. Forever.

Visit us at www.ilogix.com to download a free copy of Rhapsody.

Rhapsody®

I-Logix®

GREGORY EAKMAN

Strategies for Debugging Embedded Systems

The best time to detect bugs is early in the development process. If you instrument your UML, you can even find them during analysis and design.

Integration and testing of software is difficult, and embedded systems provide the additional challenges of limited manipulation and visibility of the system through a small number of inputs and outputs. Abnormal system states, in particular, are difficult to test, because the system must be driven into the state before its behavior in that state can be determined.

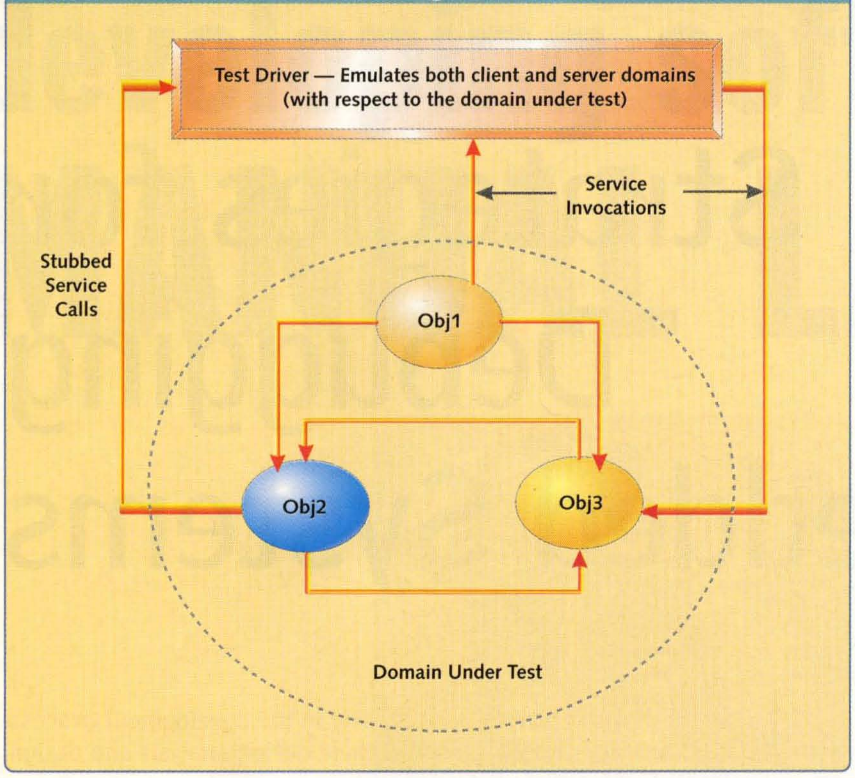
This article introduces the idea of instrumentation code injected into the implementation of UML models for the purposes of increasing the controllability, observability, and testability of the system. The instrumentation is used in both the development and the target environments, and allows interactive system debugging at the model level. In batch mode, the instrumentation serves as the basis for data collection, initialization, and test automation. My goals are to:

- Provide a brief overview of model-based software engineering and implementation of these models[1]
- Outline approaches for integration testing of model-based software
- Identify the interesting run-time data and execution points within modeled systems
- Define alternatives for collecting and manipulating model data at runtime
- Integrate the instrumentation with test automation

Integration testing

According to Roger S. Pressman, in *Software Engineering—A Practitioner's Approach*, "Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover the errors associated with

FIGURE 1 Emulation of domain's target environment


Test Driver — Emulates both client and server domains (with respect to the domain under test)
Service Invocations
Stubbed Service Calls
Obj1
Obj2
Obj3
Domain Under Test

interfacing."[2] UML models and object-oriented software tend to have classes with many complex interactions, which hinder integration testing. Combining a structured approach to UML analysis modeling with a coherent integration and test strategy will make developing quality embedded systems easier.

A software fault is an erroneous instruction or computation within a program. Execution of that fault results in an error in the state of the software. When the error is propagated to the output, and becomes visible outside the system as an unexpected result, a failure has occurred. *Controllability* of a program is the ability of a suite of test cases to force the program under test to follow a particular execution path, possibly executing faults along the way. *Observability* of a program is the ability of the test suite to detect an error state, and thereby illuminate the existence of a fault.

The internal state of the system is important in determining the correctness of tests. The output of a system is dependent upon both the initial state of the system and the inputs applied to it. The same set of inputs applied to a different initial state will result in different outputs. The final state of the system must also be examined as part of evaluating the correctness of the test, as an incorrect internal state may evntually propagate to the system output, causing a failure. System complexity may also make it difficult to predict the correct outputs of the system.

Initial State + Inputs —> Final State + Outputs

Using black box test techniques, only the external inputs and outputs of the system are available. A distinguishing sequence of test stimuli is required to propagate an error to the output so as to distinguish a faulty program from a correct one. The longer the required distinguishing sequence, the less testable the program. Embedded systems are similar to black boxes in that controllability and observability are usually limited. Evaluating the final internal state of the system results reduces the distinguishing sequence of inputs required to detect an error, resulting in smaller, more manageable test cases. Instrumentation seeks to increase both controllability and observability in a software program to result in a more testable program.

The technique of using test support instrumentation in application code is a glass box approach to testing. In developing the UML models of the system, developers express an understanding of what the system is supposed to do. Instrumentation-based fault isolation strategies help leverage the knowledge in the UML models into integration testing. The operation and state of the system are more visible at the analysis level than at the code level, where it is obscured by implementation details.

Setting the initial system state for a test only from the external inputs requires some specific sequence of external stimulus. System operation under abnormal conditions is critical to verify in many embedded applications, but creating these initial conditions may not be simple. The techniques described here enable the creation of a test harness to greatly improve controllability and observability.

Phases of integration testing

Integration testing is broken into two main phases, *dynamic verification* and *target integration.* Dynamic verification is the execution of UML models in the development environment. It focuses on determining the correctness of the models. Target integration involves software and hardware integration in the target environment. Both dynamic verification and target integration are done at the analysis level, with the same tools, using the test support instrumentation.

There are many reasons to do as much dynamic verification testing as possible: hardware availability, hardware/software isolation, shorter debugging cycle times, and access to tools. If you have high confidence in your mod-

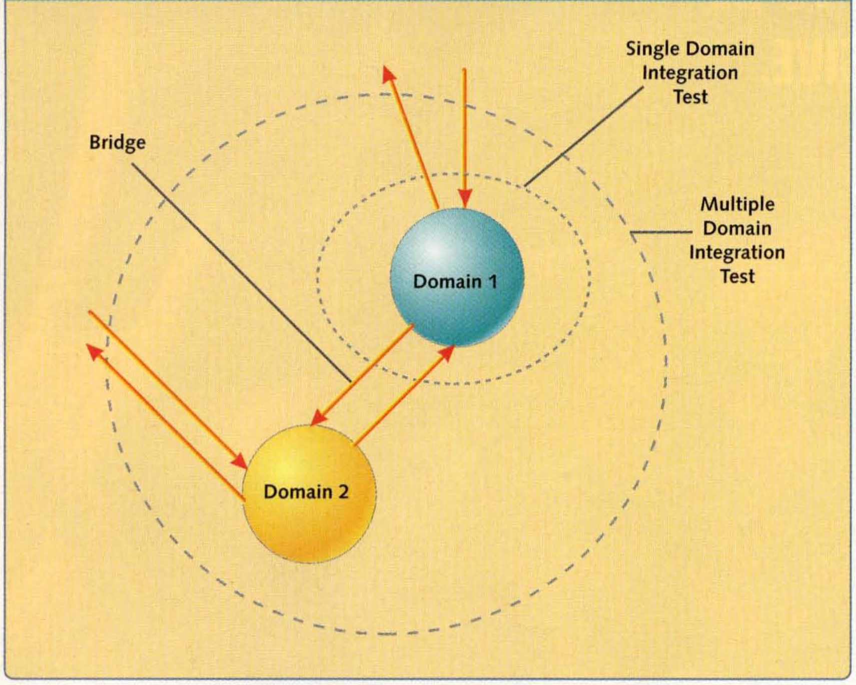
Finding defects is a painful procedure. Eliminate the pain with InstantQA.SM It identifies critical defects in C and C++ programs like NULL pointer dereferences, memory leaks, and array bounds violations. Using abstract interpretation technology, InstantQA exhaustively analyzes your software – yet requires no test cases or execution environment. It reduces testing and debugging time by pinpointing defects to specific source lines. As a service, it provides a comprehensive evaluation in less than a week without burdening developers or testers. InstantQA supports Windows, Unix, and popular embedded operating systems. Get the full story on InstantQA plus a FREE copy of our Guide to Building Great Software!

Visit www.reasoning.com/embsys.html or call 1-888-429-4111 for more information.

REASONING

HELPING THE WORLD BUILD GREAT SOFTWARE TM

FIGURE 2 Multi-domain testing


Single Domain Integration Test
Bridge
Multiple Domain Integration Test
Domain 1
Domain 2

KEIL SOFTWARE

C51 Version 6

Attention Engineers...

Keil Software C51 is the leading C compiler development suite for the 8051 family of microcontrollers. The features in Version 6 help you write and test your embedded applications faster. Call us to get the latest CD-ROM and **FREE** evaluation tools.

Keil C51 Benefits

- µVision2 IDE & debugger speed software development and application testing
- Web-based updates keep your tools current
- Training at our facility or yours shortens the learning curve
- Answers to over 1,000 questions are available around the clock on our web site

Upgrade Today...

New Features in V6

- 32-bit programs work with Windows 95/98/NT/2000 and support long file names
- Three new optimizer levels help shrink program size up to 25%
- Integrated source browser
- Complete device database sets all compiler, assembler, and linker options for you
- Kernel-aware debugging

www.keil.com

Keil Software, Inc.
1501 10th Street, Suite 110
Plano, TX 75074

Toll-Free 800-348-8051
Phone 972-312-1107
FAX...................972-312-1159

els after running tests in dynamic verification, debugging in target integration can focus more heavily on the interfaces between system components and on platform-specific issues.

Modeling embedded systems with UML

The effective application of UML models to software engineering for challenging applications—especially in the embedded context—requires a development process that will ensure:

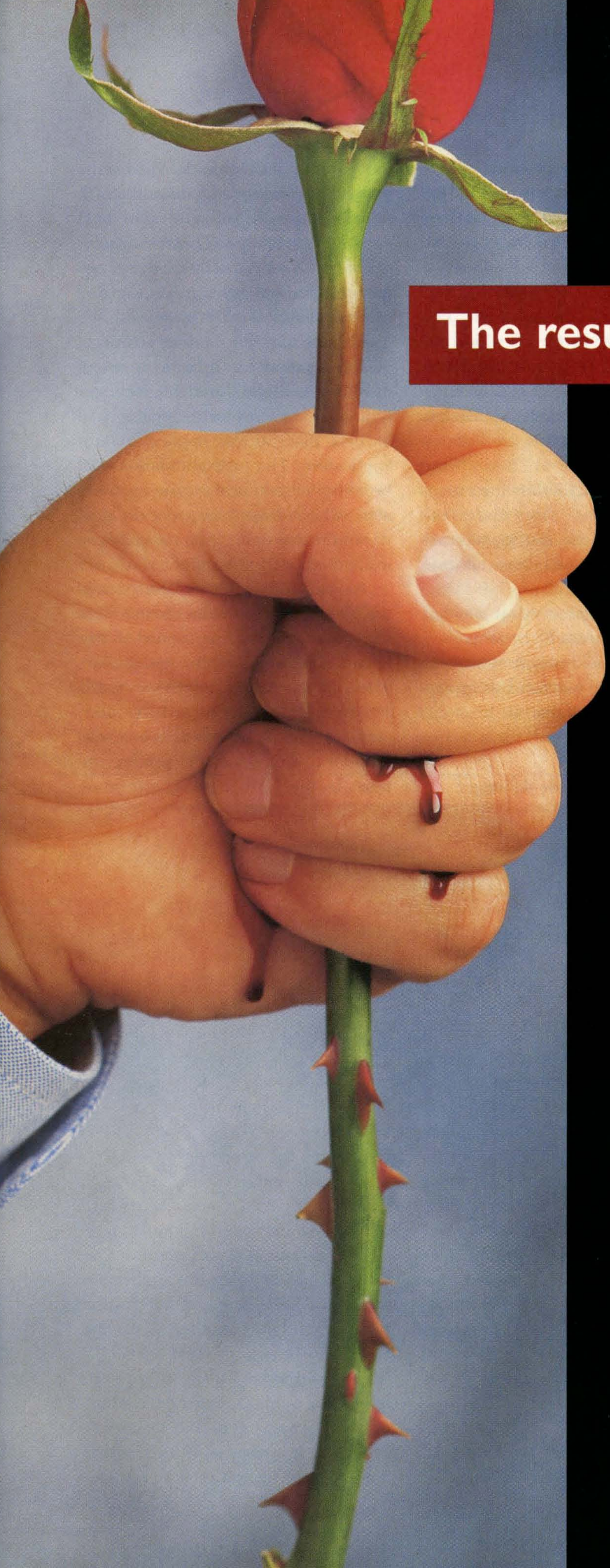
- Models are rigorous and complete
- The resulting system implementation can be optimized without impacting the models
- The overall architecture of the system is maintained by the process through multiple releases and requirement evolution

To achieve these goals, model-based software engineering employs a translational approach, defined below. This article focuses on adding test support into code using a translational approach, but the techniques can also be applied to manually implemented UML models. Specific aspects of this type of the translational process are introduced in the following section.

Analysis

The process of modeling an implementation-independent solution to a problem in terms of the problem itself is called *analysis*. Effective analysis models are rigorous and complete, and largely free of implementation bias. The Unified Modeling Language (UML) is a standard notation defined by the OMG for expressing analysis models.[3] The work products produced during analysis are:

- *Domain model*: this is a UML class diagram showing the highest level decomposition of the system into areas of separate subject matter, called *domains*. These domains are represented as *packages*, and dependency arrows show *bridges*, which are



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

the flow of requirements between domains. A domain can be analyzed, or it can be developed via other means, such as hand-written code, legacy code, generated from another source, imported from a library, and so on. Domain services are methods that make up the interface of the domain. Since the domains define a complete specification of a single problem space, they can be independently tested, then combined with other domains for further testing

- *Information model*: for each domain that is to be analyzed, a UML class diagram is used to define the classes that form the structure of the domain. Classes have associations with other classes, and inherit from other classes
- *Scenario model*: key scenarios for this specific domain are captured with UML sequence charts and/or UML collaboration diagrams to show interactions between domain services (operations), class services (methods), class events messages, and services of outside domains used in this domain
- *State model*: for each class that receives event messages, a UML state diagram is used to capture the class lifecycle, defining state-dependent behavior for that class
- *Action model*: for each domain service, class service, and state action, a detailed, unambiguous behavioral description is created. This is expressed in an action language, an analysis-level "programming" language that provides a complete set of analysis-level execution primitives without biasing the implementation. By expressing behavioral detail in action language, considerable freedom is retained until the translation phase for how each analysis primitive is implemented, which is critical for optimization

Design

Design is the creation of a strategy and mechanisms supporting the mapping of analysis constructs to a run-time environment. Design is conducted in a different concept space from analysis, and much of the preliminary design work can be completed independent of the analysis activities.

Translation

Translation is the process in which the UML models for each analyzed domain are mapped to implementation through design strategies. Design is conducted at two levels:

- *Structural design*: identify the execution units (threads/tasks/processes) of the system, allocate them to processors, and allocate domains to the units
- *Mechanical design*: develop detailed patterns (expressed in templates) to map analysis to implementation and build base mechanisms to support this implementation

Instrumentation

Parallels to source code debugger

Since the UML models represent a complete executable model of the system, the models can be translated into implementation automatically. A set of translation rules is applied to the model, much like a compiler translates high-level programming languages.

Following the language and model compiler analogy further, a model compiler can add instrumentation into the generated code, just as a language compiler adds a symbol table and debug information into the executable. Instrumentation from both compilers allows the resulting application to be tested and debugged by the developer at the same level of abstraction as it was developed. Only in very rare cases would a high-level language developer want to look at the assembly or machine code when debugging an application. Similarly, for UML models, the developer will want to debug at the higher level of abstraction of the models, rather than with the implementation code.

The translational approach uses information from the UML models to create code to support testing in addition to the application code. The instrumentation does not add any additional functionality to the software, other than enhanced testability. The test instrumentation is only available for test support and cannot be used during the normal operation of the software.

Because the instrumentation injected during model translation is based only on the UML model execution semantics, it provides a generic test harness that can be applied to any application. The instrumentation can be compiled out, or the code regenerated without the instrumentation, similar to the way debugging information is handled by the compiler.

For manually implemented systems, the level of instrumentation required is dependent on the complexity of the application, the test approach, the target environment, available memory, the support of other tools, and the time available. Thus, trade-offs must be made in order to deliver a quality product on time. In the UML models, one must identify important data values, attributes, inputs, and control points. For each of these items, add the appropriate instrumentation access. The remainder of this article will describe adding instrumentation using the translational approach, but the same principles can be applied to manual implementation.

Instrumented application architecture

The UML test architecture is broken up into two components, the dynamic verification user interface (DVUI) and the instrumentation agent. The DVUI is responsible for displaying information to the user and accepting user commands. The DVUI could be replaced with a batch interface for automation.

The instrumentation agent acts as the interface between the application and the DVUI. The communication mechanism between the agent and DVUI can be any protocol—TCP/IP, RS-232, and so on. The agent supports information marshaling and communication via generated instrumentation code. It interfaces with the instrumentation to set and retrieve instance data and to support DVUI notification of break and trace points, and interfaces with the event processing to provide execution control, system stimulus, break points, and stepping.

Data instrumentation code is injected into the application during translation to monitor and update object instance population, attribute values, event queue population, event data item values, and service parameters.

Data access

During integration testing and debugging, assessing the system state is a key requirement. Since the system state is distributed among many class instances, all instances in the system as well as the values of the attributes should be accessible to the DVUI.

Keeping a list of instances in the system is fairly straightforward. Each class must include some structure to hold the instances of that class, commonly some form of a linked list. Within the constructor, the new instance is added to the list, and in the destructor, it is removed.

