

Embedded Systems[®]

P R O G R A M M I N G

Internet Appliance Design:

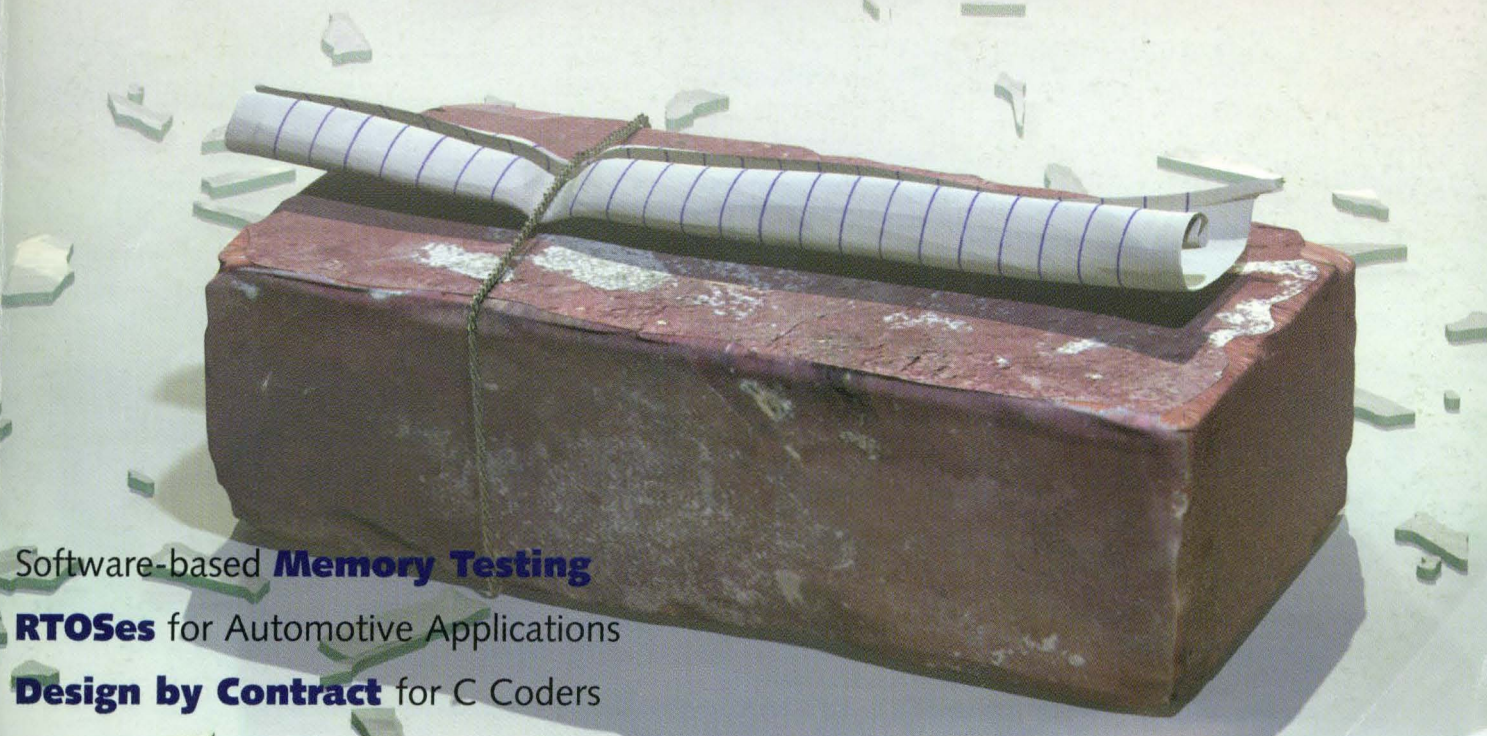


The Basics of Bluetooth

Address Resolution Protocol

Moving from PROM to Flash

Low-Cost Wireless Communications



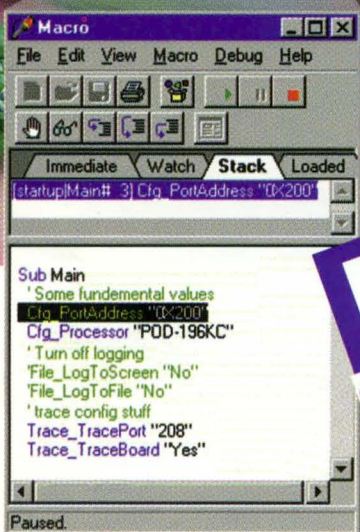
Software-based **Memory Testing**

RTOSes for Automotive Applications

Design by Contract for C Coders

Cut Development Time

Using In-Circuit Emulators and BDM's



Macros

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

Key Features

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

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

www.nohau.com
nohau

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

Final Assault

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

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

Fast. Painless. Lethal.



visionICE



visionCLICK



visionPROBE

PowerPC[™] **68K ColdFire[®]**

EST

Embedded Support
Tools Corporation

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

contents

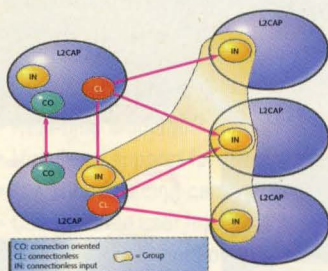
JULY 2000



COVER

Some interesting alternatives are being explored for low-cost wireless communications.

Cover illustration by Rupert Adley.



57

The Bluetooth specification is targeted for wireless communication. An overview of Bluetooth technology.

28

Software-Based Memory Testing

If ever there was a piece of embedded software ripe for reuse it is the memory test. This article shows how to test for the most common memory problems with a set of three efficient, portable, public-domain memory test functions

BY MICHAEL BARR

100

Design by Contract for C Programmers

Design by contract is a programming technique that allows the semantics of a class's behavior to be specified. This technique helps you produce more reliable software at the cost of additional formalism. Here's how to apply this traditionally object-oriented technique to procedural languages such as C.

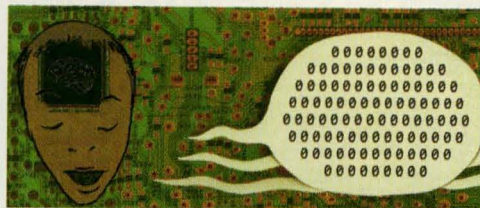
BY STEVE KAPP

108

Commercial RTOSes for Automotive Applications

Automotive powertrain applications have tough real-time scheduling requirements. Here's an approach for evaluating commercial RTOSes for such applications.

BY STEVE TOEPPE AND SCOTT RANVILLE



28

Hunt down common memory problems with software memory test.

internet appliance design

41 **CONNECTING...** Mid Year's Resolutions

The address resolution protocol provides a necessary bridge between physical and logical addresses on a TCP/IP network.

BY MICHAEL BARR

57 **Cover Story** Bluetooth Basics

Since the release of the Bluetooth v. 1.0 specification last year, the race has been on to provide Bluetooth-enabled products. Here is an overview of Bluetooth technology, its unique capabilities, and its advantages over competing wireless technologies. It's followed by an in-depth discussion of the Bluetooth link layer and Service Discovery Protocol.

BY REBECCA SPAKER

75 From PROM to Flash

Moving from PROM to flash isn't easy. Here's a list of required steps and best practices for storing executable code in a flash memory device and making it upgradable in the field.