To view the state of an instance, the instance can support ASCII serialization, similar to the Java `toString()` method. The `toString()` method would put the value of each of the instance's attributes into the string, as well as the instances with which it has associations. The string can then be transported from the application agent to the DVUI for display. To set the data of an instance, a `fromString()` method must be supported that can unpack the data and make the proper conversions and attribute assignments.

The state of an instance's state machine is also very interesting during debugging, because it represents an aspect of the control flow of the system. It is special because it affects the response of an instance to an event. Therefore, it's useful to separate the current state of the state machine from the rest of the class attributes.

Dynamic behavior

During system execution, instances are created and deleted, and events are exchanged; associations are created and deleted; domain services are invoked; and timers are set and fired. Each of these incidents is of potential interest during integration testing. Depending on the implementation, each of these incidents could trigger an interactive break point or a trace output.

The DVUI should be able to send to the agent the conditions on which to break. The break allows the DVUI to interactively browse the state of the system at that point in execution and adjust instances and attributes, if necessary. Trace points can also be set up by the DVUI. At a trace point, a mes-

AWT
Object Streaming
Floating Point
Multi-threading
JIT compiler

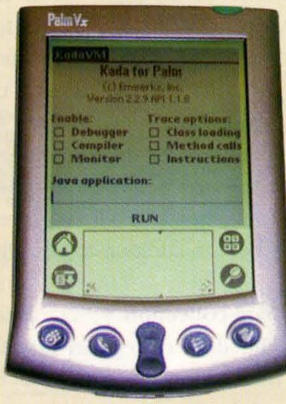
Full Function Java
...for the wireless Palm

KADA™ For Palm®
...for the wireless workspace

Kada brings Java™ applications to the Palm V PalmVII and Palm IIIe, for powerful mobile workforce applications.

The core of the Kada platform is the KadaVM, which offers the functionality of a Java Virtual Machine in a subcompact size.

KadaVM eliminates PalmOS memory limitations by providing up to 6 MB of Java heap and stack space.



Kada fully supports awt, applet, io, util, net, text and sql packages including Object Streaming, Floating point, Class Reflection and Multi-threading capabilities.

KadaVM has a built-in adaptive just-in-time compiler that speeds up execution without code bloat typically associated with JIT compilers.

em • werks
www.emwerks.com
Call 1-877-NOW-JAVA for evaluation software

KADA is property of Emwerks, Inc. All other trademarks are property of their respective owners.

sage describing the triggering incident is passed to the DVUI and stored in a log file. Trace points are a great way to follow system execution at a high level. They can even be post-processed to generate a sequence diagram describing the scenario executed, or they can be used in regression testing.

Break and trace point controls can be implemented using the publish-subscribe pattern. The agent supports registration of each type of application execution incident. The DVUI can then subscribe to incidents, such as the creation of a particular instance or initiation of a state machine transition. Instrumentation within the application code notifies the agent when an incident occurs, and the agent then notifies the subscriber with the appropriate action, break, or trace.

One implementation of a system modeled with communicating state machines includes an event queue. Events sent by objects and from outside the domain are placed on the queue. An event loop executes continuously, pulling the next event off the queue and passing it to its destination instance, which executes the specified action. With this implementation, the event loop serves as a central place to monitor system execution.

Within the event loop, an instrumentation interface is added that watches the events going by and looks up the next state to be executed by the receiving instance. These conditions can be compared against the set of break and trace conditions set by the DVUI.

The instrumentation in the event loop could also support a single event step mode. In this mode, the DVUI would command the application to execute one action, but stop before the next.

If an analysis-level action language is used to describe the UML model actions and services, you can expand the idea of event stepping to stepping of action language by instrumenting the code in a similar way. Every line of action language would be preceded by an instrumentation instruction, keeping track of the line number and local variable values. Of course, this would be tedious and error prone if done manually, but is easily accomplished by translation.

Test execution

This section describes different ways to use the instrumentation in the application to make testing and fault isolation easier.

Initialization

The instrumentation can be used to set the system to a known initial state. By creating and initializing class instances through the agent, the state can be set directly by the test case. This makes testing from hard-to-reach states easier, rather than having to apply a sequence of inputs to drive it to that state. The initialization is made possible by serializing instance data across the communications channel to the agent, where the new instance is created and initialized.

A soft reset capability would allow the system to be cleared, reinitialized with another initial state, and run with another set of test cases.

Stimulus

Stimulus is applied to the system from the DVUI through the instrumentation agent. Events can be initiated by the DVUI and delivered to a target instance through the instrumentation agent and event queue. An encoding scheme for the event and the data must be derived. Also, with some additional encoding, domain services can be invoked as system stimuli as well. This mechanism is similar to the encoding of parameters that CORBA and RPC use when calling functions across processes.

One of the more difficult problems in embedded systems tends to be reproducibility of failures due to timing or sequencing of events. By controlling the sequencing of event transmission through the event queue, orderings of actions can be tested and reproduced. Of course, the interface to the event queue would need to allow the reordering of events, in addition to viewing the events.

Data collection

The data collection interface can be used to test pre- and post-conditions of test cases. This is especially useful for determining and validating the final state of the system as part of test result evaluation.

Another side benefit of instrumentation is the ability to capture true target stimuli and replay it in the dynamic verification environment. If some part of the instrumentation were left in the finished product, it could record a subset of the system state and inputs, much like the "black box" on airplanes.

Emulation

In order to effectively test a domain in isolation during dynamic verification, the interface at the domain boundary must be well understood. The test cases that are defined for the domain will generally use this interface as the primary stimulus to the domain. The test cases and stimulus data are obviously application-specific, but use the test harness already provided by the instrumentation.

Figure 1 shows a test driver, either the DVUI or other program connected to the instrumentation agent, emulating the domain's target environment. The driver initializes the class instances that are part of the test. The test driver applies the test stimuli and captures the response. The test driver also emulates the responses of other domains by trapping the service calls and substituting the return values. The calls are a form of output from the point of view of the domain under test, and the responses provided by the test driver provide more input to the domain under test. The test driver uses the test instrumentation in the domain under test to trap and substitute messages.

Single domain to system testing

This test approach is scaleable from one domain to the integration of multiple domains and into system test (Figure 2). Single domains are first tested in isolation, using the test driver to emulate the environment of the domain under test. Domains can then be combined for testing through integration of domain services. Again, the test driver emulates the environment of the domains under test. The assumptions that one domain makes about another, in the form of service calls, can now be verified. The interface and flow of data across the test boundary should be well understood.

As confidence is gained in lower level domains, the test support instrumentation can be disabled on those domains to reduce the number of checkpoints and increase throughput. If a problem is found, the test support can be re-enabled to collect more data on the specific test case.

Leveraging potential

Instrumentation added to the implementation of UML models allows the models to be leveraged into the testing phase of development. Developers can write test cases, execute, and debug and the level of the models. Ideally, instrumentation should provide a full debug environment at the level of analysis. Full interactive manipulation of instances, events, and data should be available, as well as ways to actively stimulate the system, rather than just watch it. The instrumentation provides full access to the system under test at the level of UML modeling, allowing a glass box approach to integration testing with greater observability, controllability, and testability for embedded systems. Increased observability enabled by instrumentation also allows detection of internal error states without having to propagate the error to the output. This results in asier test case development and shorter debugging time, as the error is isolated much closer to the fault.

Once the capability of the full interactive analysis debugger exists, there is an even greater potential for improved quality and productivity by adding a batch capability to it. This batch capability would allow automated regression testing, automated data collection of failure test cases, or application of randomized test inputs and event sequences to obtain more analysis coverage. **esp**

Gregory Eakman is a principle consultant with Pathfinder Solutions and is a PhD candidate at Boston University in the area of automated testing of model-based software. He has successfully applied model-based software engineering to a varity of applications. Contact him at grege@pathfindersol.com.

References

1. Pathfinder Solutions, "MBSE Software Engineering Process," February 2000: *www.pathfindersol.com/download.html*.
2. Pressman, Roger S. *Software Engineering—A Practitioner's Approach, 3rd Edition*. New York: McGraw-Hill, 1992.
3. Object Management Group, "The Unified Modeling Language" Specfication, November 1999: *www.omg.org*.

Free Onboard TCP/IP on this Feature-rich Embedded PC

- Embedded Netsock™ TCP/IP stack
- 10BASE-T Ethernet
- TTL and analog I/O
- Turbo XT, 25MHz
- Free DOS compatible OS
- PC/104 format

Call or visit our website for the Netsock/100 data sheet or a FREE 284-page Handbook

MICRO/SYS
Glendale, CA
(818) 244-4600 • Fax: (818) 244-4246

Gotta Duplicate Your Memory Cards?

Get a CARD DUPER™ today!

CARD DUPER™ is the only memory card duplication system designed specifically for end-users.

CARD DUPER™ supports Compact Flash, PCMCIA, Smart Media, MMC, Memory Stick, Linear, SRAM, OTP, ATA, and Epson 40-pin (SRAM, Flash)

Contact us today at www.cardduper.com, before you drive yourself crazy trying to program memory cards the old fashioned way.

International Microsystems, Inc.
Milpitas, CA • (408) 942-1001
www.CardDuper.com

IAR Embedded Highway Get There Faster!

IAR Embedded Workbench™

A superior C cross compiler package offering advanced embedded C/C++ optimization techniques

- Supports the following MCUs:

Atmel	AVR, AT89
Hitachi	SH, H8S, H8/300H, H8/300
Intel	8051, 80x96, 80251
Microchip	PICmicro 16/17 and 18
Mitsubishi	M16C/60/20, MC80, 740
Motorola	6811, 6812, 6816
National	COP8, CR16
NEC	V850, 78K0, 78K4
Sharp	SM6000, SM8500
TI	MSP430
Zilog	Z80, Z180

and more... for complete list, check www.iar.com

- Different Architectures, One Solution: The same user friendly environment applies for all targets.

IAR visualSTATE®

A UML-compliant graphical state machine tool generating micro-tight code for any embedded MCU.

- Very small code footprint (A break-through patented technology)
- Table driven code (compact & fast)
- Uniquely suited for low-end 8/16-bit MCUs
- Generates MCU & Compiler independent code (works for 8/16/32/64-bit MCUs)
- 100% formal verification of state model (a second patent)
- Allows state level debugging, breakpoint setting, variable/action watch and graphical animation of model
- Static & dynamic analysis with regression testing
- Automatically generates complete design documentation in 'rtf' format.

IAR MakeApp™

The Device Driver Wizard. Automatically generates fully tested C initialization code for MCUs

- Visual configuration of device drivers.
- Supports the following targets

Atmel	AVR
Hitachi	SH, H8S/300H, H8/300
Mitsubishi	M16C
Toshiba	TX RISC

for up-to-date target support check www.iar.com

- Checks for illegal settings & resource conflicts

IAR App. Development Services

- Contact IAR to speed your next embedded development project.

For a free demo & more information, please contact IAR Systems.
Email: info@iar.com, Tel: 1-800-427-8868 Or visit **www.iar.com**

IAR Systems (US HQ & West Coast), One Maritime Plaza, San Francisco, CA 94111 Tel: 415-765-5500, Fax 415-765-5503
IAR Systems (East Coast) 2 Mount Royal, Marlborough, MA, 01752, Tel: 508-485-2692, Fax 508-485-9126

IAR SYSTEMS
From idea to target.
www.iar.com

Don Morgan

Hijacked!

Reader questions prompt a more basic treatment of PWM and motor internals.

Since I started this series, quite a lot of mail has come in with comments and questions concerning motors, PID, motion control, field-oriented control, and pulse width modulation (PWM). I've recieved enough questions, in fact, that I decided to review some of these areas before moving on. I'm going to answer some of the questions by presenting the data that we've covered from a different perspective. I'll also provide additional illustrations.

Specifically, we'll take another look at the physical aspects of controlling brush and brushless permanent magnet motors. Our coverage will also include PWM and a little bit on phase currents. Next month, I'll go into greater detail on the PID algorithms and coefficients. Following that, we will again continue with algorithms on resolver and sinusoidal encoder conversions, which was my initial plan.

PWM

Pulse width modulation is a digital technique for the control of DC supplies to provide varying voltages into a load. I use the word digital because with PWM the full DC supply is either turned on to the load or turned off. That is, power is supplied to the load by means of a series of on and off pulses. The on-time is the period during which the DC supply is placed on the load and the off-time is the period during which that supply is removed, or cut off, from the load.

In this manner, we can control the average voltage delivered to a load. For example, If we have a switch between a light bulb and a 9V battery and we turn the switch on and off at half-second intervals, the bulb experiences an average voltage of 4.5V. This process is illustrated in Figure 1. To describe this situation, we say that the duty cycle is 50% and the frequency is 1Hz. Most loads, inductive or capacitative, require a higher frequency to achieve tighter control and resolution. Common frequencies for motion control amplifiers range from 4kHz to 20kHz. Whatever the frequency, a 9V supply with a 50% duty cycle will always produce an average voltage of 4.5V.

Brush permanent magnet motors

Like brushless motors, brush motors have a rotor (the part that rotates) and a stator (the stationary part). Unlike brushless motors however, the magnets (or field) are mounted on the stator, while the windings (armature) are located on the rotor.

The armature of a brush motor has many windings, or phases, with only one portion excited at any one time, depending on the position of the rotor relative to the stator. Brushes, acting as switches, commutate the motor by connecting current to different windings as the rotor turns. This results in two flux patterns. One generated by the permanent magnets in the stator (field) and the other generated by the armature windings.

Electromagnetic torque is created by the interaction of the field flux and the armature flux. To motor, or produce torque, the electrical angle between the field and armature is typically 90 degrees (about 45 mechanical degrees). So on a brush permanent magnet motor, the brushes are arranged so that current-flow in the armature windings produces flux that leads the stator flux by 90 degrees. So many phases are possible in such an arrangement that each one represents only a few electrical degrees of rotation, making for mostly ripple-free torque.

The relationship between current and torque on a permanent magnet machine is linear. It's possible to create a current source for such a motor, but most often the control outputs a voltage to the motor windings.

This works because the loads we are dealing with in motion control are overwhelmingly inductive. In an inductor, current begins to flow after voltage is applied, and, depending upon the amount of inductance, that current will rise to a certain level in a

FIGURE 1 A PWM-operated lamp

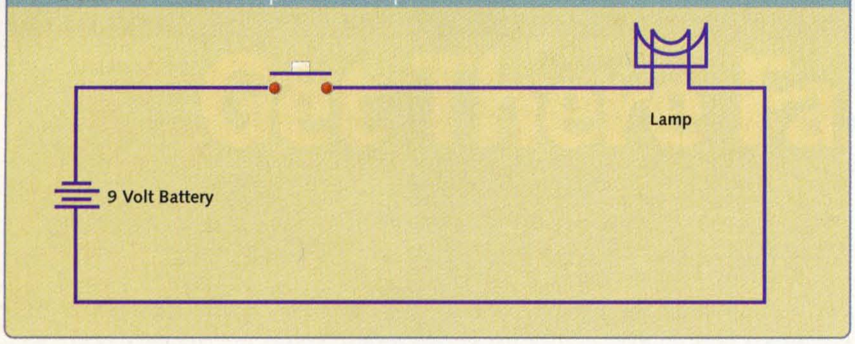


FIGURE 2 Bridge arrangement for controlling brush permanent magnet

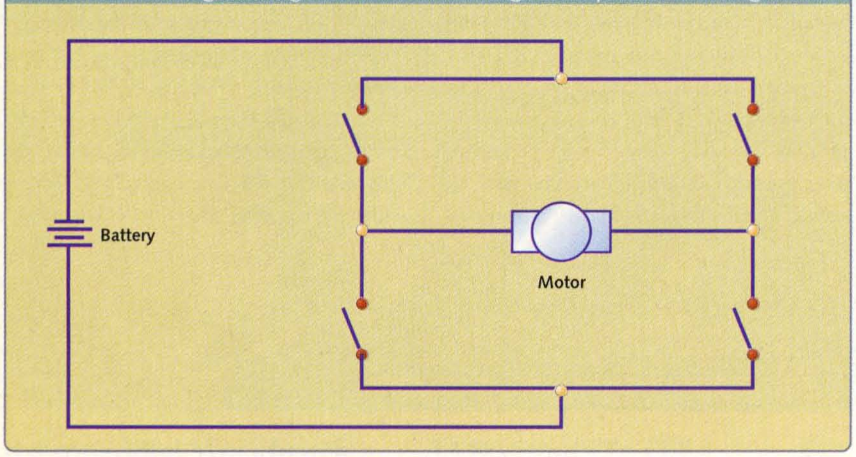
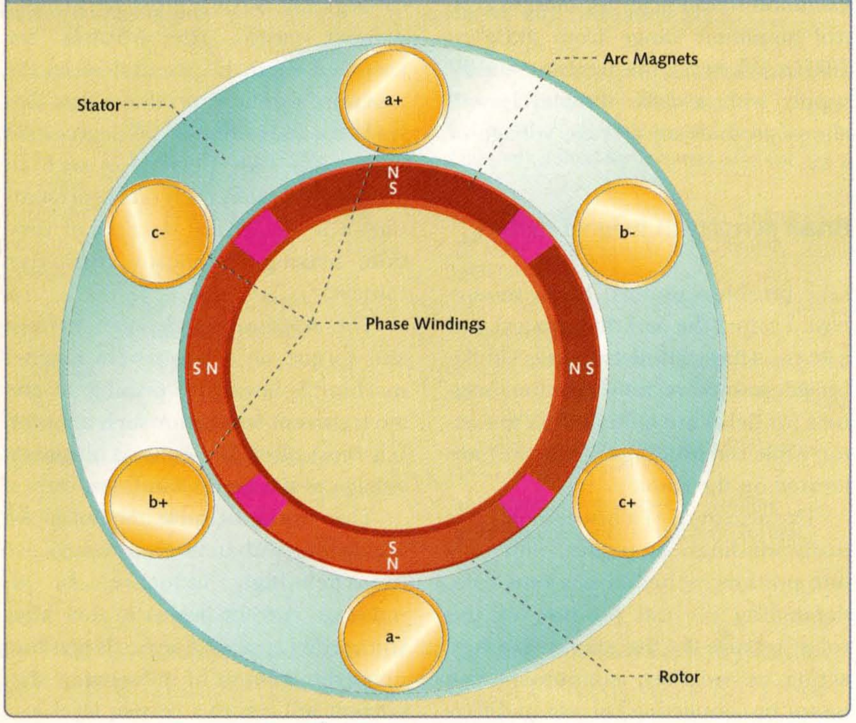


FIGURE 3 Brushless three-phase motor



given period of time. The greater the inductance, the less total current will flow in a given period of time.

Techniques for voltage control abound, but PWM is the most frequently used. PWM controls the average voltage to the windings, as with the lamp in Figure 1, and, therefore, the average current to the windings. In a brush permanent magnet motor, torque is a linear function of current.

In Figure 2, we see a typical brush permanent magnet motor driver. It's often called an H-bridge because of the arrangement of the switches and the motor. The four switches in the diagram control both the direction of current through the motor, and, because they are pulse width modulated, the average voltage across the motor. The one important thing to remember is that one never has an upper and lower switch closed in the same leg at the same time. This would result in a short circuit.

When we use a brush permanent magnet motor, we can control the voltage across the motor (and therefore the current through the motor and its torque) with PWM. We control the commutation with brushes.

Brushless motors

Brushless permanent magnet motors differ from brush motors in that the windings are located on the stator and do not move, while the magnets are found on the rotor. The windings are distributed in multiple phases, usually three. Each phase is separated by 120 electrical degrees. One electrical connection exists to each phase. Brush motors switch phases in and out mechanically, and because the device features numerous phases, the motion is relatively smooth. A similar technique is possible on a brushless motor, but because there are only three phases, this technique produces large torque perturbations at each transition.

Therefore, on a brushless motor, the commutation is done electronically. The drive, or power stage, monitors the position of the rotor, usually with an encoder, and excites the appropriate winding to maintain a 90 degree commutation angle.

Figure 3 shows a cross-section of a four-pole brushless motor. If you imagine the rotor making a complete mechanical revolution of 360 degrees, you will notice that the flux from the magnets cycles twice, once for each north/south pair. In other words, there are more electrical cycles than mechanical cycles. In fact, the number of electrical degrees is equal to one half the number of poles on the motor multiplied by the number of mechanical degrees. Most brushless permanent magnet motors have more than two poles.

Figure 4 is a diagram of a typical power stage. It consists of six switches configured much like the H-bridge depicted previously but with an extra leg for the third phase. Here, we also use PWM to control the average voltage to each winding, and, therefore, the current through each phase. Again, one never has an upper and lower switch on at the same time. In this diagram, the voltage that we speak of is measured from the input to each phase to the center of the Wye.

Thus, we can control the average voltage across each phase with PWM, as in the brush application. But how do we do the commutation?

Commutating a brushless permanent magnet motor

Another technique exists that is more suitable for three-phase motors than the six-step method I described last month (p. 179). It's called sinusoidal commutation. Because a brushless motor can control current in multiple phases independently, it moves the winding flux in very small increments.

To understand this, recall Kirchoff's law, which states that the sum of all currents, voltages, and fluxes in an electrical circuit must equal zero. That relationship is represented by the following formulae:

$$i_a(t) + i_b(t) + i_c(t) = 0 \qquad \text{for current}$$

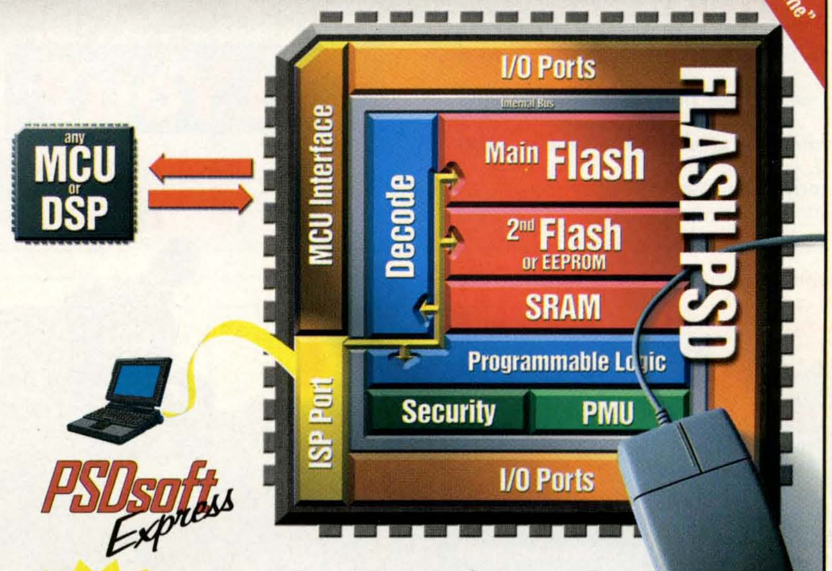
$$V_a(t) + V_b(t) + V_c(t) = 0 \qquad \text{for voltage}$$

$$\varphi_a(t) + \varphi_b(t) + \varphi_c(t) = 0 \qquad \text{for flux}$$

A simple way to understand this is to imagine a water hose. The water moving into the hose, which we'll call +water, must be equal in amount to the water coming out of the hose,

ISP/IAP Flash

"Now shipping" "Stock to 12wk lead time"



Booth #228 ESC West

Download Free software today, and configure your embedded design with your mouse!

Waferscale's **EasyFLASH™ PSDs** solve the problems faced by designers needing to integrate programmable logic along with high density Flash and SRAM into embedded designs. Unlike all other options, PSDs form an integrated system level solution that saves time, power, and costs. Waferscale's software development/programming tools... **PSDsoft Express** and the new PSDsoft 2000, provide an easy to use, point and click user interface in which an engineer can accomplish in a couple of hours what may have, in the past, taken several days. Visit our website today to learn more about EasyFLASH PSDs and to download your free copy of PSDsoft Express!

"Development Kits starting at $99 available now"

www.waferscale.com/esp/flashisp.html

WSI waferscale.com

Waferscale-USA • Tel. 800-832-6974 • Fax 510-657-5916 • info@waferscale.com which we will call –water. Therefore, if we add the +water to the –water, we'll get zero. No other waters are involved. If water goes in one end it will come out the other. A three-phase system is similar, except we have three bi-directional inputs. With a high frequency PWM we can make fine adjustments to the amount and the angle of the stator flux.

If we examine the motor from the stator point of view, we see three-phase current flux angles that change constantly. If, however, we assume a fixed position on the rotor, we can see that two fluxes must be present in a motor for it to move and produce torque, and that they will be about 90 degrees from one another. One is the rotor flux, which in a permanent magnet motor, is fixed by the circular magnets on the rotor. The other, the stator flux, is generated by the three-phase windings on the stator. These two fluxes can be mathematically generated by two currents, I_d and I_q. This separation of currents is a mathematical operation illustrated by the vector sum in Figure 5. The vector sum of the flux generated by the magnets on the armature and the flux from the windings on the stator, result in a third vector often referred to as torquing flux.

Recall that we are imagining this picture from a fixed position on the rotor. Here, the values of I_d and I_q change very slowly. You will see from the vector diagram in Figure 5 that on a permanent magnet motor, it is only necessary to manipulate the winding flux to create acceleration, deceleration, or a profile, because the flux from the permanent magnets is fixed. This means that on a brushless permanent magnet motor, the I_d may be zero.

But to control the stator flux, we need to generate phase currents; or, you could say, we want to use our power bridge to produce an average voltage across each phase that will generate the proper flux in the windings to cause the motor to turn as we choose. It can be shown (please see the bibliography) that I_q and I_d are a result of the individual phase currents and the position of the rotor as shown here:

$$\begin{bmatrix} I_d \\ I_q \end{bmatrix} = \frac{2}{3}\begin{bmatrix} \cos(\theta) & \cos(\theta-\gamma) & \cos(\theta-2\gamma) \\ -\sin(\theta) & -\sin(\theta-\gamma) & -\sin(\theta-2\gamma) \end{bmatrix}\begin{bmatrix} I_a \\ I_b \\ I_c \end{bmatrix} \tag{1}$$

This formula is the product of two formulae:

$$\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \frac{2}{3}\begin{bmatrix} 1 & \cos(\gamma) & \cos(2\gamma) \\ 0 & \sin(\gamma) & \sin(2\gamma) \end{bmatrix}\begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix}, \quad \gamma = \frac{2\pi}{3} \tag{2}$$

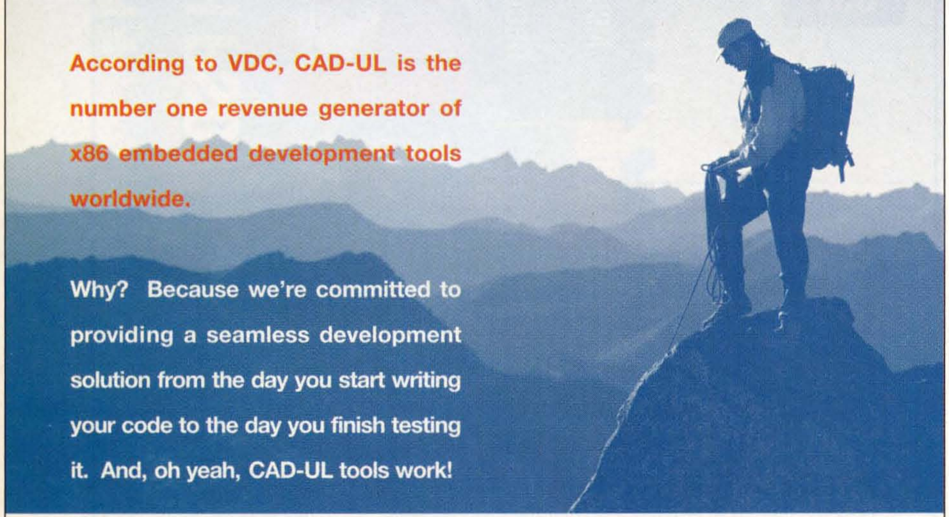
U S Software Knows Why CAD-UL is #1.

Do You?

U S Software knows who to count on when they need a partner for x86 embedded development tools. And it looks like they're not alone.

According to VDC, CAD-UL is the number one revenue generator of x86 embedded development tools worldwide.

Why? Because we're committed to providing a seamless development solution from the day you start writing your code to the day you finish testing it. And, oh yeah, CAD-UL tools work!



U S SOFTWARE. EMBEDDED EXCELLENCE

- Real-Time Operating System
- Embedded TCP/IP Protocol Suite
- Embedded File System
- Distributed Computing

www.ussw.com
(800) 356-7097

CAD-UL

- C/C++ x86 Compiler Systems
- Symbolic Debugging Systems
- IDE: CAD-UL Workbench
- Code Coverage Tools

www.cadul.com
(877) GO CAD-UL

$$\begin{bmatrix} i_d \\ i_q \end{bmatrix} = \begin{bmatrix} \cos(\theta)\sin(\theta) \\ -\sin(\theta)\cos(\theta) \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} \qquad (3)$$

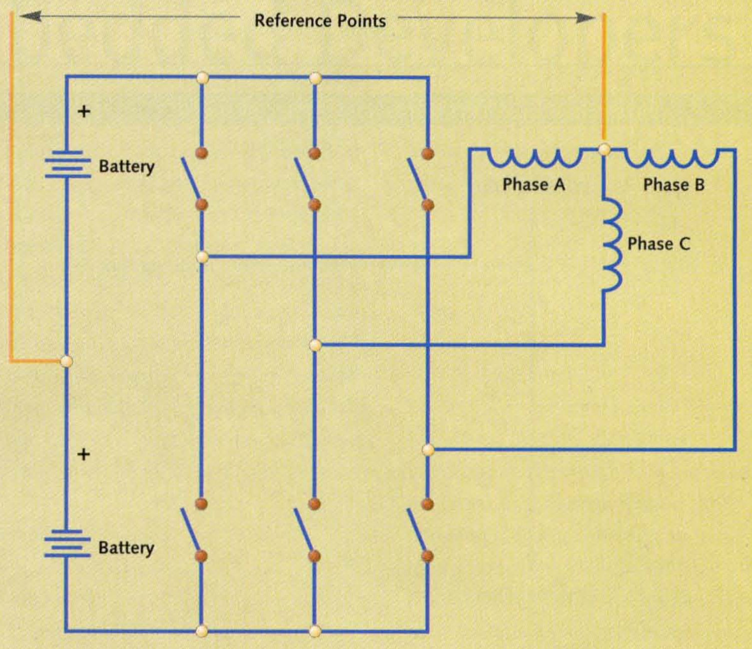
Equation 2 transforms three phase currents, i_a, i_b, and i_c, to a homopolar system, but with reference to the stator. This means the values i_α and i_β change rapidly with time. To get to the fixed position on the rotor where I_d and I_q (φ_d and φ_q) are moving very slowly, we need to make one more change, and that is a simple rotation based on the position of the rotor relative to the stator. An encoder usually supplies this angle. This transformation is given in Equation 3.

Thus, we have one of the most popular and effective algorithms for controlling permanent magnet motors.

Induction motors

We can view an induction motor much like a brushless permanent

FIGURE 4 Three-phase power bridge and Wye wound three-phase motor



32/64-Bit CPU Development Boards

- ARM
- MIPS
- Power PC
- M-Core
- Strong ARM I & II

CMA

COGENT MODULAR ARCHITECTURE

Cogent Modular Architecture (CMA) - a series of flexible development platforms offering advanced on-board I/O, PCI expansion and interchangeable CPU Modules for a variety of 32/64-Bit architectures.

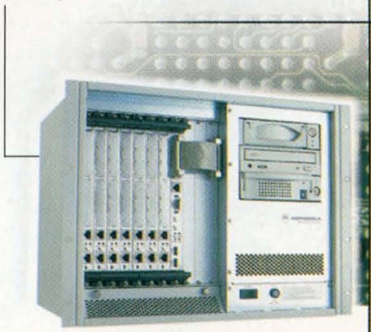
COGENT

Cogent Computer Systems, Inc.

Tel: 401-295-6505 www.cogcomp.com

Put Linux Anywhere™

LARGE...
High-Availability Servers



SMALL...
Embedded
Microcontrollers

...AND EVERYTHING IN BETWEEN:

- Smart appliances
- Modems
- Credit card readers
- Radar controls
- Industrial automation
- Mobile phones
- Thin clients

Whether your target design includes an embedded processor with a small footprint or is the next market-winning Compact PCI High Availability solution, Lineo delivers a Linux®-based solution to meet your needs.

Our uClinux™-based microcontrollers deliver sub-1 MB memory footprints and Lineo's HA Enabler delivers 99.999% reliability on High Availability Compact PCI servers.

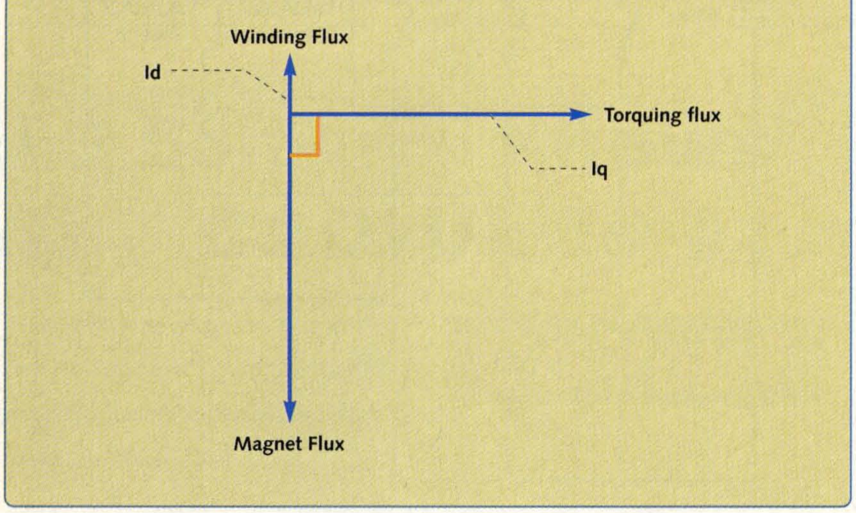
So partner with Lineo and Put Linux Anywhere.

To learn more about our tools, support and services, visit us at www.lineo.com or call 1.888.463.7367

LINEO™

magnet motor. The primary difference is that the induction motor has a wound armature instead of permanent magnets. The armature is magnetized by the stator windings; this is the I_d component in Equation 1 that must now be supplied to the motor. As a result, an induction motor requires approximately one-third more current to run than a comparable permanent magnet motor.

FIGURE 5 Fluxes in a brushless permanent magnet motor



Next month, we dig more deeply into the PID algorithm and the meaning of the terms and coefficients. **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. Don's e-mail address is dgm@baykitty.com.*

development tools

C/C++/EC++ Compilers
True Time Simulator
Real Time Debugger
I/O Simulation and Stimulation
Real Time Kernel
Peripheral Builder
BDI BDM / JTAG Interfaces

Motorola
HC05, HC08, HC11, HC12, HC16, M•CORE, M68xxx/3xx, PowerPC

Philips
8051XA

STMicroelectronics
ST7, ST16, ST19

NEW! PANTA for PowerPC with Intuitive User Interface

HIWARE Innovation for your Success is now metrowerks A Motorola Company

Metrowerks
9801 Metric Blvd
Austin, TX 78758
USA
Ph. (512) 997 5870
Fax (512) 997 4901
info@metrowerks.com

Metrowerks Europe
(formerly HIWARE)
Riehenring 175
4058 Basel
Switzerland
Ph. +41 61 690 7500
Fax +41 61 690 7501
info_europe@metrowerks.com

Metrowerks France
(formerly HIWARE France)
1 Rue de la Haye Le Dôme
95731 Roissy CDG Cedex
France
Ph. +33 1 41 51 19 72
Fax +33 1 41 51 19 73
info_europe@metrowerks.com

www.metrowerks.com
www.hiware.com

Tools for Embedded Developers

Software

Driver tools

WinDriver v. 5.0 and KernelDriver v. 5.0 are software tools that together form a driver development environment for Linux, Solaris, and Windows. WinDriver features a wizard that enables developers to detect and diagnose their hardware using graphical tools without writing any code. WinDriver also includes remote access capabilities that enable communication between the development environment and the hardware through any network connection. KernelDriver automates the development of standard drivers, layered and filter drivers, and miniport drivers by means of a wizard that generates the necessary C++ code. A free 30-day evaluation package is available now at the company's Web site.