BY STEVE SUMNER

95 Embedded Internet Tools

departments

5 **#INCLUDE** A Message from Space

BY LINDSEY VEREEN

128 **NEW PRODUCT GALLERY**

130 **RECRUITMENT**

132 **MARKETPLACE**

136 **ADVERTISER INDEX**

columns

7 **PROGRAMMER'S TOOLBOX** Rants, Accolades, and Minimization

BY JACK W. CRENSHAW

97 **PROGRAMMING POINTERS** Arrays as Function Parameters

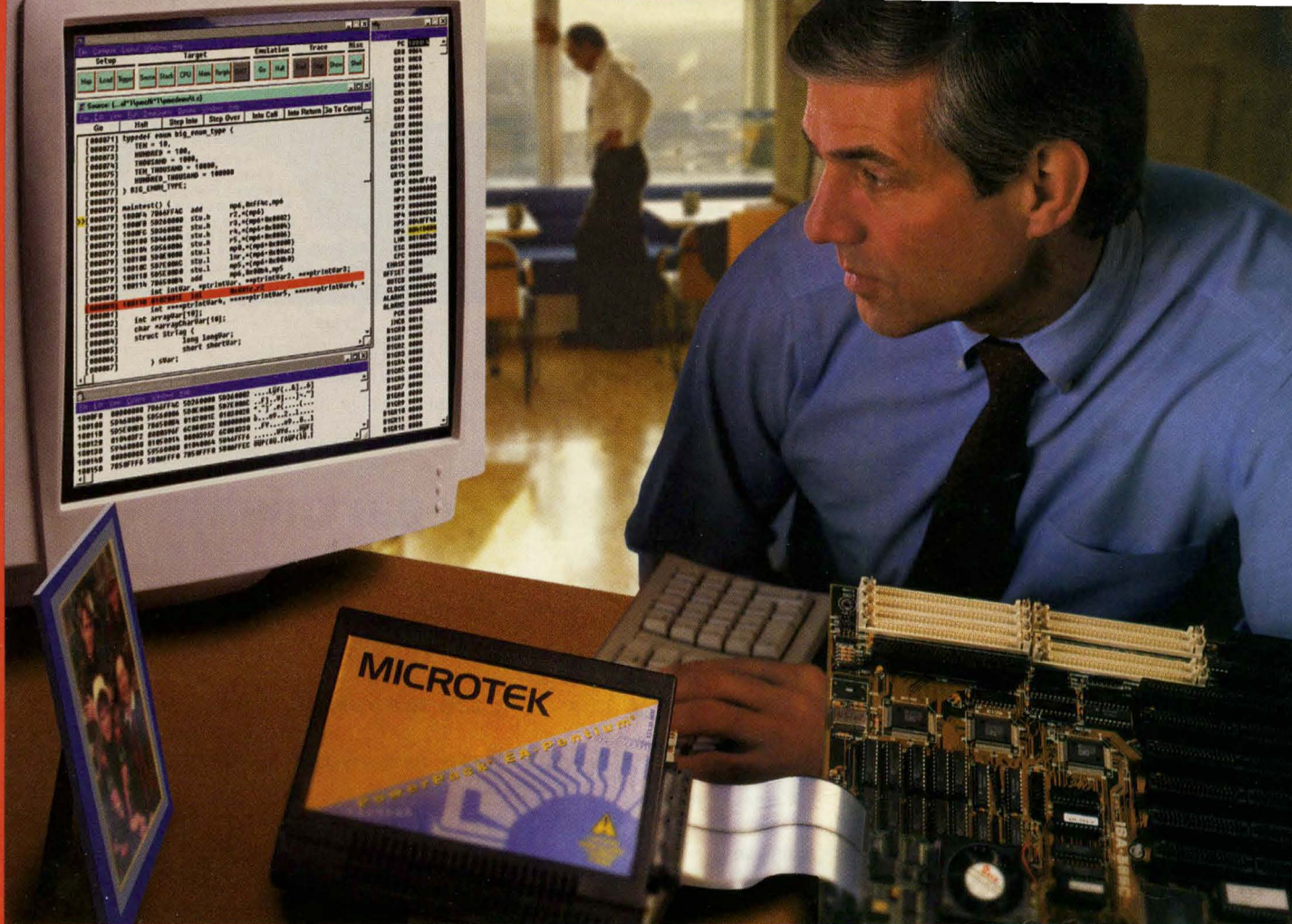
BY DAN SAKS

125 **SPECTRA** The PID Filter

BY DON MORGAN

139 **BREAK POINTS** The Story of Language

BY JACK G. GANSSE



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

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

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

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

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

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

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.

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.

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



Lindsey Vereen

EDITORIAL DIRECTOR

Lindsey Vereen, lvereen@cmp.com

MANAGING EDITOR

Felisa Yang, fyang@cmp.com

TECHNICAL EDITOR

Michael Barr, mbarr@cmp.com

SENIOR SPECIAL PROJECTS EDITOR

Tarita Whittingham, twittingham@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

VICE PRESIDENT/ELECTRONICS

Donna J. Esposito

EMBEDDED/DSP GROUP DIRECTOR

Mike Flynn, (415) 278-5251

PUBLISHER

Eric Berg, (415) 278-5220

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 ASSISTANT

Molly Bruns, (415) 278-5298

WESTERN ACCOUNT EXECUTIVE

Sam Louis, (415) 278-5223

PRODUCTION COORDINATOR

James Whitehead

CIRCULATION MANAGER

Jennifer Schuler

CIRCULATION DIRECTOR

Susan Harper

SUBSCRIPTION CUSTOMER SERVICE

Toll free: (877) 676-9745

(847) 291-5215

esj@cmp.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

PRESIDENT/ELECTRONICS

Steve Weitzner



Visit our Web site at
www.embedded.com

A Message from Space

A key factor in the development of the Lunar module for Apollo 9 was "test, test, and more test," according to the HBO miniseries "From the Earth to the Moon" (Part 5, "Spider," 1998). That's not a bad guideline in an engineering effort that demanded so many things to be accomplished that had never been done before. Given that precedent, what went wrong with the Mars polar lander, which disappeared last December and presumably crashed into the red planet? Various stories have circulated, some pointing to gross management incompetence, but whatever scenario you believe, the one commonality among the stories is that the system was not properly tested before launch.

"There was inadequate software design and testing. The software should have been designed to prevent premature engine shutdown," said former Lockheed Martin executive Tom Young who presented the NASA report on the polar lander's fate. "In space, one strike and you're out." (See www.cnn.com/2000/TECH/space/03/28/lander.report.02/index.html.)

Test not only allows you to identify and fix faults, but more importantly, it gives you the data you need to feed back into and improve the process so as to prevent future failures. In that respect, the space program is atypical since the products are more or less one-offs, which makes it more difficult to fix the process. Test, especially systems integration test, necessarily takes place at the end of the development cycle, and when intermediate deadlines slip, the pressure mounts to shrink the test phase to get the product to market on time. If your company has a cash-flow problem, the temptation to shortchange test can be strong. In non-mission critical applications, that can result in annoyances for customers, who sometimes find them-

selves involuntarily recruited into doing final test in the course of using their purchased software, which just goes to justify the admonition against buying version 1.0 of any software package. In safety-critical applications, shipping inadequately tested products is not an option.

Hardware system quality is at an all-time high, but that's not where the current challenge is. The increase in the software content of systems coupled with shorter design cycles means that producing bulletproof software is more difficult than ever. Since safety-critical designs can't rely on the customer doing final test, the development process had better assure software as well as hardware quality. Thorough testing is key to gaining that assurance.