Jungo
Natanya, Israel
(877) 514-0537
www.jungo.com

Development environment

The MULTI 2000 integrated development environment now supports every chip in Hitachi's SuperH RISC line of processors. The environment is customized to take advantage of the chip line's features. For instance, the MULTI 2000's compiler supports the SH3-DSP's ability to execute four independent operations per instruction. It also structures code around the SH3-DSP's zero-overhead looping capability. Designers can assemble and disassemble DSP code and view internal DSP registers. The MULTI 2000 is also designed to utilize the SH-4 processor's 16-bit fixed-length instructions. The MULTI 2000 IDE, when combined with a compiler, automates many aspects of the development process. It features a window-oriented editor, a source-level debugger, a graphical program builder, and a run-time error checker. It also includes a version control system, an instruction set simulator, a performance profiler, a real-time event analyzer, and a context-sensitive HTML help system. MULTI 2000 is available now. It costs $5,900 for Windows and $8,900 for UNIX.

Green Hills Software
Santa Barbara, CA
(805) 965-6044
www.ghs.com

Chip support for tool suite

TriCore Tool Suite v. 1.1 is available for Infineon Technologies' TriCore Unified Processor Core, as well as for standard products based on merged MCU/DSP architecture. The suite includes an assembler that supports the TriCore Unified Processor's peripheral control processor, floating-point unit, and memory management unit. The suite's debugger works with a cycle-accurate instruction set simulator. The suite is available now, starting at $3,900.

Tasking
Dedham, MA
(800) 458-8276
www.tasking.com

Tool family

The DriveWay Lite family is a series of tools that automates the design of board-support packages comprised of device drivers, boot code, and glue code. Products in the DriveWay Lite line support the simplified hardware requirements of evaluation boards. Processor-specific versions of DriveWay Lite are available now for $7,500.

Aisys
Santa Clara, CA
(408) 327-8820
www.aisysinc.com

Development software

Dynamic C Premier is a development environment for the Rabbit 2000 microprocessor. It features all the functionality of previous versions and now includes the MicroC/OS-II real-time kernel, as well as Fast-Fourier Transform (FFT) and TCP/IP capabilities. FFT transforms a waveform into its constituent frequencies. With Dynamic C Premier, Rabbit 2000 processors can include pattern-recognition, filtering, and interpolation in applications for data analysis in areas ranging from earthquakes and oceanic waves to electrical distribution. When using Dynamic C Premier, developers can monitor events in real time while their program runs in a controller. It's available now for $295.

Z-World
Davis, CA
(530) 757-3737
www.zworld.com

Design documentation

Imagix 4D is a reverse engineering tool that now features software documentation capabilities. Using information located in its database and in the designer's sourcecode, it documents each software component. This includes all files, classes, functions, and variables. Imagix 4D features a control flow analysis tool that analyzes the sequences and conditions of function calls and variable usages in the code.

I stopped at a yardsale this afternoon and found a terrific packet-voice processing engine.
I hope you checked to see if it's preconfigured with echo cancellation functionality.
1941
58N28

Welcome to embeddedtechnology.com. Your kind of community.

An entire community devoted to embedded computer design? In a word, yes. It's an online community brought to you by VerticalNet, a company dedicated to helping people conduct business as easily as possible. You'll have access to the latest industry news, extensive career opportunities, and a thriving marketplace where industry professionals buy and sell every product imaginable. To visit, simply go to www.embeddedtechnology.com. The resources are there for you. All you have to do is take advantage of them.

embeddedtechnology.com a VerticalNet® community VerticalNet®

Developers also have control over the representation of the data, choosing where to insert cross-references and chart graphics into the documentation.

Imagix
San Luis Obispo, CA
(805) 781-6002
www.imagix.xom

Development software for non-programmers

SoftWIRE v.. 2.0 is a tool creates programs using Visual Basic v.. 6.0. It features database, TCP/IP, arithmetic, financial, and analysis functions. Among the database functions are a query builder, a database grid, and the ability to read and write from Access, Oracle, or SQL databases. Existing features of SoftWIRE include a graphical interface for the oscilloscope, strip chart, and digital volt meter. Many of the new controls are purely graphical, which means that all properties and variables are set by modifying and coloring graphical displays. Its available now for $495.

Computer Boards
Middleboro, MA
(508) 946-5100
www.computerboards.com

Hardware

Analyzers

FireSpy400 combines an analyzer, a generator, a monitor, and a controller in a single unit. Together, these devices provide protocol analysis for IEEE1384-based systems at transaction level. In addition, FireSpy can analyze a device's registers and memory addresses. USB agent is a USB bus and protocol analyzer. It records all USB traffic non-intrusively at low or full speed. They're both available now for $3,500 each.

Hitex Development Tools
Sunnyvale, CA
(800) 454-4839
www.hitex.com

Development kit

The DK900-HC11 is a development kit for the design of 68HC11-based systems that require in-system programming during system operation (a process also called in-application programming, or IAP). Systems of this type include automotive diagnostic systems, medical monitoring systems, and remote industrial control systems. IAP requires two flash memory arrays so that boot code and flash algorithms can be executed at the same time that data is being erased or written into the system. The DK900-HC11 provides these arrays, as well as all the other hardware and software required to implement an IAP system. Specifically, it includes a 68HC-11-based development board, a FlashLINK programmer and cables, a null-modem serial cable, power supply, and a CD-ROM with PSDsoftExpress EDA tool and PSDload UART download software. The CD-ROM also includes programming examples and tutorials on JTAG-based in-system programming. The development board includes a PSD9XX programmable system device, a 16-character LCD display, power switch, UART serial port, a JTAG programming port, a reset button, a DIP switch, two LEDs, and pads for additional RAM. The DK900-HC11 development kit is available now for $99.

Waferscale Integration
Fremont, CA
(510) 656-5400
www.waferscale.com

Chips

High-end processor

The 400MHz to 500MHz UltraSPARC 64-bit processor is targeted for embedded applications in the telecommunications, network infrastructure, and ISP markets. It integrates a 256KB Level 2 cache; a 32-bit, 66MHz PCI bus; and an SRAM controller and

When putting together an **Embedded System**, remember...
A system is more than the sum of its parts.

ADS GRAPHICS CLIENT™ PLUS
Application Ready Systems at your fingertips!

- **StrongARM 206 MHz**
- **Embedded Linux**
- **WinCE 3.0**
- **Java/VxWorks/OS-9**

ADS engineers complete embedded system solutions so that you can focus on what's most important: **Your Application**

Ask about the BITSY

ADS *www.applieddata.net*
APPLIED DATA SYSTEMS, INC.
Hardware • Software • OS Integration • 301.490.4007 • 800.541.2003

Emulators, Programmers, C-Compilers and Emulation Boards

CEIBO SUPPORTS:
XA
PIC
251
ST6
X86
8051
C500
Dallas
V-Series

Ceibo USA:
1-800-833 4084
Tel: 314-830 4084
Fax: 314-830 4083
info@ceibo.com

CEIBO Development Tools of Choice

www.ceibo.com

- Online Ordering
- Electronic Catalog
- Software Updates
- Price List

Product and company names are trademarks of their respective organizations

memory interface. Power consumption is estimated to run at a maximum of eight watts for a processor running at 400MHz and 15V. Maximum power consumption is 13 watts for a processor operating at 500MHz and 1.7V. Built-in power management functions permit the chip approximately three watts maximum power usage in sleep mode. The UltraSPARC processor will be available in November. It costs $230 for the 400MHz version and $357 for the 500MHz version.

Sun Microsystems
Palo Alto, CA
(650) 856-2114
www.sun.com

Floating-point DSP

The TMS320C6712 DSP performs up to 600 million floating-point operations per second. It operates at 100MHz with a core of 1.8V and an I/O supply 3.3V. In addition, it features an internal memory architecture that can be used as a two-level cache, a one-level cache with direct-mapped memory, or a combination of two-level cache and direct mapped memory. Also included is a peripheral set that features 16-channel direct memory access, 16-bit external memory interface, two multi-channel buffered serial ports, and two 32-bit timers for data I/O and design flexibility. The C6712 is sampling now, with volume production scheduled for the first quarter of 2001. It will cost $9.95 each in quantities of 100,00 units.

Texas Instruments
Santa Clara, CA
(800) 477-8924, ext. 4500
www.ti.com

EEPROM family

A family of standard silicon EEPROMs has been made suitable for use in space. The family offers units of a total dose hardness ranging from 50 to 250 krad, depending on the customer-specified orbit in which they will travel. The products feature no Single Event Latchup and a ten-year data retention rate. They are all capable of in-system electrical byte and page programmability. The EEPROM feature data polling and a Ready/Busy signal to indicate the completion of erase and programming operations. Hardware protection is provided with a RES pin, in addition to noise protection on the WE-signal. Software data protection is implemented using a JEDEC-optional standard algorithm. Units in the family are available now. One megabyte versions cost $750, 4MB versions cost $1,750, and 8MB versions cost $2,500.

Space Electronics
San Diego, CA
(858) 279-5100
www.maxwell.com

Altera chip subs

The CL3000A family of laser-processed logic devices can be used interchangeably in the same socket as Altera MAX 3000A CPLDs. Their manufacturer has developed a software technology that allows them to extract configuration information and reproduce it on new, non-programmable chips. They also take up less space and use fewer transistors. If you send the company a MAX3000A programming file, they send you configured sample chips within two weeks.

Clear Logic
San Jose, CA
(888) 548-7788
www.clear-logic.com

Core embedded in FPGA

The AT94K40 is a field programmable system level integration circuit. Rated at 30 MIPS at 40MHz, these devices are designed for use in low power, portable applications for areas such as telecommunications, networking, instrumentation, and automotive. They combine the embedded AT40K FPGA core with a high-performance version of the AVR 8-bit RISC microcontroller, two UARTS, timer/controllers, interrupt controller, programmable I/O ports, and 36KB of program and data RAM on a single chip. The microcode for the AVR microcontroller and the FPGA logic can be reconfigured an unlimited number of times. The AT94K40 is supported by the System Designer tools suite. The company says it will be the first in a series of similar devices. AT94K40 is sampling now and will start at $50 per unit when it ships.

Atmel
San Jose, CA
(408) 441-0311
www.atmel.com

OEMs

Pentium SBC

The VMICPCI-7710 is a Pentium III-based single-board computer. The processor has a 32-bit addressing and 64-bit data bus. Its superscalar architecture allows three instructions to be executed per clock cycle. A dynamic branch prediction unit, separate instruction and data caches, and MMX technology contribute to the processor's performance. Features of the VMICPCI-7710 include BIOS support for temperature-based clock throttling, BIOS support for USB legacy keyboard and mouse, an SVGA controller with 2MB internal SDRAM, a PMC mezzanine expansion site, support for on-board optional flash memory by means of a socket for an M-systems DiskOnChip flash device, an SMBus routed to a CPCI bus J3 connector, hardware support for software power-off, remote Ethernet booting, passive heat sink, up to 384MB SDRAM using two 144-pin SODIMM modules, on-board Fast Ethernet controller supporting 10BaseT and 100BaseT interfaces, Ultra-DMA-33 hard drive and floppy drive controllers through CPCI J3 connector, two high-performance 16550-compatible serial ports, a parallel port supporting ECP/EPP modes, two USB ports, PS/2-style keyboard and mouse port on front panel, and software-selectable watchdog timer with reset. The VMICPCI-7710 is available now for $3,100.

VMIC
Huntsville, AL
(256) 880-0444
www.vmic.com

PowerPC single-board computer

The EP1A-8240 is the latest member of the EmPower family of COTS VMEbus products, a series of communications-oriented boards in 6U and PMC form factor. The EP1A-8240 is based on the MPC8240 processor. Software support for the EP1A-8240's serial communication channels includes standard communication protocol modules from Trillium Digital systems. The modules have been ported to the EP1A-8240's hardware and provide a modular communications stack that offers configuration for adaptation to differentapplications. Modules currently available include HDLC, LAPB, and X.25 for synchronous operation. The EP1A-8240's single channel, dual redundant MIL-STD-1553B interface is implemented by a DDC mini-ace. Other supported features include a CompactFLASH site, 64MB SDRAM, 9MB FLASH, 10/100BaseT Ethernet and two PMC sites. In addition to the onboard PMC sites, PCI sub-system expansion is possible for both air and conduction cooled applications, based on a range of PCI mezzanine cards, including MIL-STD-1553, ATM, Serial, Graphics, Fiberchannel, and flash extensions.

Radstone Technology
Towcaster, Northants, England
44 (132) 735-9444
www.radstone.com

EMAC, Inc. has been designing Single Board Computers Since 1984, and offers a comprehensive line of products and services for the embedded systems market.

Turn-Key Solutions!

Web: www.emacinc.com • Phone: 618-529-4525 Fax: 618-457-0110

workingEngines[*inc]
Post-PC Era Talent Firm

placement / consulting / alliances

We strategically match, technically and culturally, highly skilled engineers with leading-edge companies who build smart devices and development tools.

Our vxJobs Practice focuses exclusively on positions requiring VxWorks expertise.

303-628-5560
www.workingEngines.com
www.vxJobs.com

(Complimentary resume & interview support. All fees employer paid.)
VxWorks is a registered trademark of Wind River, Inc.

DESIGN/DEVELOPMENT
ENGINEERS:

NES

OPPORTUNITIES IN:

Wireless Communications (Data, PCS, Cellular, Networks, Satcom), Digital Imaging, Computers, Software, Semiconductors, Medical, CATV, Defense, Process Control, Consumer Electronics

SKILLS IN ANY OF THE FOLLOWING:

Embedded SW, OOD/OOP, C, C++, WindowsNT/98, Solaris/UNIX, JAVA, BIOS, VRTX, PSOS, DSP, MIPS, PCI, VME, Mixed Signal, ASIC/FPGA, VHDL/Verilog, Device Drivers, System Architecture, LAN/WAN, IP, Wireless Design, CDMA, GSM, Video Compression

National Engineering Search® is the leading search firm placing Engineers nationwide. Contact us for immediate access to today's most exceptional engineering career opportunities! Our clients range from Blue Chips to today's newest emerging technology companies.

800/248-7020
Fax: 800/838-8789
esp@nesnet.com
See many of our current opportunities on-line at:
nesnet.com

What are you worth?
$ee our Online $alary Calculator!

scientific.com

Software and Hardware Professionals

Nationwide opportunities available for:
Engineers/Designers
Quality Assurance/Testers/Verification

- Embedded SW
- RTOS
- ASIC/FPGA
- IC Design
- C/C++
- Firmware
- VLSI
- Java

Scientific Placement, Inc.
800-231-5920 800-757-9003 (Fax)
crs@scientific.com

Free Resume Assistance
All fees are employer paid

EDS ▾ OPPORTUNITIES ▾

EDS is growing. Are you?

If you're stagnating in your present job, recent account wins have created huge new opportunities at EDS without cutting into your family life.

More than 100,000 professionals are enjoying the benefits of an EDS career. We offer excellent training, competitive salaries, and above all, room to grow. We provide cutting-edge systems and insights to top clients worldwide. If you're looking for a dynamic environment with all the opportunities you can handle, send us your resume today.

Senior Embedded Systems Software Engineers

- Embedded programming for microcontrollers using C and Assembly language including interrupt handlers and use of real-time executives
- Algorithm development, device drivers, Project Management, Formal Configuration Management, Bench development skills, J1850 and CAN

Embedded Systems Software Engineers

C or Assembler programming, Microprocessor Architecture and Interfacing experience, real-time microprocessor program development, Software Development Lifecycle, Requirements Analysis, Software Design, Coding, Unit and Integration Testing, Bench Development skills.

Windows CE Software Engineers

- 1-2 years Windows development experience using MS Visual C++
- Experience in Windows CE preferred, but will provide training if necessary

Engineering Associates

- 4-year degree in Electrical Engineering or Computer Engineering or equivalent work experience required

Positions available in North America.

Please mail, fax or email your resume, indicating position of interest, to: EDS Recruiting, Dept. 72-9722, Attn: Deborah Polvi, 700 Tower Drive, 5th Floor, Troy, MI 48098; Fax: (520) 833-9309; Email: deborah.polvi@eds.com.

EDS

EDS is a registered mark and the EDS logo is a trademark of Electronic Data Systems Corporation. EDS is an equal opportunity employer and values the diversity of its people. Copyright ©2000 Electronic Data Systems Corporation. All rights reserved.

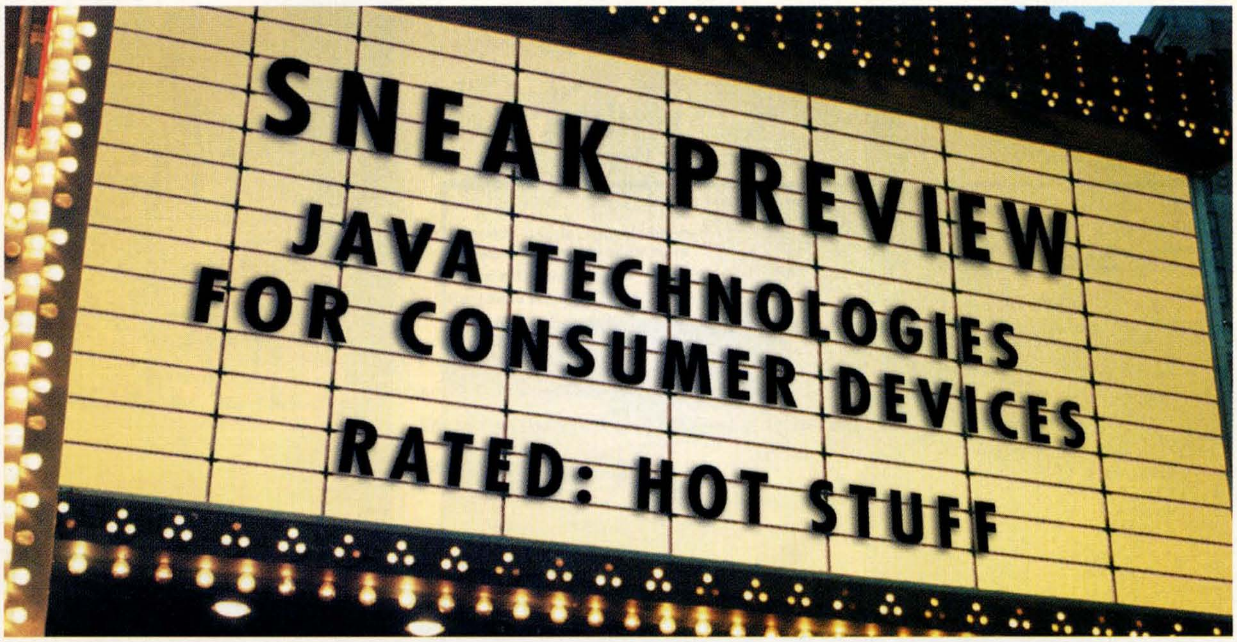
Having problems landing qualified embedded designers?