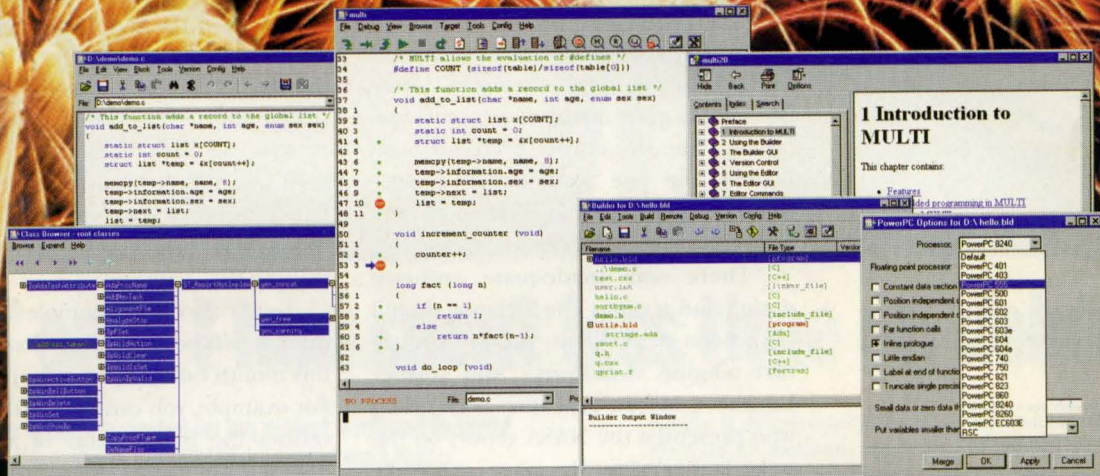
Test is also more complex than you think it will be, as Michael Barr's article this month on memory test, um, attests. For example, you can develop a test that verifies the functionality of a memory chip but still cannot detect if the chip is missing from the board.

Despite its importance, test is neither glamorous nor a career toward which engineers generally aspire. In electronics manufacturing companies, test engineers get less respect than do software engineers. The need to test and improve the process is one we seem to have to relearn periodically. High-profile, expensive failures, such as the loss of the Mars lander, serve to remind us of this valuable lesson.

lvereen@cmp.com

MULTI[®] 2000

The New IDE for the New Millennium



- New and Improved GUI
- Graphical Browser
- On-Line Help
- Syntax Coloring and Auto Indenting

- Simulator
- Version Control
- C++ Debugging
- EventAnalyzer

- Project Builder
- Code Coverage
- Run-Time Error Checking
- Profiling

The Best Compilers.

The Best Debuggers.

Now, all wrapped up in a brand new, powerful, easy to use IDE. Once you try MULTI 2000, you won't want to struggle with "old fashioned" tools ever again.

Contact us now for a
Free Evaluation CD and

start the millennium
with a bang!



Tel: 805.965.6044 • Fax: 805.965.6343 • Email: sales@ghs.com • www.ghs.com

Copyright ©1999 Green Hills Software, Inc. MULTI is a registered trademark of Green Hills Software, Inc.



Jack W. Crenshaw

Rants, Accolades, and Minimization

I'd like to start out this month by apologizing for missing last month's column. I know how much some of you readers count on the latest update, and believe me, no one is more aware than I how much trouble it causes when a column is skipped. That's especially true when we're in the midst of a series, as we are now. In fact, this series on function minimization promises to set an all-time record for the longest-running series in a column that's not supposed to have series in the first place.

I know how anxious most of you are to get this minimization thing behind us, so we can get on with other topics. I'm anxious too, because there are so many other exciting things I have in the queue to talk about, and they're all waiting for me to get past Brent's algorithm. Even on the subject of minimization, we still have the entire world of multivariate optimization to deal with, including the popular Simplex method, steepest-descent, and all kinds of good and important stuff. So, again, my apologies for the miss. My bad.

Speaking of bad—before getting into the meat of the column, I have more news from the Good Guy/Bad Guy front. I suppose by now, those of you who took my advice and went to the Mathtools Web site to download the Matlab clone, Mideva, learned the same thing I did: they've been bought out by The MathWorks. The MathWorks says that they intend to use the Mathtools compiler, Matcom, in

future releases of Matlab. Not sure about the other stuff. But if you were planning to turn to Mideva as a low-cost alternative to Matlab, I'm afraid we're both out of luck.

On the other hand, I did download SciLab, from Inria, recently. Let me

actually work, and who provide tech support that actually supports, I completely forgot to mention one of my favorite computer programs of all time: Partition Magic (PM), by PowerQuest. Simply stated, of all the Windows programs in my impressive

Herein lies a Dell Computer horror story, the key to succesful disk partitioning, and, oh yes, more on minimization.

tell you, SciLab is for real. Also, the price is right: it's free.

Scilab was developed by the Scilab Group (INRIA-Rocquencourt Metalau Project), a French group that seems to be offering it to the world as a gift, in the same spirit as the Free Software Foundation. I've been tinkering with it lately, and it seems complete. It even comes with a Simulink-like graphic tool, though that part doesn't work to well on my computer at work, which has a plain vanilla (16-color) SVGA card.

Several people pointed out to me that I gave the wrong URL for the Scilab/Inria site. The correct URL is www-rocq.inria.fr. Check it out. I think you'll be impressed.

Partition Magic rules!

Last month while I was listing the "good guys" who sell products that

arsenal of software and shelfware (about a 50/50 split), PM may well be the only Windows program I own that actually does what it's supposed to do, every time, without failure and without error.

Good thing, too, because PM (currently in version 5.0) works its magic on...um...your hard drive partitions. It does what Microsoft's cryptic and archaic FDISK does, only moreso. In its simplest terms, PM allows you to divide your hard drive(s) into multiple partitions, just as FDISK does. The difference is, if you change your mind and use FDISK to set partitions differently (adding or deleting partitions, resizing them, and so on), you should get ready to do backup, because FDISK will reformat your drive. PM lets you do the same thing while preserving your data. Nothing else even comes close.



WindRiver®
VI 2000

How Smart Things Think for the New Millennium



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
things **smart**
think™



An Elemental Operating Systems Company Has Formed.

Roll Up Your Windows.

We all have our tales of woe. No doubt, if you've bought a new computer yourself, you have some stories of your own, and don't need to hear mine. So I won't bore you with tales of software drivers that didn't drive, or tech support folks who didn't support.

Seeing that PM is dinking around with that precious data on your hard drive, even changing from, say, FAT16 to FAT32 format, it danged well had better get things right, or your hard drive is toast. Nevertheless, that imperative doesn't stop other critical programs such as uninstall programs from messing things up. PM doesn't. Ever.

From my perspective, PM's value lies in its support for multi-boot systems. When you're partitioning a drive, FDISK allows you to create one primary partition and one extended partition (which can contain up to three logical partitions). PM lets you set up multiple primary partitions, and hide all but one. Some operating systems (I believe NT is one) can boot from logical partitions. For that matter, many can share the same partition with other OSes. An OS boot process is, after all, just another computer program, and there's no basic reason that they can't coexist. But certain OSes, notably Windows, insist on driving, and they want to be on primary partitions. PM lets you do this for more than one OS. Combine this with a boot manager such as System Commander (from Vcomm) or PowerQuest's own Boot Magic, and you have a system that can boot as many OSes as you will ever want. I've been running that way for years, with nary a problem.

I can't say enough good things about Partition Magic. Thanks, PowerQuest.

Dell forfeits game

If you're wondering why I missed last month's deadline, you don't have to look much farther than my good friends at Dell Computer. I ended up fighting with these people for no less than six weeks to get the device drivers

that should have come with the computer in the first place. When I finally got to the right person (one of their hardware tech support people), the software arrived within 48 hours. Over the six week period, three other people promised to send it. None did.

We all have our tales of woe. No doubt, if you've bought a new computer yourself, you have some stories of your own, and don't need to hear mine. So I won't bore you with tales of software drivers that didn't drive, or tech support folks who didn't support. Nevertheless, just as I felt duty-bound to warn those to whom I'd recommended Mathcad, now I feel duty bound to warn you away from Dell.

To refresh your memory, the reason I bought the Dell Dimension was because I had understood that Dell and Red Hat had formed an agreement for Dell to sell computers with Linux pre-installed. Couple my interest with Linux and my long-term fondness for Dell, and it seemed to be a match made in Heaven.

In Hell, more like. I spent most of the next six weeks with my ear glued to a telephone, stuck on perpetual hold. I waded through armies of officious and condescending customer support folks, and friendly, helpful, but ineffective tech support folks. (What good is that vaunted 24/7 support, if the limit of their skills is to tell you how to close the cupholder?) After all the he-saids and she-saids are over and done, my beef with Dell boils down to two show-stopping items:

- A Dell with Linux pre-installed is an accident waiting to happen. It might actually work, when it arrives, but have no illusions about what you're getting. You're getting a Dell with Linux pre-installed. Period. It will work only as long as

the hard drive never crashes, you never need to re-install or upgrade Linux (Red Hat v. 6.2 is now out), and never make any changes to the hardware. The Linux installation disks Dell ships with its systems are plain vanilla, Red Hat v. 6.0 Linux, and cannot restore the system back to the condition it was in when shipped. In short, the software Dell loads into the computer is different than the software they send you on disk. Good luck trying to reproduce the software as shipped

I don't know how this strikes you, but to me it's an absolute show-stopper. I've been buying personal computers now since 1974—26 years, by my reckoning. This is the first time I've bought a computer that didn't come with all the software and manuals needed to restore it to the condition it was in when shipped. I haven't decided yet which makes me the most angry: the fact that the Dell computer didn't come with the right software, or the fact that no one at Dell seems to see anything wrong with this.

- If you order a Dell with Linux installed, forget trying to get Windows device drivers. Dell's position is, Linux you wanted, Linux you got. So what's your beef?

When I bought my Dell system, I made it very plain to the salesman that I intended to run it in dual-boot fashion, using both Linux and Windows 98. At the time, he assured me that I'd get all kinds of support from Dell in doing this. Hmph. That turned out to be nothing but salesman's hype. What I got was, "Go fish." Most of those six weeks on the phone were spent trying to get Windows device drivers for the hardware Dell had installed.

Again, as I tried to explain to the Dell customer support stone-wallers, if I'd bought my system from, say, Computer City or Comp-USA, I'd have gotten everything that came with each item of hardware: manuals, device dri-

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

FUJITSU

HITACHI



Italtel

NEC

OKI



©Motorola, Inc. 2000. All rights reserved. Motorola, The Heart of Smart, DigitalDNA and the DigitalDNA logo are trademarks of Motorola, Inc. The PowerPC name is a trademark of International Business Machines Corporation and is used by Motorola, Inc. under license therefrom. 3Com is a registered trademark of the 3Com Corporation. Nuon is a trademark of VM Labs. All other logos are used by permission, and are trademarks or registered trademarks, of their respective companies.

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™

When I chose to buy from Dell, I certainly didn't intend to imply license to hold back key items as a cost-saving deal. If that's the price one has to pay to get discounts, forget the discount. I'd rather have the goodies.

ver software, even the registration cards and other lapflaps that come in the box. Heck, if I'd bought it from a local storefront OEMer, I'd probably have even gotten the boxes the hardware came in. When I chose to buy from Dell, I certainly didn't intend or imply license to hold back key items as a cost-saving deal. If that's the price one has to pay to get discounts, forget the discount. I'd rather have the goodies.

This last item is not a show-stopper like the first two, since it only applies to those who want Linux. But despite Dell's much-publicized recent partnership with Red Hat, no one in Dell's tech support or customer service departments seems to have ever even heard of Linux. Never mention the word "Linux" to any Dell service rep. If you do, you will soon find yourself talking to Linuxcare.

That's the company Dell has contracted with to provide support for Linux. The good news is, the Linuxcare folks are smart, pleasant, and helpful. They were also most sympathetic to my cause, and got in my corner in terms of trying to get Dell to come across with the right device drivers. The bad news is, they have about as much clout with Dell as I do, which is to say, none.

Also, unfortunately, the Linuxcare people know as much about Windows as the Dell people know about Linux. I gather they are mostly Unix wizards from way back (can you say, "vi"?). Neither Dell nor Linuxcare people ever heard of Partition Magic or System Commander.

Finally, you should know that Dell is replacing failed components with refurbished ones, not new ones. Their rationale is now that you have the computer, it's no longer new. It's used, and therefore only deserves a used replacement part. I'm told that Dell

isn't the only large vendor to take this view. I still think it stinks. If I buy a new car, and the engine blows the next week, no way are they getting away with putting a rebuilt engine in it (trust me on this—Ford tried and failed).

Oh, remember the Dell customer service manager? The one who promised to reimburse me for my expenses in getting support? He's been incognito ever since. Even Michael Dell seems unable to find him.

The last missing software piece—the Sound Blaster driver—arrived not long ago, nearly two months after this odyssey started. Once I got hold of the right techie, the disk arrived within 24 hours. No less than three other folks at Dell had promised to send the driver. None did, though one did manage to send a bill for \$60 (good luck collecting it, Dell).

Except for one small detail (no sound!), I'm in great shape. With the software drivers installed, I thought I might now finally be shed of Dell. Alas, 'twas not to be. Last week, an internal cooling fan failed. We thought it was the CPU fan, and installed a replacement. Turns out, it was the fan on the video card. (Video cards have fans? Who would've thought it?) Sigh. I miss my Kaypro. No fan, no noise, no problems. I've never trusted a computer that sends data to its keyboard, anyhow.

Since I first began this odyssey, I've discovered www.consumeraffairs.com. They seem to have a mission similar to mine: To identify both the good guys and the bad guys. They post horror stories from folks who feel they've been ripped off, as well as Good Guy stories about companies you can trust. You'll find Dell under their heading, "Rogue's Gallery." I'll be posting the

full story of my experience with Dell, soon. Read 'em and weep.

KVM switch

Oh, I almost forgot: I've discovered a new (to me, anyhow) gadget, that's worth its weight in gold. It's called a KVM switch, and it lets you run multiple computers using a single keyboard, video monitor, and mouse (hence the acronym). It's much like the old A-B serial/parallel switches, but it switches the different signals in parallel, so they all switch together. Talk about reducing the acreage! This is great. I currently have two computers running through it, and two more (one being the dual-boot DEC Alpha) waiting in the wings.

My particular switch is the OmniCube, by Delrin. It works very nicely. You can switch computers via a hot key sequence, so there's no need to push buttons. Check it out.

Psst! Hey, buddy! Wanna buy some video monitors, cheap?

Dual boot for dummies

Now that I've got my system set up, I seem to be the current reigning expert on dual-boot systems (at least, I'm way ahead of Dell). Getting to this point took a lot of trial and error—error you needn't commit, because I'm going to tell you how to do this thing right.