Advertise in Embedded Systems Programming's career section!

East Coast Sales
Damon Graff
781.839.1285

California Sales
Andres Diaz
415.278.5274

West Coast Sales
Sam Louis
415.278.5223



SCENARIOS THAT WERE HERETOFORE FOUND ONLY IN THE BIG SCREENS ARE BECOMING REAL, WITH SUN'S CONSUMER AND EMBEDDED TECHNOLOGIES PLAYING A FEATURE ROLE IN MAKING THEM HAPPEN. SUN IS COMMITTED TO PROVIDING END-TO-END SOLUTIONS BASED ON OPEN STANDARDS THAT WILL MAKE CONSUMERS' LIVES MORE PRODUCTIVE AND ENJOYABLE. IF YOU WANT A STARRING ROLE, THIS IS YOUR TICKET! THE NETWORK SERVICE PROVIDER DIVISION GROUP IS SEEKING: ★ HW ENGINEERS, BOARD DESIGN ★ HW ENGINEERING MANAGER/PROJECT LEAD, BOARD DESIGN ★ SYSTEM EMBEDDED ENGINEERS. CHECK OUT WWW.SUN.COM/SPJOBS

THE CONSUMER AND EMBEDDED DIVISION GROUP IS SEEKING: J2ME™ CORE DEVELOPMENT: ★ VM AND COMPILER DESIGN ENGINEERS ★ J2ME SOFTWARE TESTERS ★ GUI DEVELOPERS ★ JAVACARD DEVELOPERS ★ J2ME API DEVELOPMENT: WIRELESS ★ JAVATV DEVELOPERS ★ CONTENT ENGINEERS ★ SR. PRODUCT MGR., EMBEDDED TECHNOLOGIES. SUN IS PROUD TO ENSURE THAT EQUAL TALENT ALWAYS GETS EQUAL OPPORTUNITY. DROP YOUR RESUME OFF WITH SUBJECT HEADING AESP100AB/CS AT WWW.JAVA.SUN.COM/JOBS.

Sun® microsystems

We're the dot in .com™

JAVA

SIEMENS

Shifting towards innovative solutions for the wireless world.
Engineering Innovation San Diego Style

The wheels of the wireless mobility market are turning and there's a new competitor leading the pack. Combining the agility and energy of a start-up, Siemens Mobile Phones is poised to become an overnight success with the help of some of the nation's finest professionals. Which is where you come in. You're invited to bring your wireless prowess and innovation to the table. You'll get the chance to design and develop some amazing wireless technology for the future.

We'll give you the resources, the tools and the freedom to reach your potential. And at Siemens Mobile Phones, you've got the room to take your career as far as you can imagine.

We have opportunities in the following areas for CDMA and TDMA Technology at our San Diego Location:

- **Embedded Software Engineers**
- **HW/Baseband Engineers**
- **MMI Software Engineers**
- **Marketing Positions**
- **Mechanical and PCB Layout Engineers**
- **Product Managers**
- **RF Engineers**
- **Software Integration Engineers**
- **Software Engineers**
- **Supply Chain Manager**
- **Systems Engineers**
- **Test Engineers**
- **Technical Project Lead**

For other exciting opportunities visit our web site.

To apply, send your resume to: **sdresumes@icp.siemens.com** indicating code: 029IESP. Siemens is an equal opportunity and diversity employer.

EURO-ENGINEERING, CALIFORNIA STYLE.

Siemens Mobile Phones

www.icm.siemens.com

Rent the Hottest List around!

Embedded Systems PROGRAMMING

Over 55,000 Qualified Subscribers

Call Rubin Response 847-619-9800

In-Circuit Emulators

• 8051 • 80186 • 80196
• DSP 320Cxxx • TriCore • PIC

- Real-time trace
- Serial, parallel, & LAN interface
- C/C++ level Chameleon Debugger
- Multi-core debugging

www.signum.com

SIGNUM SYSTEMS
800-838-8012
805-523-9774
Fax: 805-523-9776
11992 Challenger Ct. • Moorpark, CA 93021

We can save your valuable time!

INTRODUCING! **Full Featured In-Circuit Emulator**

- Supports a wide range of micro-processors & micro-controllers
- Universal Architecture
- Real Time Filtering Trace
- 100MHz Bus Speed Support
- C/C++ Source Level Debugger
- Easy to Use!

- Hitachi H8/H8S SH1/2/3/4
- Qualcomm MSM3000 MSM3100
- ARM ARM7/9 *"ETM support"*
- NEC V800/V850
- Motorola 68K/PowerPC
- Mitsubishi M16C/M32R
- Fujitsu SPARClite FR30/FR40
- Toshiba, Sharp, Panasonic, OKI & more !!

Also ask about our Flash Programmer!

Yokogawa Digital Computer Corp of America
Tel: (408) 244-1932
Fax: (408) 244-1881
http://www.ydcusa.com
email: info@ydcusa.com

YOKOGAWA

MICROPROCESSOR CORE SAVES COST AND DESIGN TIME

Design your custom board around the Rabbit 2000™ core module. Easy and rapid development, plus you save production and component costs!

$29 qty 100
$39 qty one
1.90" x 2.30"

- 40 I/O pins • 4 serial ports • 7 timers
- Battery-backed time/date clock
- 128K-512K SRAM • 256K flash
- Clock up to 25.8MHz

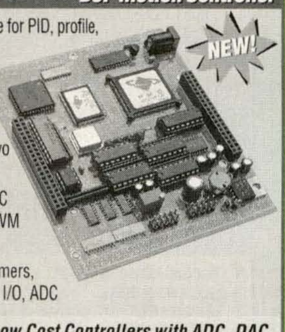
Dev. Kit Only $169
Includes RCM2020 core module, prototyping board, Dynamic C® software (not a trial version!), serial cable for real-time debugging, and documentation on CD-ROM

Z-WORLD
Call toll-free
888.362.3387
or buy online at
www.zworld.com

2900 Spafford Street, Davis, CA 95616 • Tel 530.757.3737 • Fax 530.753.5141

MotionC™ 2140

C/C++ programmable Low Cost, Standalone DSP Motion Controller



- Built-in firmware for PID, profile, electronic gear
- 4.7" x 3.8", 386EX/186 PMD MC2140
- 2- or 4-axis servo motion control
- Quadrature/ADC inputs, DAC/PWM outputs
- RS-232/485, timers, watchdog, TTL I/O, ADC

We have 30+ Low Cost Controllers with ADC, DAC, solenoid drivers, relays, PCMCIA, LCD, DSP motion control, 10 UARTs, 300 I/Os. Custom board design. Save time and money!

TERN INC.
1724 Picasso Ave., Suite A
Davis, CA 95616 USA
Tel: 530-758-0180 • Fax: 530-758-0181
http://www.tern.com
tern@netcom.com

68HC12

Background Mode Debugger

EMUL12™-PC/BDM

Real-Time Microprocessor Development Tools

Call **(408) 866-1820** for a product brochure and a FREE Demo Disk.
Information is also available via Fax, call our 24-hour Fax Center at (408) 378-2912.
Visit our web site- **http://www.nohau.com**

See EEM '97- pages D 1274-1282

NOHAU CORPORATION
51 E. Campbell Avenue
Campbell, CA 95008-2053
Email: sales@nohau.com

In-Circuit Emulators

Please email us or visit our website for our new US office contact and full-line catalog!

ACROSSER *Industrial PC-Based Control Systems*
www.acrosser.com
Tel: 886-2-8512-1616
Fax: 886-2-8512-2282
webmaster@acrosser.com

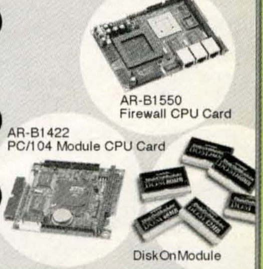
Total Embedded Solution

Embedded PC
PC/104 Module
386 / 486 / 586

Embedded OS
Embedded NT
Linux (Fully Support)

Embedded Storage
DiskOnModule

Embedded Housing
Microbox
PC/104 box



* H/W with full embedded OS compatibility
* Flash Disk with no special socket and driver needed Only regular IDE required.
* With a complete embedded platform, Acrosser takes care all your embedded needs!

Fast. Reliable. Affordable.

Parallel port EMP-20 $449.95
Parallel port EMP-10 $219.95
Parallel port EMP-30 $995.00
PC based $139.95

NEEDHAM'S DEVICE PROGRAMMERS are the easiest and most cost-effective way to read, program and verify 2716 – 8 meg EPROMS. Support for Micros, Flash, EPROM, 16-bit, PLDs, Low Voltage and Mach (call for support list for specific models, or download demos from our BBS or web site). Easy to use menu driven software features on-line help, and a full-screen editor. Support for macros, read and save to disk, and split and set options.

- **Free technical support • Free software upgrades**
- **1 to 2 year warranty on all parts and labor**
- **30-day money-back guarantee • Made in the U.S.A.**
- **All models include software, on-line help, cables, and power transformers (where applicable)**

NE **(916) 924-8037**

NEEDHAM ELECTRONICS, INC.
4630 Beloit Drive, #20, Sacramento, CA 95838
FAX (916) 924-8065 • BBS (916) 924-8094
(Mon. – Fri. 8 am – 5 pm, PST)
http://www.needhams.com/

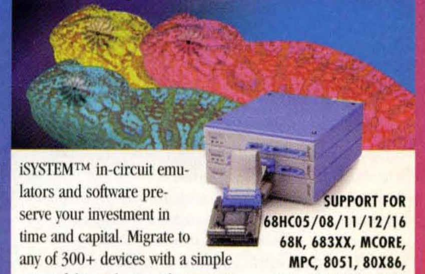
PromICE with Trace

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

Emulators that change so you don't have to.



iSYSTEM™ in-circuit emulators and software preserve your investment in time and capital. Migrate to any of 300+ devices with a simple swap of the pod or a software setup. And retain the same user interface with winIDEA™, our integrated development environment. Adaptable, powerful, and easy to use–that's great value.

SUPPORT FOR 68HC05/08/11/12/16 68K, 683XX, MCORE, MPC, 8051, 80X86, C500, Z80/180, CR16, PIC AND MANY MORE.

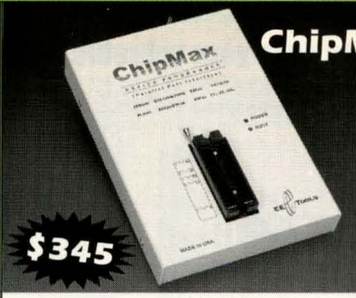
Call for your free demo CD
1 888 543-5300

iSYSTEM
usa@isystem.com
www.isystem.com

8051, 80C196
PIC®, AVR®

Integrated Development Systems:
Compilers, Simulators, Programmers,
In-Circuit Emulators
Non-intrusive, with trace feature, hardware unconditional & complex breakpoints, triggers, programmable clock, memory mapping & banking, full project & source level support for C compilers…
www.phyton.com
(718) 259-3191 **Phyton**™

ChipMax



- Programs over 1200 devices (E(E)PROM, Flash, Serial, PALCE, GAL, 875x/895x, 93Cxx, 17xx, PIC16xx)
- Fast parallel link to any PC, even laptops
- 40-pin universal pin driver and current limit
- On-board processor and built-in power supply
- Unbeatable programming speed
- Checks for incorrect device insertion
- Automatic EPROM ID search
- Supports WIN95/98/NT
- NO fans & modules are required in circuit
- Made in USA, Lifetime free software
- Visit web site for demo software

EE TOOLS
Sunnyvale, California, USA
Tel: 408•734•8184
Fax: 408•734•8185
www.eetools.com

TargetFFS™
Flash File System

- Re-entrant Embedded File System
- POSIX and Standard C API
- Guaranteed Integrity Across Resets
- Complete Wear Leveling
- "Thin" Driver Layer
- 100% ANSI C Source Code
- Royalty Free

BLUNK Microsystems
(408) 323-1758
www.blunkmicro.com

INTROL

INTROL CODE 4.00 is a full featured cross-platform, multi-target microcontroller development environment.

68HC05
68HC08
68HC11
68HC12
68HC16
68332

- **Simulators**
- **C Compilers**
- **Macro assemblers**
- **Source Level Debugger**

Integrated Development Environment

CODE comes with a full year of technical support and free upgrades.

Visit us on the web @ www.introl.com
email: sales@introl.com
U.S. and Canada: 800.327.7171
Other: 414.273.6100
FAX: 414.273.6106

Vesta Technology, Inc.
Single Board Computers for the 21st Century

SBC2000-332

Vesta BASIC
or C Software
32 bit 68332
Low Power
2 x RS-232
Watchdog
Real Time Clock
E²PROM
LCD/Keypad
1 Meg FLASH
1 Meg ROM
1 Meg RAM
$299 @ 1
$188 @ 100

(303) 422-8088
www.sbc2000.com

IN-CIRCUIT EMULATORS
The Right Emulator at the Right Price!

8051 • 68HC05
68HC11 • COP8 • CR16
DS80C320/520/530 nX65K
78K4 • SM6000/SM8500

Analog Devices • Atmel
Dallas • Intel • Motorola
National • NEC • OKI
Philips • Sharp • Siemens
SST • Temic • Winbond

- Windows–Based Interface
- Source-level debug
- Real-time/Non-intrusive
- Rentals available

MetaLink® Corporation

Headquarters, Arizona, USA	MetaLink - Europe GmbH
Tel: 480.926.0797	Tel: 49 (8091) 56960
Fax: 480.926.1198	Fax: 49 (8091) 2386
sales@metaice.com	islinger@metalink.de
www.metaice.com	www.metalink.de

SX Debugger+Programmer

Complete System **$375**
F R E E
Demo/Prototype board
All tools **qualified** by Scenix Semiconductor

SX-ISD

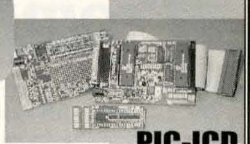
• Supports SX18/20/28/48/52 • In-circuit run-time debugging • Real-time code execution • Source level debugging • Built-in programmer • Real-time breakpoint • Conditional animation break • External break input • Frequency synthesizer • Selectable internal frequency • External oscillator support • Software trace • Unlimited watch variables • Parallel Port Interface • Runs under Win 95/98/NT • Comes with SASM Assembler

Single, Gang Programmers and SMT adapters are also available

Advanced TransdAtA CORPORATION
Dallas, Texas
Tel 972.980.2960 **www.adv-transdata.com**

PIC Real-time 'Emulator'

$159



PIC-ICD

PIC-ICD
= Debugger + Programmer + DemoBoard

• Designed for 16F87X • Can also support most 16C6X/7X • In-circuit run-time debugging • Real-time code execution • 2.5-6V operating voltage • Source level debugging • Built-in programmer • Real-time breakpoint • Conditional animation break • 2 External break inputs • Selectable internal frequency • Software trace • Unlimited watch variables • Parallel Port Interface • Runs under PICICD IDE (Win95/98/NT) or MPLAB (Win95/98)

Advanced TransdAtA CORPORATION
Dallas, Texas
Tel 972.980.2960 **www.adv-transdata.com**

PIC16/17 Emulator

New Low Prices

Complete system starts at **$599**

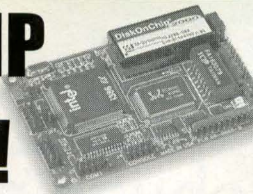
RICE17A

• For PIC12/16/17 • 3-5volt emulation • 64K program memory • 32K real-time trace • 12-clip external probe • Source level debugging • External break input • Trigger and break output • Real-time breakpoint • Unlimited software break and trigger points • Selectable internal frequency • Unlimited watch variables • Parallel Port Interface • Runs under Win95/98/NT

Probes for **16F87x** and **16C77x** by Jan 2000
With data break support

Advanced TransdAtA CORPORATION
Dallas, Texas
Tel 972.980.2960 **www.adv-transdata.com**

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**
See us at the Emb Sys Conf San Jose #5028
www.jkmicro.com/uflash

JK *microsystems*

From the Author of WATTCP

eRTOS

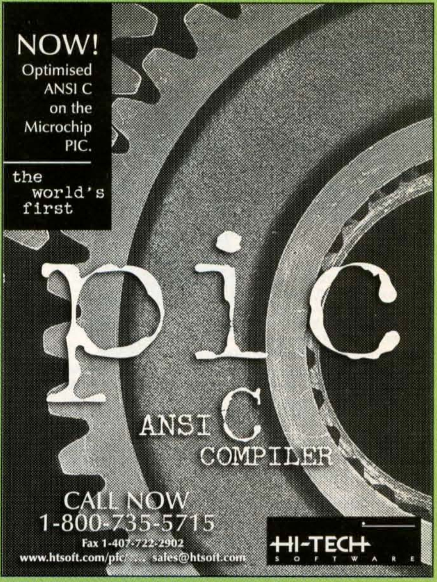
DOS Realtime Kernel with TCP/IP Support

- ***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*



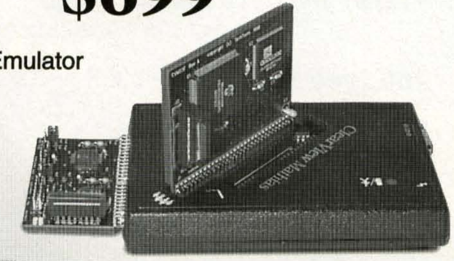
TechTools™ Embedded Systems Development Tools

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.