For starters, many people are now buying Microsoft Windows 98 Second Edition (Win98SE). It's the most current of the Windows series, and includes all the service releases. It's also good, as well as legal, for new systems, since it isn't an upgrade disk. (I once had so many Microsoft upgrades, I can remember having to do new installs beginning with DOS 3.2. But that's another story). Be advised, Win98SE is supposed to be for new systems, and it doesn't like to coexist with other OSes. When you install it, the first thing it will do is go out on a search-and-destroy mission. If it finds



If your oscilloscope is the window into your world,
this is the one without curtains.

from **\$2,295***

Agilent 54600 Series Oscilloscopes

- 2-channel, 4-channel,
2-scope +16 logic channels
- 60-100 MHz
- 2 MB **MEGAZoom** Deep
Memory
- 32 levels of gray scale
high-definition display
- Flexible triggering (Edge,
Pulse width, Pattern, I²C)

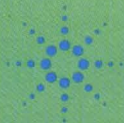
www.agilent.com/find/megazoom

1-800-452-4844,* Ext. 6847

Since the advent of mixed analog and digital designs, it seems you need more channels, more memory and triggering capability than ever to clearly see your signals. Now you can see them as never before, on the 54600 Series Oscilloscopes from Agilent Technologies.

On our Agilent 54622D model, you can watch analog and digital signals interact simultaneously, through a combination of two scope channels and 16 logic channels. And all of our 54600 Series scopes offer our MegaZoom® Deep Memory technology with 2 MB of memory per channel, so you can easily look deep into MCU-based designs. In fact, the combination of deep memory and our patented new high-definition display system uncovers signal subtleties other scopes can't begin to show you.

As for getting your tough measurement questions answered, at Agilent Technologies you can talk to another engineer just by calling our toll-free number. And of course, we'll understand if you don't just take our word for it when we tell you how these scopes can improve how well you see into your design. You'll probably want to see the scope in action. No problem. You can arrange a demo of any Series 54600 scope with no obligation. Or visit our web site for an on-line demo. Either way, we think you'll enjoy the view.



Agilent Technologies

Innovating the HP Way



DEVICE SERVER™



TECHNOLOGY



THE MOST **POWERFUL** WAY TO

INTERNET CONNECTIVITY. IT'S NO LONGER A QUESTION OF "WHY". TODAY, YOU HAVE TO FIGURE OUT "HOW". HOW CAN YOU GAIN ACCESS TO VITAL INFORMATION THAT IS PRESENTLY TRAPPED IN UNCONNECTED DEVICES? HOW CAN YOU LEVERAGE THE INTERNET TO SOLVE YOUR BUSINESS PROBLEMS SUCCESSFULLY?

PUT THE INTERNET IN

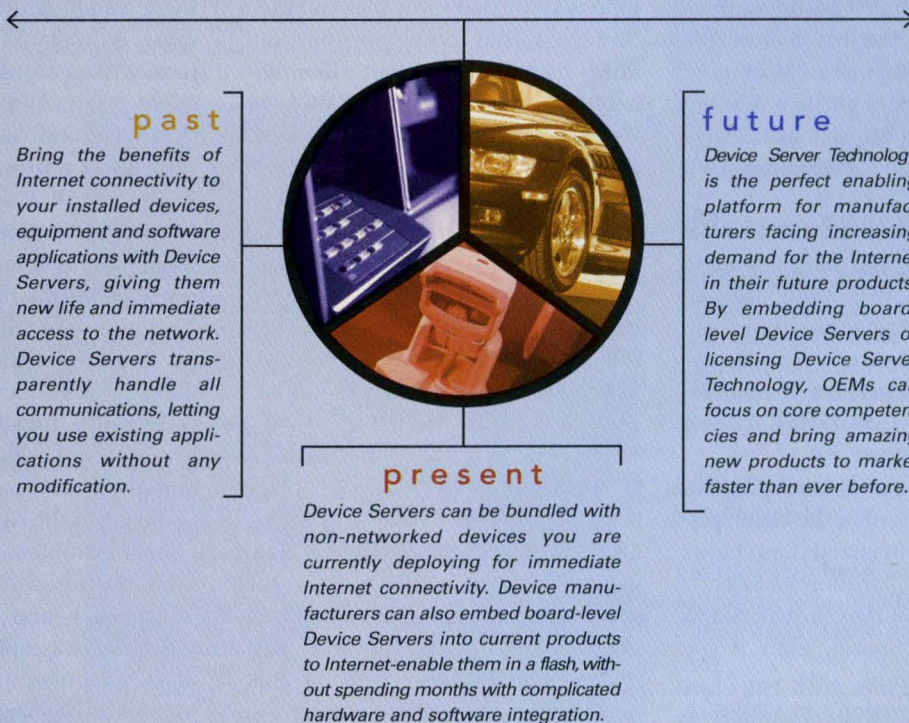
THE ANSWER IS SIMPLE – PUT THE INTERNET IN YOUR DEVICES. GIVE DEVICES THE POWER AND FREEDOM TO SHARE INFORMATION DIRECTLY OVER THE INTERNET WITHOUT THE NEED FOR BULKY, EXPENSIVE PCs. PLUS, YOU'LL ALSO HAVE THE ABILITY TO ACCESS CRITICAL INFORMATION, MANAGE DEVICES IN REAL-TIME AND CREATE INTERNET-ENABLED APPLICATIONS THAT YOU NEVER IMAGINED POSSIBLE. THE POST PC ERA HAS ARRIVED. ARE YOU READY?

DEVICE SERVER TECHNOLOGY BENEFITS

- Embed the Internet in any device
- Give devices the power and freedom to share critical information
- Fast time to market — weeks instead of years
- Save thousands of dollars in deployment and connectivity costs
- Create Internet-enabled applications that were never before possible
- Gain a new competitive edge
- Develop new business opportunities and generate new revenue streams
- Access, manage and control devices from anywhere in real-time

LEVERAGE THE INTERNET. ←

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



15353 Barranca Parkway
Irvine, California 92618 USA
949-453-3990
sales@lantronix.com



LANTRONIX®

www.lantronix.com
Europe: +31-76-565-8176
Asia: +65-447-4222

Okay, enough about computers. Let's get back to our main goal in life, which, for the moment, is to find a good, robust, general-purpose function minimizer.

any evidence of another OS, even on supposedly invisible partitions, it will methodically erase them and reformat your hard drive. All of it. It will then write its own stuff into your hard drive's master boot record (MBR), thereby ensuring that you don't repeat the infraction. If you try to overwrite the MBR, you will discover that Win98SE has vanished.

So, based upon this data, here's the drill (Dell tech support, take note):

- Install Win98SE first. Might as well, since it's all that will be there afterwards, anyhow
- After installing SE, install Partition Magic and Boot Magic. This gets you back control of your MBR
- If you want true MSDOS, you need to put it in the first partition, since it's limited to the first 2GB of disk space. To do that, use PM to move Win98SE up, and create a new primary partition below it. Install DOS and, if you like, Windows 3.11, there
- Create whatever extended and logical partitions you like. I tend to keep things in FAT16 format, which means no partition can exceed 2GB I use one partition for apps, one for data, and so on. This way, I usually need to back up only the data partition.
- If you plan to install Linux, use PM to set up and format the Linux partitions. PM understands both Linux data and swap partition formats
- Now run the Linux install. Select the "expert" option, even if you don't feel like one
- The Linux version of FDISK is called Disk Druid. However, since your partitions are already set, all you need to do in Druid is to identify the root ('/') and swap partitions. Answer the questions the installation program asks, to configure the system to your liking.

(Note: Many of these questions are anything but obvious. If in doubt, call your Linux supplier.)

Enjoy. Oh, one more thing: make sure you have all the proper device drivers for your hardware, before you begin this process. If the vendor doesn't send them with the computer, find another vendor. Also, be advised that if you have a large hard drive (larger than 8GB), MSDOS will only see drive C:. If there is any portion of the extended partition beyond 8GB, MSDOS will not see any of the logical partitions, even those that are themselves below 8GB.

Finally, be advised that, unlike Win95 or Win98, Win98SE will not see hidden partitions. Well, it's not really supposed to (that's why they call them "hidden"), but other versions did, which made it convenient when moving files between, say, the MSDOS and Windows partitions. We all know that at least the SE setup program can see the hidden partitions, because it destroys them. So how come SE itself can't see them?

Hm. On second thought, maybe it's best not to use Win98SE at all. If possible, get an earlier version and download the patches from Microsoft's Web site.

In search of Brent

Okay, enough about computers. Let's get back to our main goal in life, which, for the moment, is to find a good, robust, general-purpose function minimizer. Since our last meeting on this subject, I've been putting a lot of thought into the subject, and I think I've formulated some general, overall principles which are worth stating explicitly.

The first thought is, if you look at the kind of function we're trying to minimize, such as our now-famous test

case, shown in Figure 1:

$$f(x) = \cos(2\pi x^3) \quad (1)$$

you may be wondering what all the fuss is about. I mean, after all, it's a fairly well-behaved function. Why not just pick a place, anywhere but $x = 0$ (where the function is very flat), and slide downhill to what seems to be an obvious and well defined minimum?

This notion has two catches. The first is that we don't have a clue which way "downhill" is. Remember, we are assuming that we have only Equation 1, so we can only evaluate the function, not its slope. In truth, there's a whole family of methods that one can apply if an analytical value for the slope is available. These methods come under the general heading of Newton's methods. We'll be talking about them later on in this (seemingly never-ending) series. For now, though, we're assuming that only the function value is available, not its slope. So terms like "downhill" don't really apply in this context.

The second catch is that we don't really have a nice, pretty picture like Figure 1 at all. If we did, we could just cybernetically point our finger at the minimum, and say, "there." What we really have are only samples of the function, something like Figure 2. As far as our computer is concerned, we don't have a function at all, only discrete points, for which we may or may not be able to deduce a recognizable and predictable pattern.

To get in the right frame of mind to solve the problem, we must stop thinking of the function as the pretty curve of Figure 1, and think of it only in terms of the samples we see in Figure 2. As we choose more and more samples from the space between x_0 and x_2 , we might find a point that has a smaller y values than any previous one. We'll naturally call that point our best guess as a minimum, until an even better one comes along.

The key thought is that we must never, ever abandon our previous best

Finally

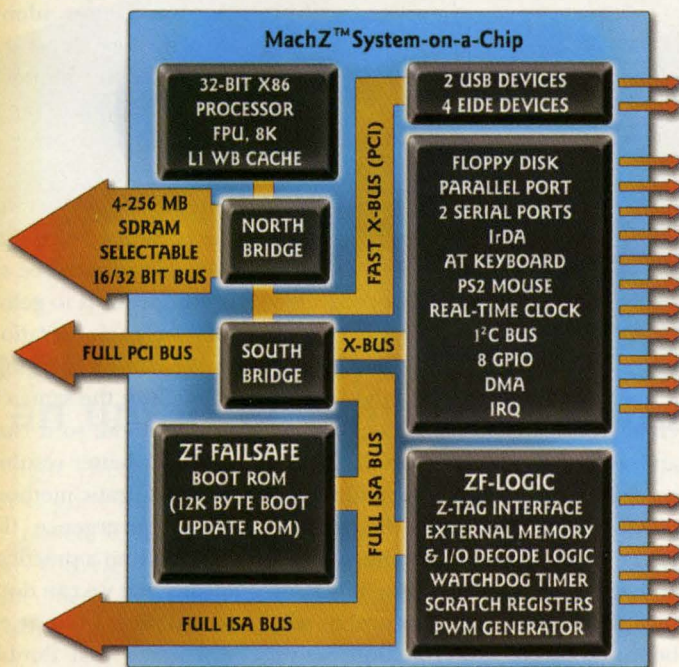
a *real* x86 System on a Chip

(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



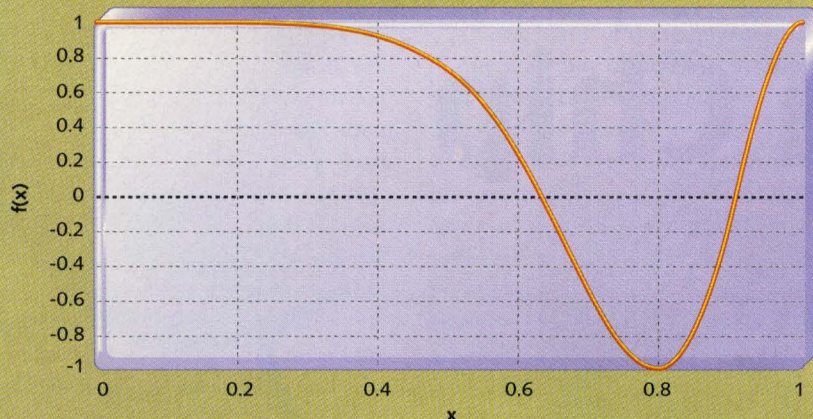
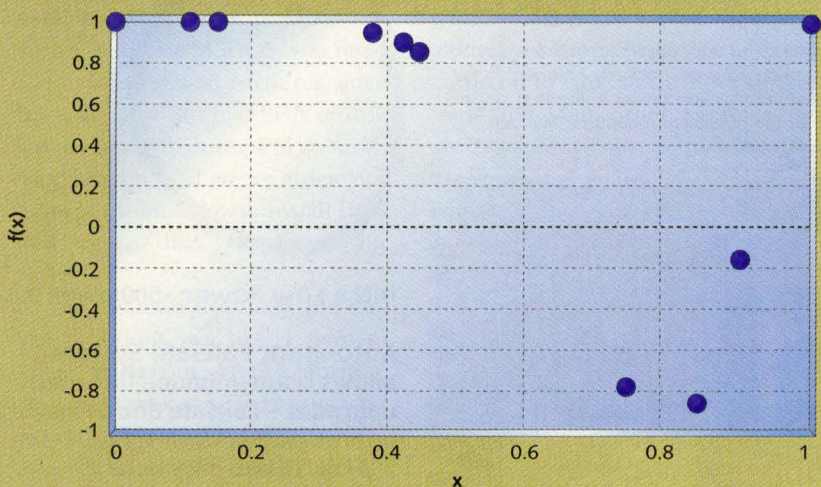
Oh, and did we mention
that it runs cold?



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

ZFLINUX
DEVICES

800.683.5943 • www.zflinux.com

FIGURE 1 The test function.**FIGURE 2** What we really see

point, until we are 100% sure that we have a better one. Think of the sampled points as stepping stones across a pond. We won't leave a nice, secure stone until we're absolutely sure we have a better one to leap to. If you think of the problem in this fashion, you will see that the best polynomial fit, be it second, third, or 100th order, is of absolutely no help at all, unless and until it actually leads us to a value for x that produces a better value for y .