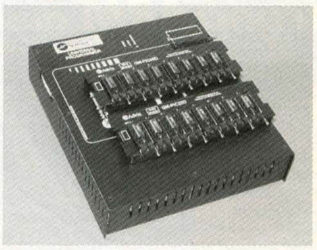
See why Engineers choose TechTools
Visit our website and request your FREE CDROM!

www.tech-tools.com

CONTACTS: (972) 272-9392 Fax: (972) 494-5814 Email: sales@tech-tools.com

---

Advin



Production and Engineering Programmers
Extensive support for
Microchip PICs and others,
Wide variety of package types supported.
Thousands of happy customers worldwide
are living proof of our product quality.
Please contact us and find out why
Advin is your best choice.
– Accept trade-ins of out-dated Data I/O models –
www.advin.com
1-888-GO-ADVIN 408-243-7000

SB-56K Multi-DSP Emulator

Support for the Motorola DSPs:
DSP560xx, DSP563xx, DSP566xx, DSP568xx

SB-56K supports any combination and any count (up to 255) of the devices from the above families. With its accurate counter allowing to measure code execution (benchmarking), small size (1"x2.5"x4"), high speed RS-232 interface, the SB-56K can provide independent support for multiple devices with option to access each device on the target board from different workstations connected through LAN, WAN or Internet.

DOMAIN TECHNOLOGIES, INC.
1700 Alma Dr.,#495, Plano, TX 75075
Tel.:(972) 578-1121 / Fax: (972) 578-1086
info@domaintec.com
www.domaintec.com

GALEP-III

Pocket Multiprogrammer

This size fits all!

◆ Programs 8-bit and 16-bit EPROMs, EEPROMs, Zero Power RAMs, Flash, serial EEPROMs ◆ GAL, PALCE, ATF ◆ 87xxx, 89xxx, PIC12/16/17Cxx ◆ All DIL devices **without adaptor** ◆ **Lightning fast** parallel data transfer (e.g. 27C512 read/compare 2 sec!) ◆ Power supply independent due to **rechargeable battery** ◆ Uses PC printer port ◆ Hex, JEDEC, and binary file formats ◆ Hex and fusemap buffer editor ◆ Split & shuffle for 8-bit, 16-bit and 32-bit targets ◆ Runs under Win3.1, 95, 98, **NT** ◆ 'Remote control' by DDE scripts ◆ Designed for the future due to flexible pin driver technology - new devices will be added every month ◆ Device list, demo software and **lifetime free updates** from our website www.conitec.com!

GALEP-III Set with cable, battery, recharger... **$333.00**

PLCC Adaptor for 8-bit EPROMs / 16-bit EPROMs / GALs .... **each $149.00**

CONITEC 1951 4th Ave, Ste 301 • San Diego, CA 92101
Tel: (619) 702-4420 • http://www.conitec.com

NORAL
The Debugging Company

Real-Time Debugging

CPU32 ColdFire 68HC12
68302 68020 68306

call toll free info@noral.com

888-88-DEBUG

www.noral.com

UniSTAC™ for Strong ARM® designs (SA-1110, SA-1100)

UniSTAC is a full-featured, high-end development system for Strong ARM® SA-1110 designs.
Powerful In-circuit Emulator featuring: 8 Mbytes emulation memory, non-intrusive Real-time trace capture and display, support for hardware breakpoints.

■In-Circuit Emulator Features:
- 64K frames Real-time Bus Trace
- 8 Mbytes emulation memory
- 1 hardware breakpoints
- Unlimited software breakpoints
- Program performance analysis and code coverage
- Test target included

■Sophia Interface with target
- Sophia original connector-without removing target CPU
- Adapters also available for BGA256.

■Advanced GUI source level debugger Watchpoint®
Sophia's powerful high-level language debugger hosted on Windows® 95/98, WindowsNT®

Also Supported:
Power PC, ARM7/9, 186/386/486, Pentium®, 680x0, SuperH, V800/850E, R3000/4000, M16C, M32R

Sophia systems™
408-467-9911

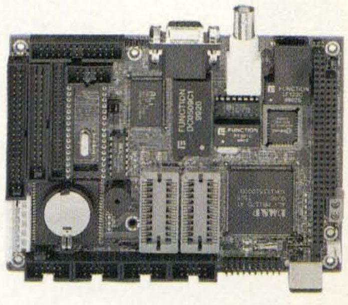
www.sophia.com

A Real x86 PC Based Embedded Solution

It's a System on a Chip

embedded@nucleus1.com

- Wide ranges of 386 and 5x86 available
- Low power consumption and extended temperature ranges
- Guaranteed low cost and long-term supply
- ISA bus, PC/104, 3.5" or custom sizes available
- Custom design or De-population available



Toll Free 1(800) 683-7335

www.nucleus1.com

NUCLEUS Electronic Corp.
Tel: (909) 468-5700 Fax: (909) 468-5704

SIMULATOR

ChipView®

8051 and derivatives
Intel 80196 / 80296

This remarkable tool utilizes ActiveX technology to provide fast, cycle-accurate simulation of CPU, interrupts, and on-chip peripherals.

Add your own ActiveX controls to simulate off-chip peripherals, ASICs, and even entire systems!

Attach to HTML ActiveX controls to mock-up and test your interface!

Interface to your HDL simulator for hardware/software co-verification!

Download a 30-Day Trial Today!
www.chiptools.com

ChipTools®
905-274-6244 FAX:905-891-2715

DEBUGGERS

Windows® CE Embedded Computer

Windows & Microsoft Are Registered Trade Mark Of Microsoft Corp.

Touch Screen 6"
100 Mhz CPU
1/4 VGA 320 x 240

FREE! *Visual Basic* *Visual C++*
Embedded Tools Microsoft®

***100%** Ready to program using the latest Microsoft 32-Bit Embedded tools. Built-in Remote Debug and Touch Screen makes this complete package easy to use. For More Information and Pricing Visit Our Web Site.*

Features
- *Elan SC-400 (486 100 Mhz)*
- *Solid State Disk*
- *Serial Ports (3)*
- *Printer Port*
- *Digital I/O (16)*
- *Analog Inputs (6), 12 Bit*
- *PCMCIA Card Interface*

RLC ***www.RLC.com***
Enterprises, Inc.
Toll Free 1-888-RLC-TECH

Embedded Systems
Programming
Marketplace

East Coast Sales
Damon Graff
781.839.1285

California Sales
Andres Diaz
415.278.5274

West Coast Sales
Sam Louis
415.278.5223

QuickTool Finder

www.avocetsystems.com

go

Every Chip
Every Tool
One Source
One Click

C Compilers

Assemblers

Simulators

RTOS

In-Circuit Emulators

BDMs

AVOCET SYSTEMS, INC.
(800) 448-8500

www.avocetsystems.com

Select Chip
PowerPC Families
MPC5xx
MPC8xx
PPC4xx
68K Families
680x0
683xx
Motorola 8/16 Bit
68HC05/08
68HC11
68HC12
68HC16
ColdFire
6801/03
8051 All Derivatives
80C320/520
8xC751/2
C500
DS5000
8080/85
8051XA
80166
80196xx
8086/88
x86 Intel/AMD
80386EX
80186/188xx
Conexant/WDC
6502/65816
C18/19/29
Z180/64180/Z80
Z8/Z8000/Zx80
Microchip PIC
V20/30/50
320C20
TLCS-90

(800) 448-8500

"The best emulator I ever used!"

8051

- more than 500 derivatives supported
- small emulation probes
- real-time access to internal bus
- can trigger on internal bus events
- cascading triggers
- trace with timestamps
- dual-ported emulation memory
- external trace and triggers
- excellent HLL support
- code coverage
- performance analysis

hitex
DEVELOPMENT TOOLS

1-800-45-hitex **www.hitex.com**

WRITE FOR EMBEDDED SYSTEMS PROGRAMMING

share your
knowledge

contact us
mbarr@cmp.com

www.embedded.com/wriguide.htm

Advertiser	URL	Page
Abraxas Software	www.abxsoft.com	173
Accelent Systems	www.accelent.com	46
Accelerated Technology	www.atinucleus.com	125
Acrosser Technology Co.	www.acrosser.com	163
Advanced Transdata Corp.	www.adv-transdata.com	164
Advin Systems	www.advin.com	165
Agilent Technologies	www.agilent.com	97
Agilent Technologies	www.agilent.com	19
American Arium	www.arium.com	C3
American Raisonance	www.americanraisonance.com	88
American Raisonance	www.americanraisonance.com	100
Applied Data Systems	www.flatpanels.com	157
ARC Cores Ltd	www.arccores.com	81
Arcom Control Systems	www.arcomcontrols.com	126
Artisan Software Tools	www.artisansw.com	143
Axiom Technology	www.axiomtek.com	77
Blackhawk	www.blackhawk-dsp.com	106
Blunk Microsystems	www.blunkmicro.com	164
Bsquare Corporation	www.bsquare.com	63
Byte Craft Limited	www.bytecraft.com	114
Cadence	www.cadence.com/systems	39
Cad-ul	www.cadul.com	152
Ceibo	www.ceibo.com	158
Chip Tools	www.chiptools.com	166
CMX Systems, Inc.	cmx.com	175
Compuware NuMega	www.compuware.com	29
Conitec	www.conitec.com	166
Copper Mountain	www.jobs.coppermountain.com	107
Cosmic Software	www.cosmic-software.com	94
Dinkumware Ltd.	www.dinkumware.com	173
Domain Technologies	www.domaintec.com	165
dSPACE	www.dspaceinc.com	89
EBS, Inc	www.ertfs.com	59
EBSnet	www.ebsnetinc.com	175
EDS	www.eds.com/careers	160
Electronic Engineering Tools	www.eetools.com	164
EMAC, Inc.	www.emacinc.com	159
Embedded Power	www.embeddedpower.com	119
Embeddedtechnology.com	www.embeddedtechnology.com	156
Emware	www.emware.com	67
Emwerks	www.emwerks.com	145
Enea OSE Systems, Inc.	www.enea.com	123
esmertec ag	www.esmertec.com	105
Espial Group	www.espial.com	71
Express Logic	www.ghs.com/armsolutions	116
Express Logic	www.expresslogic.com	127
General Software	www.gensw.com	90
Grammar Engine Inc	www.gei.com	163
Green Hills Software Inc	www.ghs.com/armsolutions	116
Green Hills Software Inc	www.ghs.com	4
Hitachi Semiconductor	www.superh.com	103
HI-TECH Software	www.htsoft.com	109
HI-TECH Software	www.htsoft.com/pic	165
Hitex Development Tools	www.hitex.com	167
Hitex Development Tools	www.hitex.com	93
Hiware	www.hiware.com	154
IAR Systems	www.iar.com	148
I-Logix	www.ilogix.com	138
Intel Ceg	www.intel.com	16
InterNiche Technologies	www.iniche.com	60
Int'l Microsystems, Inc	www.cardduper.com	147
Introl	www.introl.com	164
iSYSTEM USA, Llc	www.isystem.com	163
JK Microsystems, Inc.	www.jkmicro.com	165
JK Microsystems, Inc.	www.jkmicro.com/uflash	165
Kadak Products	www.kadak.com	121
Keil Software	www.keil.com	142
Lantronix	www.lantronix.com	55
Lauterbach	www.lauterbach.com	41
Lineo	www.lineo.com	113
Lineo	www.lineo.com	153
LynuxWorks, Inc.	www.lynuxworks.com	110
Metalink Corp	www.metaice.com	164
MetaWare, Inc.	www.metaware.com	135
Metrowerks, Inc.	www.metrowerks.com	137
Micro Digital	www.smxinfo.com	115
Micro/sys	www.embeddedsys.com	147
Microsoft Windows CE	www.microsoft.com/windowsce/embedded	26,27
Microtek International	www.microtekintl.com	6
Microware Systems Corp	www.microware.com	129
MontaVista Software, Inc.	www.mvista.com	20,21
Motorola Semiconductor Hp	www.digitaldna.motorola.com	43
National Semiconductor	www.national.com	83
Nat'l Engineering Search	www.nesnet.com	160
Needhams Electronics	www.needhams.com	163
Net Media, Inc.	www.netmedia.com	C2
NETsilicon	www.netsilicon.com	10
NewMonics, Inc.	www.newmonics.com	15
Nohau Corporation	www.nohau.com	163
Nohau Corporation	www.nohau.com	34,35
Noral Micrologics Limited	www.noral.com	166
Nucleus Electronics Corp	www.nucleus1.com	166
Pacific Softworks	www.pacsoft.com	51
Pacific Softworks	www.pacsoft.com	49
Paradigm	www.devtools.com	130
ParaSoft	www.parasoft.com	133
Phar Lap Software Inc	www.pharlap.com	118
Phoenix Technologies Ltd	www.phoenix.com	72
Phyton, Inc.	www.phyton.com	164
Precise Software	www.psti.com	131
QNX Software Systems Ltd	www.qnx.com	8,9
Rabbit Semiconductor	www.rabbitsemiconductor.com	95
Radisys Corp.	www.radisys.com/ss7	101
Rational Software	www.rational.com	74
Reasoning	www.reasoning.com	141
RLC Enterprises	www.rlc.com	166
Scientific Placement	www.scientific.com	160
Siemens	www.icm.siemens.com	162
Signum Systems	www.signum.com	163
Silicon Storage Technologies	www.superflash.com	25
Silicon Storage Technologies	www.superflash.com	23
Sleepycat Software	www.sleepycat.com	122
Sophia Systems	www.sophia.com	166
Sun Microsystems	www.java.sun.com/jobs	161
Tasking	www.tasking.com	18
Tech Tools	www.tech-tools.com	165
Tektronix	www.tektronix.com/tla600	99
Tern Inc	www.tern.com	163
The Math Works	www.mathworks.com	33
Tilcon Software Ltd.	www.tilcon.com	22
Treck, Inc.	www.treck.com	53
Trillium Digital Systems	www.trillium.com	65
US Software	www.ussw.com	57
Vesta Technology Inc	www.sbc2000.com	164
Vita Nuova Holdings Ltd.	www.vitanuova.com	31
Waferscale	www.waferscale.com	151
White Mountain DSP	www.wmdsp.com	91
Wind River Systems	www.wrs.com	1
Wind River Systems	www.wrs.com	12,13
Working Engines Inc	www.workingengines.com	160
Xilinx	www.xilinx.com	85
Yokogawa Digital Computer	www.ydcusa.com	163
ZF Linux Devices	www.zflinux.com	69
Zilog Inc.	www.zilog.com	44
Zucotto Systems, Inc.	www.zucotto.com	98
Z-World Engineering	www.zworld.com	163

FREE SUBSCRIPTION REQUEST FORM

EmbeddedSystems
P R O G R A M M I N G

P.O. BOX 3404 • NORTHBROOK, IL 60065-9468
www.embedded.com

PLEASE ANSWER ALL QUESTIONS
(FRONT AND BACK) THEN SIGN AND DATE THE CARD.
Incomplete cards cannot be processed or acknowledged.

1. Do you wish to receive/continue to receive EMBEDDED SYSTEMS PROGRAMMING?

☐ **YES** ☐ No

Signature (required) X______________________________

Name (please print) ______________________________

Date ______________________________

Job Title ______________________________

Company Name ______________________________

Address ______________________________ ☐ Work ☐ Home

Mail Stop______________________________

City ____________ State ________ Zip ________

Phone ( ) ______________________________

Fax: ( ) ______________________________

E-Mail:______________________________

If you do not wish to receive future communications from *CMP/Miller Freeman, Inc.* via e-mail please check here ☐

If you would prefer delivery to your home, please complete home address.
Company name and address are still required to qualify.