It's clear that we can probe the function between x_0 and x_2 in any way we choose. We might even choose to probe

it randomly, which is what I did to generate Figure 2. Probe it enough times, and we're bound to find a value that's acceptably close to the true minimum.

Of course, if we retain the original range for the function, between $x = 0$ and $x = 1$, we can be reasonably sure that most of our random probes will be in areas of the function we've already covered, and have no chance of producing a better minimum (unless, of course, we guessed wrong about the shape of the function). We can improve our chances of finding better points by narrowing the range

of the search. As long as we have the three points x_0 , x_1 , and x_2 such that the condition,

$$f(x_0) > f(x_1) < f(x_2) \quad (2)$$

we can be sure that we have the minimum bracketed, and any further probes we make can only improve our estimate of the true minimum.

When we get down to the bottom line, then, all methods for finding the minimum without using derivatives depend on only two things:

- Probing the function space for better values
- Narrowing the range of search

How do we decide, then, whether one method is better than another? That's easy: it's the one that uses fewer probings to home in on the minimum.

So far, we've explored three methods to find this minimum: the bisection method, the golden ratio search, and the quadratic method. The first two are almost identical in performance; they depend strictly on the "divide and conquer" approach. By narrowing down the range at every step, they inexorably close the noose on the minimum. However, they require several iterations to improve the accuracy. For the method of bisection, it's easy to see that the best we can hope for is to gain one bit of accuracy at each iteration. The golden ratio search is marginally better, but it still falls in the same class as the bisection method, so it can't produce dramatically better results.

A quadratic method can, because it gives convergence that's, well, quadratic. From a practical viewpoint, this means that we can double the number of significant bits at each iteration. If you agree that doubling the bits of accuracy is better than adding just one more bit, we agree that quadratic methods are the way to go.

Understand, though, that the minimum given by a quadratic method is almost certainly not the true mini-

There is nothing Rational about paying for UML Modeling.



We hate to say it, but Rhapsody Modeler has an unfair advantage over Rational Rose

It's free. That's right, free. Of course, we also think Rhapsody Modeler is great for other reasons, but free is a pretty darn good reason to start with. So what makes Rhapsody Modeler the right choice for you? You can analyze, design, document and generate code frames for your applications using UML. It's available in C, C++ and

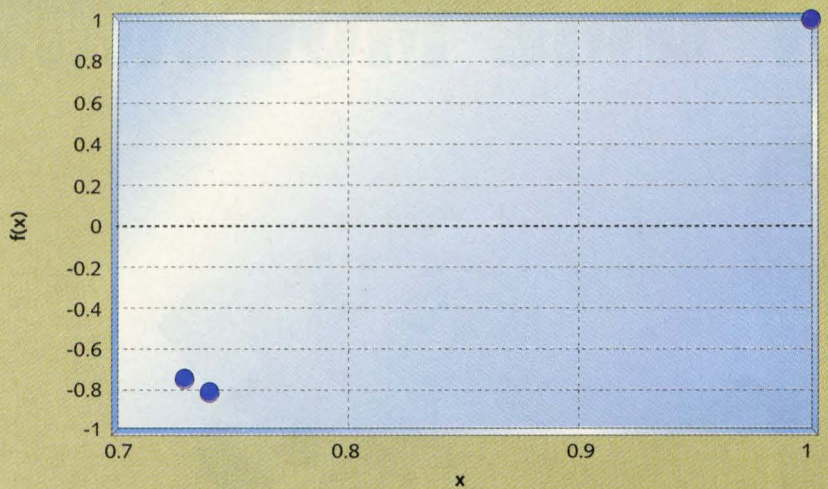
Java®. It provides a compatible upgrade to Rhapsody Validator for design validation and Rhapsody Developer for validation and full production code generation without losing any of your data. And – as if that's not enough – if you use Rose, your models can be read directly into Rhapsody Modeler. Need we say more? Oh yeah, it's free.

So what are you waiting for? Go to www.ilogix.com/modeler and sign up for your free copy of Rhapsody Modeler today.

Rhapsody®

© 2000 I-Logix Inc. Java is a registered trademark of Sun Microsystems, Inc. Rational Rose is a registered trademark of Rational Software Corporation.

I-Logix®

FIGURE 3 A bad choice

mum; it's only our best estimate of it, based upon fitting three adjacent points with a quadratic function. We should still never, ever take the formula's word for it that the estimate is a good one, until we verify it by sampling the function there.

I see two possible problems with relying heavily on the quadratic method, which is why I suggested in my last column that perhaps some combination would be more appropriate. The first problem was that I noticed that the method tends to hang onto points that are well away from the minimum. For example, one test retained the same value of x_2 for several steps. What this means to us is that the interval of search is not being reduced as nicely as we might like, and not even as nicely as a bisection method.

The second problem is, in a sense, an embarrassment of riches. Suppose, for example, that we execute three quadratic fits in a row. If our quadratic method is working well, we can expect that these fits will produce estimates for the minimum that are almost identical. Once we've evaluated the function at these points, what are we to do with the points thus produced? If the middle one also happens to be the lowest, we're in fat city. We've just nar-

rowed the range down tremendously, which is just what we'd like to do. But if the middle one is not the smallest, we'll end up throwing one of the other points away and have a situation like Figure 3. Here, we have two points very close together, and one point quite a bit further away. This is not a good situation, because remember, we don't really know the shape of the function near its minimum; we're only guessing that it's approaching a quadratic. If the shape is much different from that, we can be sure that our next estimate will not be a very good one.

Then, of course, there's always the issue of numerical accuracy. If two of our three points are close together, we're going to end up subtracting nearly equal numbers, which plays havoc with our computations. We could lose the minimum entirely, because of plain old floating point error. For these reasons, I think we need a mechanism that forbids us from having a configuration that's too lopsided. For the record, Brent's method has such a mechanism.

What if it's a parabola?

Since our last meeting, I've been trying to learn more about Brent's method. The reference given by Press, et al, in

"Numerical Recipes" is Brent, Richard P., *Algorithms for Minimization without Derivatives*, (Englewood Cliffs, N.J., Prentice-Hall, 1973). Now out of print, the book was apparently based upon Brent's PhD dissertation. The method is listed as one of several that are known to give exact results in a finite number of steps, if the function being used is, in fact, a parabola. That raises an interesting question for which I wish I had an interesting answer: even if the function is a parabola, how would we know, for sure?

You've heard me say, more than once, that any well-behaved function (that is, any function whose derivatives are continuous—mathematicians call such functions analytic) looks like a parabola near one of its extrema. The quadratic methods count on this fact to give rapid convergence in this region. Using a quadratic method, we fit a parabola through three adjacent points, and use the result to predict the location of the minimum. If the function does indeed look like a parabola, our estimate is going to be very close to the true minimum. But what if the function really is a parabola? Surely our method should not just converge asymptotically to the minimum, but should tell us in a finite number of steps that we absolutely have found it. Can our approach really tell us that? I'm not sure. It's tempting to say that if the algorithm gives the same answer for two different sets of points, it must have converged. But we have already seen a counter-example in which it does that, but we were far indeed from the correct answer. If the two end points have the same y-value, as they do in our function of Figure 1, a quadratic method will always predict the minimum halfway between them, regardless of the position of the middle point. So just getting the same answer twice in a row is no measure of correctness.