Address ______________________________

City ____________ State ________ Zip ________

2. Check all of the following that you are involved in doing or managing at your company or at those companies to whom you consult. (Please check all that apply)

01 ☐ Architecture Selection/ Specification
02 ☐ Writing Software for Embedded Systems
03 ☐ Writing/Embedding Real-Time Operating System/Kernel
04 ☐ Debugging Software
05 ☐ Debugging Hardware
06 ☐ Hardware/Software Integration
07 ☐ Hardware/Software Co-Design
08 ☐ Device Programming
09 ☐ Project Management
11 ☐ Software Design/Analysis
12 ☐ Prototype Testing
13 ☐ Designing Hardware for Embedded Systems
14 ☐ Board Layout/Design
17 ☐ Hardware/Software Co-Verification
18 ☐ Hardware/Software Partitioning
19 ☐ Software Testing
20 ☐ SOC (System-on-Chip) Design
21 ☐ Internet Appliance Design
15 ☐ Other (please specify) ______________________________
16 ☐ I'm not involved in Embedded Development in any way.

www.espmag.com

B010

3. What is your principal job function? (Please check only one)

Engineering/Computer Management
01 ☐ Executive Management (i.e., President, VP, Owner, Chairman, Partner)
02 ☐ Engineering Management (i.e., Technical Director, Chief Engineer, Department Manager, Group Manager)
03 ☐ Software Engineering/ Programming/Development Management (i.e., Sr Software Engineer, Principal Software Programmer)
04 ☐ Systems Engineering/ Development Management (i.e., Sr Design, Hardware, or Test Engineer)
05 ☐ Other Management (please specify)______________________________

Engineering/Programming Personnel
06 ☐ Software Engineering/ Programming/Development
07 ☐ Systems Engineering/ Development (i.e., Design, Hardware, or Test Engineer)
08 ☐ Engineering Support (Technician, Programming Staff)
09 ☐ Scientific/R&D/Education
10 ☐ Other staff (please specify)______________________________

4. Please check all products which you specify, recommend, authorize, or purchase. (Please check all that apply)

ICs and Semiconductors
01 ☐ Microcontrollers/ Microprocessors
02 ☐ 4/8-bit µC/µP
03 ☐ 16-bit µC/µP
04 ☐ 32-bit µC/µP
05 ☐ 64-bit µC/µP
06 ☐ X-86/Pentium
07 ☐ Digital Signal Processors
08 ☐ EPROM / EEPROM
09 ☐ Flash
10 ☐ DRAM/SRAM
11 ☐ Communication ICs
12 ☐ Media Processors
13 ☐ CPLDs/FPGAs
14 ☐ System-on-Chip (SOC)
15 ☐ MCU Peripheral Chips
16 ☐ Hardware IP/Cores

System Boards
20 ☐ Single Board Computers
21 ☐ VME Boards
22 ☐ Embedded PCs
23 ☐ PCI Boards
24 ☐ cPCI Boards
25 ☐ DSP Boards

Computer Systems
29 ☐ PCs
30 ☐ NT Workstations
31 ☐ Unix Workstations
32 ☐ Linux Workstations

Software
36 ☐ Real-Time Operating Systems/ Kernels
37 ☐ Compilers/Cross Compilers
38 ☐ Assemblers/ Cross Assemblers
39 ☐ Software Debuggers
40 ☐ Object-Oriented Design Tools
41 ☐ Simulators/Modeling Tools
42 ☐ Version/Change Control Software
43 ☐ Communications Software/ Protocols
44 ☐ ROMable DOS Tools
45 ☐ Device Driver Tools
46 ☐ Embedded Databases
47 ☐ Embedded Web/Internet Tools
48 ☐ GUI Development Tools
49 ☐ Open Source Tools
50 ☐ Java Tools
51 ☐ Software Testing Tools
52 ☐ Integrated Development Environments (IDEs)
53 ☐ Windows CE Tools

Design Tools/Test Equipment
56 ☐ In-Circuit Emulators
57 ☐ Logic Analyzers
58 ☐ Oscilloscopes
59 ☐ Data Acquisition Equipment
60 ☐ Device Programmers
61 ☐ Hardware/Software Co-Design Tools
62 ☐ Hardware/Software Co-Verification Tools
63 ☐ None of the above

5. What is the primary end product or service performed at your location? (Please check only one)

01 ☐ Computers/Peripherals/Office Automation
02 ☐ Communications/Telecommunications/Networking
03 ☐ Consumer Electronics/Entertainment/Multimedia
04 ☐ Automotive Transportation Systems and Equipment
05 ☐ Government/Military Electronics
06 ☐ Aerospace/Space Electronics
07 ☐ Industrial Controls
08 ☐ Electronic Instruments/ATE/Design & Test Equipment
09 ☐ Medical Electronic Equipment
10 ☐ Other: (please specify)______________________________

6. What is your engineering/development responsibility? (Please check only one)

01 ☐ I manage an engineering or software development department
02 ☐ I manage a project team
03 ☐ I manage a project
04 ☐ I am a member of a project team
05 ☐ Other (please specify)______________________________

MORE QUESTIONS ON BACK ➤

CMP

7. Do you specify or buy through distributors?

01 ☐ Yes
02 ☐ No

8. How many employees are there at your company?

01 ☐ 1000 or more
02 ☐ 500-999
03 ☐ 250-499
04 ☐ 100-249
05 ☐ 50-99
06 ☐ 1-49

9. Please check the publications below that you receive personally addressed to you by mail. (Please check all that apply)

02 ☐ EDN
03 ☐ EE Times
04 ☐ Electronic Design
06 ☐ Penton's Embedded Systems Development
05 ☐ None of the above

10. How many other people read your copy of EMBEDDED SYSTEMS PROGRAMMING?

01 ☐ 1	05 ☐ 5
02 ☐ 2	06 ☐ 6
03 ☐ 3	07 ☐ 7
04 ☐ 4	08 ☐ None

11. Please list the names of others at your location who may be interested in a free subscription to EMBEDDED SYSTEMS PROGRAMMING

Name Title

Company name

Company address

City State Zip

PUBLISHER RESERVES THE RIGHT TO SERVE ONLY THOSE INDIVIDUALS WHO QUALIFY.

www.espmag.com

FOR FASTER SERVICE FAX BOTH SIDES TO: (847) 291-4816

FOLD FOR MAILING

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 2253 NORTHBROOK IL

POSTAGE WILL BE PAID BY ADDRESSEE

CMP

EmbeddedSystems PROGRAMMING

PO BOX 3404
NORTHBROOK IL 60065-9468

PLEASE TAPE ALONG THIS EDGE. NO STAPLES.

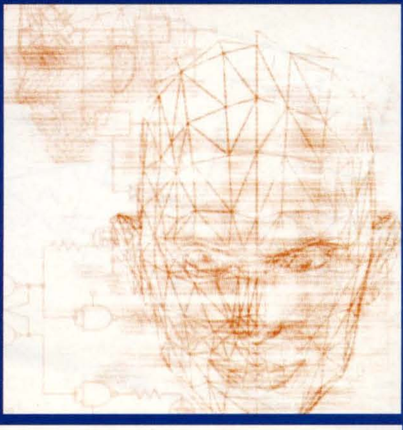
ICSPAT

ICSPAT

International Conference on Signal Processing Applications & Technology

October 16-19, 2000 • Adam's Mark Hotel • Dallas, TX, USA

Meeting of the Minds



4 easy ways to register

Web:
www.dspworld.com

Phone:
(800) 789-2223

Fax:
(415) 278-5390

Mail:
ICSPAT
C/o Corporate Registration
P.O. Box 612768
Dallas, TX 75261-2768

Media Co-Sponsors:

CMP EETimes

CMP Communication Systems design

CMP Embedded Systems Programming

Gold Sponsors:

Motorola

StarCore

Silver Sponsors:

VxTel

Chameleon Systems, Inc.

Pentek

Analog Devices

BOPS

OSE

Precise Software Technologies Inc.

CMP

www.dspworld.com

Attention signal processing engineers and programmers!

Don't miss this unique opportunity to enhance your signal processing knowledge and design skills. Take advantage of the many opportunities available this fall to enhance and update your signal processing knowledge at **ICSPAT**, the premier international forum for emerging signal processing research and development.

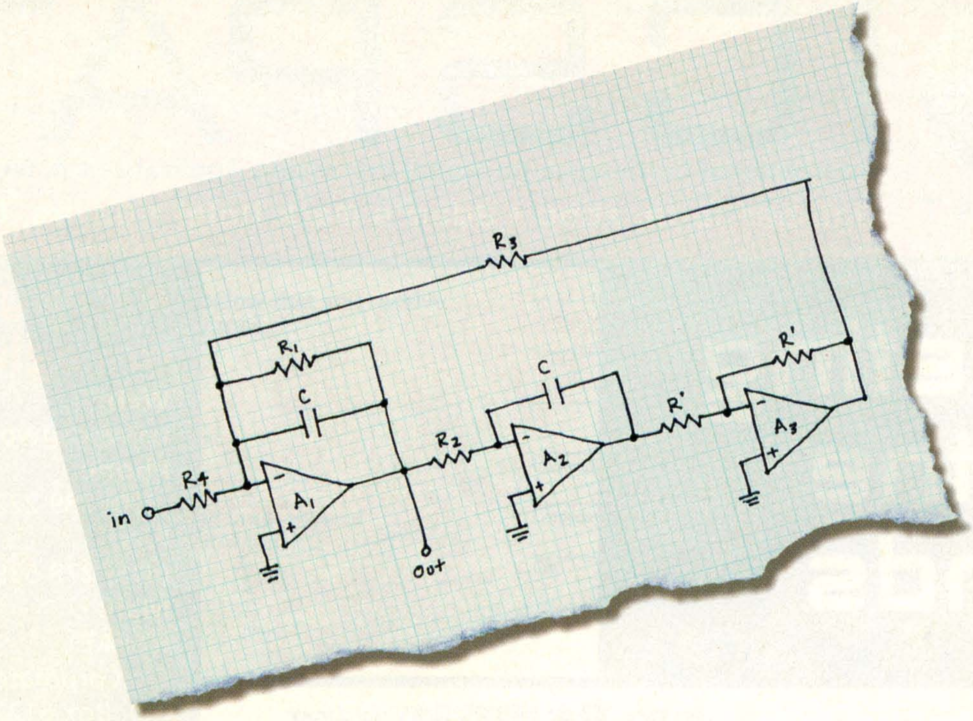
At **ICSPAT**, you can review cutting-edge research from around the world. Hundreds of white papers will be presented in more than 30 application areas including: Audio, Biomedical, Communications, Industrial Applications, Image Processing, Multimedia, Robotics, Telephony and more!

What you'll get at ICSPAT:

- Access to white papers in more than 30 application areas in the ICSPAT program
- A window on the future of signal processing and the applications in which it is used
- High quality technical knowledge and skill enhancement
- Expanded lectures offering more in-depth, enhanced coverage of topics
- Tutorial and plenary sessions as well as free product training representatives
- The official proceedings on CD-ROM

ESP

For a complete listing of ICSPAT lecture and poster abstracts, plus a comprehensive guide to plenary sessions, special events and industry activities, go to www.dspworld.com.



Analog spoken here.

In this digital world, analog engineers often tell us they feel misunderstood. When the talk around the table is in ones and zeros, they're thinking in terms of currents and voltages. They say it's almost as though if you're doing analog, you're living in a different place. And speaking a different language.

If this is *your* world, we've got some good news: now you have a place you can feel right at home. We call it Planet Analog. And it's built just for you.

Planet Analog is a place on the Web devoted entirely to analog technology. A place where you can get the latest news, in-depth technical coverage and product data. Plus articles contributed from some of the finest technologists in the world. And in coming months, we'll add seminars, chats, polls, conference registration and a lot more. All under the leadership of Steve Ohr, regarded by many as the leading analog advocate in the world today.

So if you're an analog engineer, come to www.planetanalog.com for a place to call your own. You'll find we speak your language.

SPONSORED BY:

Texas Instruments

Analog Devices

National Semiconductor

EDTN NETWORK PARTNERS

EBN, EET, SBN, Embedded.com, ISD Magazine, Power Designers, Home Toys, Electronic Design, Microwaves & RF, EE Product News, Wireless Portable, Wireless System Design, PCN Alert, ChipCenter, Communications System Design, Design & Reuse, Circuits Assembly, HDI, PC Fab, Printed Circuit Design

www.edtn.com

EDTN network

It's all right here.

Jack G. Ganssle

Momisms

Momism (môm iz' em) n. 1. *A brief statement of a principle passed maternally.* 2. *A tersely worded statement of an observation of truth*: APHORISM.

The image of mom gently guiding her young ones down paths of righteousness, teaching them the basic elements of being civilized, helping with school work, is powerful indeed. Yet we still leave home ill-equipped for real life; college itself does but a poor job in preparing us for careers and adulthood. Perhaps mom should have taught us some more lessons. Here are a few thoughts.

Look here if you want to know what Moms have to offer the embedded world. You'll also benefit from some of Jack's tips.

Interrupts

Keep ISRs short. Debugging interrupt service routines is tough and, in some cases, almost impossible. Too often those expensive tools work poorly or not at all inside an ISR. Breakpoints fail because they operate at human speeds, while the interrupts come much faster. Single stepping, the old standby of many developers, just won't work where interrupts arrive at any sort of reasonable rate. Single step in a section of code where interrupts are reenabled, and you'll likely debug different instantiations of the ISR with each step. An emulator with trace will capture the service routine's execution, but even the largest trace buffers fill quickly from loops and recursion.

A very wise friend taught me the fundamental rule of debugging ISRs: don't. Keep the routine so short, so simple, that you can debug by inspection. A good rule of thumb is to limit ISRs to a dozen or so lines. Worst case, keep them shorter than a page. If the ISR really must do a lot of work, why not spawn a task that handles the complexity?

Avoid NMI. Non-maskable interrupt, also known as Trap, level 7, or any of a number of monikers, can't be shut off, ever. Other interrupt inputs succumb to the "disable" instruction, and generally turn off automatically when a hardware-initiated interrupt occurs. Until you explicitly turn the interrupt back on, the unavoidable non-reentrant parts of the ISR are safe. An NMI handler, however, is never safe. Non-reentrant code will be destroyed if the interrupt recurs. Many CPUs use an edge-sensitive input for this beast, so the slightest bit of noise can create multiple false NMIs over the course of a few microseconds. And debugging tools, like emulators, often couple small bits of spurious noise into the target system. Reserve NMI for one-time events like power failure or the apocalypse.

Fill unused vectors. Though a CPU might support hundreds of interrupt sources, each one defined by an entry in the dispatch table, we rarely use more than a handful. If you leave those unused dispatch table entries blank, any weird vectoring will crash the application horribly, leaving no trail of evidence to the root cause.

Why would spurious interrupts occur? Maybe the hardware is defective or glitchy—it's a prototype during development, isn't it? Perhaps you've misprogrammed one of the hundreds of registers inside of today's too complex peripherals.

Better to fill all unused vectors with a pointer to a debug routine that either logs the erroneous interrupt or reaches a lurking breakpoint.

Listen—don't interrupt others. You'll learn far more listening than talking, and the listener never puts his foot in his mouth.

Bugs

Inspect rather than debug. Bottom line: code inspections find bugs some 20 times more efficiently than debugging by test. Inspect the code, design, specs, and all relevant design documents to find problems before writing/debugging/testing and then chucking a lot of expensive firmware. Inspections won't find all of the problems, but a well-implemented inspection process wrings out 70% to 80% of the defects for a fraction of the cost.

Studies indicate that in many systems 50% of the code never gets tested. It's difficult at best to devise test conditions for every error condi-

Embedded Systems Conferences

April

ESC San Francisco

conference: April 9 –13, 2001
exhibits: April 11 –13, 2001
Moscone Center

July

ESC Chicago

conference: July 9 –11, 2001
exhibits: July 10 –11, 2001
Navy Pier

September

ESC Boston

conference: Sept. 4 – 7, 2001
exhibits: Sept. 5 – 7, 2001
Hynes Convention Center

November

ESC Europe

conference: Nov. 7 – 9, 2000
exhibits: Nov. 8 – 9, 2000
Maastricht, Netherlands

www.embedded.com
esc@cmp.com
1.800.789.2223

CMP

tion/exception handler, and for IFs nested five deep. Since post-compile error rates run around 5%[1] (five bugs per 100 lines of code), even a small system with 10,000 lines of code might have 250 lurking bugs after it's completely "tested." Though devising better tests is surely a good idea, inspections will bring most of these hidden problems to light.

Inspect approximately 100 to 200 lines of code per hour. There's a sweet-spot at 150 lines of code per hour where inspections proceed very efficiently yet unveil most of the defects.[2] Monitor the inspection rate. These numbers, which come from data I've been accumulating for a number of years, suggest that inspections cost (assuming there's no benefit!) approximately $2 per line of code, or 10% of the usual $15 to $30 per line cost for most commercial firmware.

Track debugging time. It sure is fun to crank code. But in many organizations a large portion of a project's time gets consumed in the debugging phase. This is a certain sign of dysfunctional development. It indicates that the developers either write the code carelessly or spend too little time on specification and design.

Measure bug rates. A few functions or modules typically exhibit most of the product's defects. We've all been there. We've all worked on a function so complex, so poorly understood, and so badly coded that we're terrified of opening it in the editor. Change a single character in a comment and the code stops working. Barry Boehm, the software estimation guru, has shown that these error-prone functions, which typically represent a small percentage of the total code base, contain 60% to 80% of the errors. More compellingly, his data indicate these problem-functions eat up four times more effort than their well-behaved brethren. It behooves us to take data to quantitatively figure out which are bad, and then toss bad code and start again.

Debug proactively. Face it, you're going to have problems. The code will be far from perfect. Plan for bugs and instrument your code to find them quickly. Does your RTOS include a stack-overflow checker? Leave it enabled whilst debugging. Or seed the stack with a pattern, then stop the debugger from time to time to see if stacks are too big or too small.

Why not fill unused ROM/flash with nasty instructions, like software interrupts, that vector off to a debug routine? When code crashes it often just wanders off, perhaps into your carefully seeded ROM area. The software interrupts and associated handler will capture the crash quickly, and in safety-critical systems can bring the system to a known harmless state.

Dinkumware, Ltd.
Genuine Software
www.dinkumware.com

Dinkum C++ for GCC/Linux

Dinkum C++ for GCC/Solaris

Includes the Dinkum C99 Library

Comes ready to run with GCC 2.95.2 on either SPARC Solaris or PC Linux

Dinkum® C99 Library

A complete implementation of the ISO/IEC 9899:1999 Standard C Library. The Dinkum C99 Library has rich math and locale support, and has the extensive additions required to conform to the C++ Standard.

*Dinkumware & Dinkum are Registered Trademarks of Dinkumware, Ltd. Solaris & SPARC are Registered Trademarks of Sun Micorsystems , Inc.

Custom C++ Code Compliance Analysis since 1984!

CODEFIX® Version 2.0

C++ CODE MAPPING

CODEFIX Version 2.0 is a programmable tool for modifying all C and C++ source code on a file or project basis.

- Obfuscation and shrouding for C++ programs.
- Insertion of code for dynamic runtime testing.
- Database generation from C++ source code.

PCYACC® Version 9.0

PROFESSIONAL LANGUAGE DEVELOPMENT TOOLKIT

Includes "Drop In" Language Engines for SQL, dBASE, POSTSCRIPT, HYPERTALK, SMALLTALK-80, C++, C, PASCAL, PROLOG, FORTRAN, COBOL, BASIC, SGML, ASN, RPG, REXX, PL1, SNA, RTF, VISUAL BASIC, SQL2, DB2, VHDL, HTML, VMRL, JAVA, ODMG-ODL/OQL, SQL3, MODULA-3, DELPHI, VBS, and ADA.

- Portable Object Oriented Classes for C++ and JAVA – Error, Symbol Table, Syntax Tree, Yacc, and Lex.

CODECHECK® Version 9.0

SOURCE CODE ANALYST

Includes "Drop-In" Rules for Compliance analysis, Adherence to specifications, Measures of complexity, Silent error detection, Code maintainability, and Portability.

- **Maintainability** – CODECHECK identifies and measures complex, sloppy, and hard to maintain code.
- **Compliance** – CodeCheck allows your corporate coding and project specification standards to be completely automated for compliance validation.

30 day Money back guarantee! Free AIR Shipping anywhere in the world!

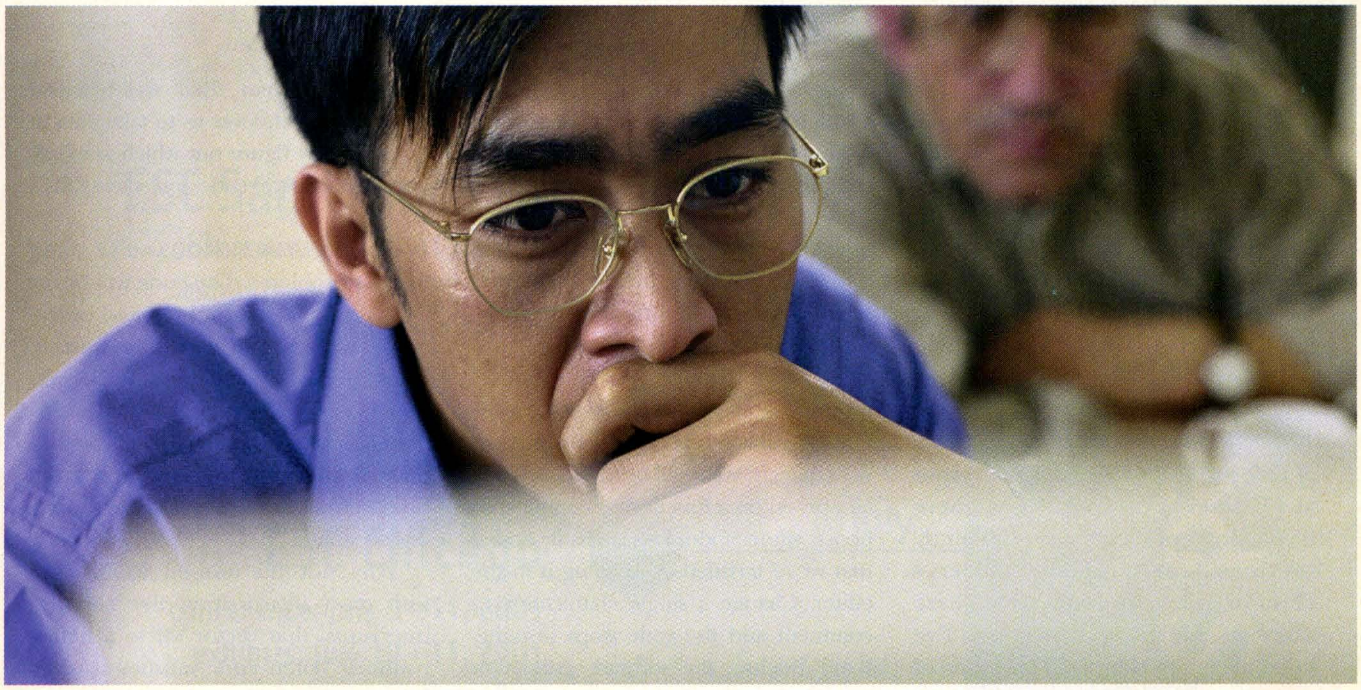
www.abxsoft.com

ABRAXAS™ Software, Inc.
4726 S.E. Division St.
Portland, OR 97206

TEL (503) 232-0540
FAX (503) 232-0543
E-Mail: sales@abxsoft.com

To Order Call
1-800-347-5214

DOS $495, MAC $495, LINUX $495, WIN 2000 $995, UNIX $1995.



We interrupt this frustrating web search for

EDA tools and design information

to bring you the following announcement...

Now you have one place on the Internet where all the latest design automation news and product information is just a click or two away. It's www.eedesign.com. And it draws on the resources of the leading information providers, analysts and manufacturers in the business.

At EEdesign, you'll have instant access to the latest news coverage from *EE Times'* Richard Goering, Michael Santarini and Peter Clarke. You'll get the inside story on solving real-world design problems from the peer experts in *Integrated System Design.*

And speaking of engineering experts, you'll also have an exclusive direct connection to John Cooley's DeepChip.com site, home of the fiercely independent ESNUG email newsletter. That's something no other EDA portal site can offer!

Plus we're featuring Toolwire, creator of the Design Chain Management Network, as our partner for web-based electronics design.

You'll also have access to an industry-wide interactive design tools product directory. What's more, in coming months we'll add seminars, bulletin boards, polls and a lot more. All under the direction of Tets Maniwa, regarded by many as the leading EDA advocate in the world today.

So if you're looking for EDA tools and information and want it *right now*, log on to www.eedesign.com. It just might be the most productive design decision you've ever made!

www.eedesign.com

EDTN NETWORK PARTNERS

EBN, EE Times, SBN, Embedded.com, ISD Magazine, Power Designers, Home Toys, PCN Alert, ChipCenter, Communications System Design, Design & Reuse, Circuits Assembly, HDI, PC Fab, Printed Circuit Design, Toolwire, DeepChip, EDA Connect, WebPRN (Web Product Realization Network)

www.edtn.com

EDTN network

It's all right here.

You'd be surprised how many embedded systems, those that are "done" and shipping, access memory in bizarre ways. Writing to ROM. Reading from unused memory. Though perhaps harmless, these odd behaviors indicate lurking software problems. Consider setting up unused/extra chip selects to trigger on any errant memory access, or expand your PLD decode logic to signal such problems by means of an extra output. Code that behaves unexpectedly is flawed, even when the symptoms seem benign.

Clean your room. Bugs thrive in messy places.

Performance and size

Sometimes engineering costs more than faster hardware. If you're building a million of something, production costs overwhelm development costs. That's much less true for small production runs.

Never forget that one of the "production" costs is that of the amortized engineering. If a design decision adds a month to the project, at perhaps a cost of $20,000, then the product's price must include this additional cost: $20,000 divided by the number of units made. Does an 8051 really make sense for your low-run application? Would a bigger CPU dramatically reduce development costs? Will shoehorning bytes into an undersized code space eat weeks of expensive developers' time?

A tiny CPU is, without question, the perfect choice for a huge range of applications. You just can't beat them for minimizing PCB real estate, recurring costs, and power consumption. And I've long been a proponent of distributing small CPUs around a board to handle small chores like I/O processing. But do understand the very real costs of working in a confined address space with perhaps underpowered tools and languages. Make CPU tradeoffs that minimize total system cost, from engineering through production.

Overload a CPU at your peril. A 90% loaded processor doubles development time. At 95%, figure on tripling the effort. This is hardly surprising to our intuitive understanding of programming. At one point or another, we've all battled a performance-bound system by tuning every bit of code it contains, instead of the 20% that is typically responsible for most of the real-time problems. Margins minimize engineering costs by allowing us to be a little sloppy. They let us deliver a product which is not quite tuned to perfection, avoiding the extreme costs that entails.

DOS COMPATIBLE FILE SYSTEM

ERTFS - EMBEDDED REAL TIME FILE SYSTEM

Since 1987 in Hundreds of Applications Worldwide

Selected as the File System for Several Major Embedded Operating Systems

DOS/Win95/FAT32 Compatible

Simple OS and CPU Porting Layer

Contiguous File Support

Realtime Extensions

Includes Support for IDE, Floppy, ROM/RAM Disk, PCMCIA, Compact Flash

CDROM Support Available

100% 'C' Source Code • Royalty Free

Only $4500⁰⁰

Embedded Systems Software ebs INC

Visit Our Web Site at:

www.ertfs.com

TOLL FREE **1 800 428-9340**

Outside U.S. Call 978 448 9340

email: sales@ertfs.com

Can CMX Really Put TCP/IP On My Little Ole 8-bit Chip?

Yes, Ma'am,

CMX has been doing amazing things with RTOSes and TCP/IP stacks for many years now. If you haven't visited us in a while, you are missing a lot of cool, new technology that is economical, royalty free, and comes with source code.

It might just put the sparkle back in your eyes!

CMX RTOSes and TCP/IP Stacks Support Most 8-, 16-, 32-bit Processors and DSP's.

CMX SYSTEMS

680 Worcester Road
Framingham MA 01702
Ph: (508) 872-7675
Fax: (508) 620-6828
email: cmx@cmx.com
WWW: www.cmx.com

Create a dynamic model of code size. Few of us really create a meaningful estimate of ROM/flash needs. Instead, we tend to ask for as much ROM as possible, or perhaps double the amount used on the previous project. It's tough to estimate binary sizes when starting a large project.

But we can't abdicate our responsibility to monitor code growth. Telling the boss a month before delivery—when the hardware design is cast in PCBs—that we need more ROM is a sure path to career stagnation.

It's a simple matter to build a spreadsheet that lists all of the modules the system will contain with estimates of their size (in lines of code, function points, or any other reasonable measure). Edit in the real source line and object size of each module as it's completed. Over time you'll find a reasonable approximation to the number of bytes of code per line of C; have the model apply this to the as-yet-uncompleted portions of the code to predict final system size. Odds are you'll spot ROM shortages early on, when there's still time to take design action.

Size doesn't matter. Be content with yourself and who you are.

Reuse and maintenance

Be realistic about reuse. Reuse is hard. Good rules of thumb: Before you can develop code for reuse you must have developed it at least three times. Before you can reap the benefits of reuse you must have reused it three times. One proposal for Reagan's version of the Star Wars missile defense system, which was pegged at 100 million lines of code, was that every module had to have been used three times before being included in the system. Not a bad idea, especially for a system so difficult to test.

Avoid dependencies. Global variables are responsible for most of the evil in the world. A program infested with globals becomes non-maintainable, buggy, and a nightmare for all team members. Globals also make reuse all but impossible.

Embedded systems suffer from another dependency problem: code that talks to hardware. Encapsulate all I/O operations.

Self documenting code does not exist. Long variable names do not self documenting code make. Judicious name selection is just a part of good coding.

Comment aggressively. Any idiot can write code. Even teenaged hackers manage to crank out working software. Professionals create beautiful code that is crystal clear and a joy to maintain. Accurate, lucid comments are an important ingredient of well-written firmware. Code is nothing more than the computerese description of what's going on; comments are the human description.

Use active voice. Capitalize using standard English rules. Check your spelling. Describe concisely the goes-intas and goes-outtas, as well as what happens and why. Some enlightened programmers write all of the comments first, and then fill in the C at their leisure. The hard part, after all, is creating an accurate, documented design. The code is nothing more than a simple translation of a good design into computer-lingo.

Keep compiles clean. Don't come to the dinner table with dirty hands, and don't deliver code reeking of unpleasant warnings.

Why do we tolerate warning messages from our compiler? Firmware lives forever. When someone else opens your code five years from now for an upgrade and finds hundreds of warnings scrolling off the screen, he'll have no idea if the messages are expected or are an effect of the way he's reinstalled the tools. Maintenance is an unavoidable aspect of the software development process; he who programs without maintenance in mind is an amateur.

Keep the code strictly ANSI compliant to minimize warnings and maximize portability. Segment unavoidable deviations from the standard to separate modules which document expected unusual compiler behaviors.

Encapsulate. The OOP folks chant "encapsulation, polymorphism and inheritance." Of those three, encapsulation is the easiest and most powerful tool for building well-written, easy-to-understand code. It's equally effective in assembly, C, or C++. Bind "methods" (code that accesses a device or data structure) with the data itself.

Floss. You'll miss your teeth when they're gone. **esp**

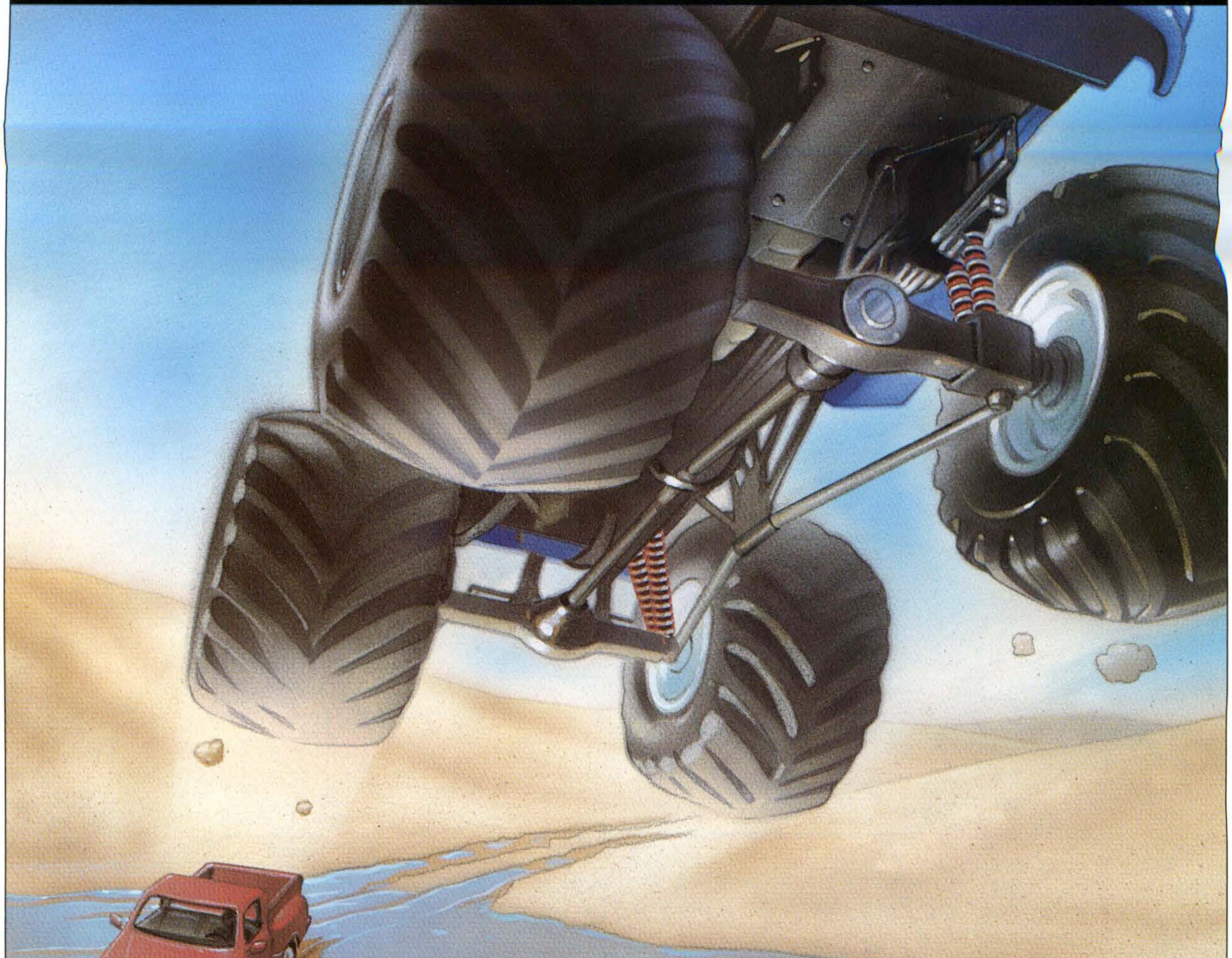
Jack G. Ganssle is a lecturer and consultant on embedded development issues. He conducts seminars on embedded systems and helps companies with their embedded challenges. He founded two companies specializing in embedded systems. Contact him at jack@ganssle.com.

References

1. Gilb, Tom and Dorothy Graham. *Software Inspection*. Reading, MA: Addison-Wesley, 1993.
2. Wheeler, David A., Bill Brykczynski, and Reginald N. Meeson, eds. *Software Inspection: An Industry Best Practice*. Los Alamitos: IEEE Computer Society, 1996.

EMBEDDED SYSTEMS PROGRAMMING (ISSN 1040-3272) is published monthly, with an additional issue published in September, by CMP Media, 525 Market St., Ste. 500, San Francisco, CA 94105, (415) 905-2200. Please direct advertising and editorial inquiries to this address. SUBSCRIPTION RATE for the United States is $55 for 13 issues. Canadian/Mexican orders must be accompanied by payment in U.S. funds with additional postage of $6 per year. All other foreign subscriptions must be prepaid in U.S. funds with additional postage of $15 per year for surface mail and $40 per year for airmail. POSTMASTER: All subscription orders, inquiries, and address changes should be sent to EMBEDDED SYSTEMS PROGRAMMING, P.O. Box 3404, Northbrook, IL 60065-9468. For customer service, telephone toll-free (877) 676-97457. Please allow four to six weeks for change of address to take effect. Periodicals postage is paid at San Francisco, CA and additional mailing offices. EMBEDDED SYSTEMS PROGRAMMING is a registered trademark owned by the parent company, Miller Freeman Inc. All material published in EMBEDDED SYSTEMS PROGRAMMING is copyright © 2000 by CMP Media. All rights reserved. Reproduction of material appearing in EMBEDDED SYSTEMS PROGRAMMING is forbidden without permission. EMBEDDED SYSTEMS PROGRAMMING is available on microfilm/fiche from University Microfilms International, 300 N. Zeeb Rd., Ann Arbor, MI 48106, (313) 761-4700.

Debugger, In-Target Probes & In-Circuit Emulators for Intel Processors



It's hard to compete without the right tools.

Let's suppose you've just been tasked to develop an important new Intel processor based system and that management wants to take it to market *yesterday*.

Now what do you do? You certainly don't have time to get bogged down using those dusty old, general purpose tools you've got sitting around.

You need a proven development system and the tech support to make the most of it. You need American Arium and the planet's best debugger. We offer...

- GNU support
- Unmatched breakpoint options
- Fast setup/ease of use
- Free upgrades and tech support for 1 year

We've got the tools you're going to need so why wait—call us today.

american arium

14811 Myford Rd., Tustin, CA 92780
Ph: 877-508-3970 Fax: 714-731-6344
E-mail: info@arium.com

For more information on any of our in-circuit emulators, visit...

www.arium.com

Compilers Assemblers Simulators RTOS In-Circuit Emulators BMDs

www.avocetsystems.com

Every Chip, Every Tool, One Source

QuickTool Finder

Select Chip

PowerPC Families
MPC5xx
MPC8xx
PPC4xx
68K Families
680x0
683xx
Motorola 8/16 Bit
68HC05/08
68HC11
68HC12
68HC16
ColdFire
6801/03
8051 All Derivatives
80C320/520
8xC751/2
C500
DS5000
8080/85
8051XA
80166
80196xx
8086/88
x86 Intel/AMD
80386EX
80186/188xx
Conexant/WDC
6502/65816
C18/19/29
Z180/64180/Z80
Z8/Z8000/Zx80
Microchip PIC
V20/30/40/50
320C20
TLCS-90

AVOCET SYSTEMS, INC

(800) 448-8500

sales@avocetsystems.com