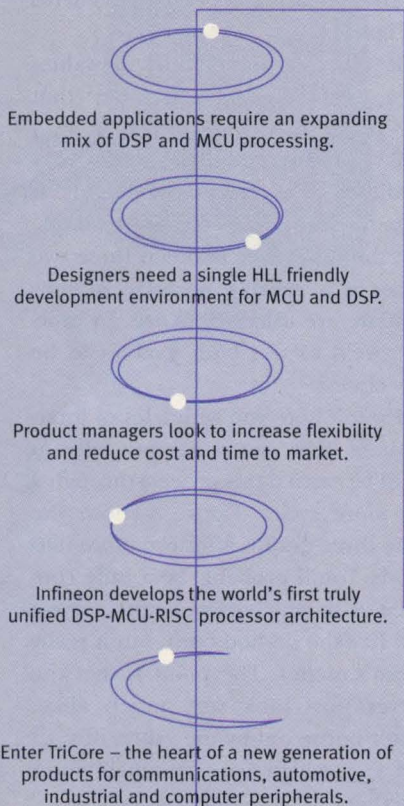
The fact is, there is no possible way we can fit a curve with only three points and discover if the data repre-

TRICORE. Three in ONE.



{PERPETUAL THINKING PROCESS}

INFINEON CYCLES



TRICORE is Infineon's award winning architecture unifying the best of three worlds – real-time capabilities of microcontrollers, computational power of DSPs, and the price/performance benefits of RISC architectures. By enabling efficient single-core DSP-MCU solutions, TriCore reduces complexity and costs, and increases flexibility. Infineon offers TriCore in a wide range of application-specific and general-purpose solutions. The core is also available through Infineon's IP licensing program. TriCore is just the latest example of what happens when a company insists on extending its creative horizons – when its people never stop thinking.

www.infineon.com/tricore

The fact is, there is no possible way we can fit a curve with only three points and discover if the data represents a true parabola or not. To do that, we need redundancy, and we can't get that with only three points.

sents a true parabola or not. To do that, we need redundancy, and we can't get that with only three points. Since we can always fit a parabola through any three points (except for pathological cases), we always seem to have a good fit. The only way we can get both an estimate of the minimum, and a confidence level for the estimate, is to use more points in the fit. I can think of two or three methods that might work:

- Use two sets of three points each, and compare the results
- Fit a cubic function through four points, and examine the third derivative
- Fit a least-squares fit through at least four points, and examine the residuals

You may recall that in the last column, we used the first option. Given four successive points, with the two middle ones both lower than the two end ones, we fitted parabolas through two sets of three. That gave us two independent estimates of the minimum, and we compared the fitted quadratics to get a measure of the goodness of fit. It might be well if we review the math for that option, here.

Fitting four points

Let the four points be P_0, P_1, P_2 , and P_3 , with the condition that:

$$f(x_0) > f(x_1) < f(x_3) \quad (3)$$

and

$$f(x_0) > f(x_2) < f(x_3)$$

We can perform two quadratic fits for the two sets (P_0, P_1, P_3) and (P_0, P_2, P_3) . We find that we can fit two functions:

$$f_1(x) = y_0 + (x - x_0)[m_3 + c_1(x - x_3)] \quad (4)$$

$$f_2(x) = y_0 + (x - x_0)[m_3 + c_2(x - x_1)] \quad (5)$$

$$m_3 = \frac{y_3 - y_0}{x_3 - x_0} \quad (6)$$

$$m_1 = \frac{y_1 - y_0}{x_1 - x_0} \quad m_2 = \frac{y_2 - y_0}{x_2 - x_0} \quad (7)$$

$$c_1 = \frac{m_3 - m_1}{x_3 - x_1} \quad (8)$$

and

$$c_2 = \frac{m_3 - m_2}{x_3 - x_2} \quad (9)$$

Differentiating Equations 4 and 5, we get the estimated minima:

$$x_{\min 1} = \frac{x_0 + x_3}{2} - \frac{m_3}{2c_1} \quad (10)$$

and

$$x_{\min 2} = \frac{x_0 + x_3}{2} - \frac{m_3}{2c_2} \quad (11)$$

The difference, of course, is:

$$x_{\min 2} - x_{\min 1} = \frac{m_3}{2} \left(\frac{1}{c_1} - \frac{1}{c_2} \right) \quad (12)$$

Don't forget, however, that this difference is not a reliable measure of the goodness of fit. If $y_0 = y_3$, m_3 must also be zero, and so will the difference between estimated minima, even if we're far away from the true minimum.

A far better measure of the goodness of fit is given by looking at the difference in the predicted y -values, which is:

$$\Delta y_{\min} = \frac{3m_3^2}{4} - \left(\frac{1}{c_2} - \frac{1}{c_1} \right) - \frac{h^2}{4}(c_2 - c_1) \quad (13)$$

where:

$$h = x_3 - x_0 \quad (14)$$

Let's see how this process works for our test function. For simplicity, I'll revert to the bisection method in choosing the x 's. We end up with the five values shown in Table 1:

Evaluating the functions using points $P_1..P_4$ gives:

$$\begin{aligned} m_1 &= -1.15231 \\ m_2 &= -3.75421 \\ m_3 &= 0.00642 \\ c_1 &= 2.317464 \\ c_2 &= 15.04253 \\ x_{\min 1} &= 0.623615 \\ x_{\min 2} &= 0.624787 \\ y_{\min 1} &= 0.671695 \\ y_{\min 2} &= -1.11776 \end{aligned}$$

Looking only at the two estimated minima, $x_{\min 1}$ and $x_{\min 2}$, we might get the impression that we are already pretty close to convergence. The true story, however, is seen by comparing $y_{\min 1}$ and $y_{\min 2}$, which are profoundly different. Just how different, we can see by comparing the two fitted curves in Figure 4.

If you're confused about the values of $y_{\min 1}$ and $y_{\min 2}$, remember that they're not the values of $f(x)$ at $x_{\min 1}$ and $x_{\min 2}$, but rather, the values of y predicted by the two parabolic fits. It certainly seems from this one example that the difference between these two values is a reliable measure of how close we are to convergence. In practice, we'd expect both y -values to be very close.

I won't bore you with a lot of intermediate steps, but rather just tell you that I've carried this process through a few more cycles. Figure 5 shows the same three graphs after two more iterations. You'll probably be a little confused as to which curve is the correct one (it's the dashed one), but it really doesn't matter. The point is that the curves now look very much alike. That's borne out by the values of $y_{\min 1}$ and $y_{\min 2}$, which are now much closer together:

Windows CE is connecting people.

VISIT OUR
WEBSITE FOR
YOUR EVALUATION
COPY OF
**WINDOWS CE
PLATFORM
BUILDER 3.0**
ACT NOW!



navic
systems

Imagine a payphone that merges voice and data—transforming an ordinary appliance into a personal communications portal where callers pay for transactions, change hotel reservations, or purchase tickets. A public phone terminal that provides detailed information about local restaurants, events and weather; sports results; stock performance and information, or bus schedules. Elcotel imagined and then built the Grapevine terminal, with the help of Navic Systems and the Microsoft® Windows® CE operating system.

Windows CE offers a wide array of connectivity options, including TCP/IP, WinSock, IP Multicast, Flash, IrDA, and IrSock. Which enables Elcotel to connect callers with the information they need. And with support for SSL 2.0 and 3.0, Windows CE helps secure financial transactions, so callers can make instant purchases safely.

Windows CE is modular so you can customize features like sound, I/O, device drivers, and file systems. That lets Elcotel scale down code for future standalone units or reuse the modules for other devices. And Elcotel can add more features as quickly as they dream them up.

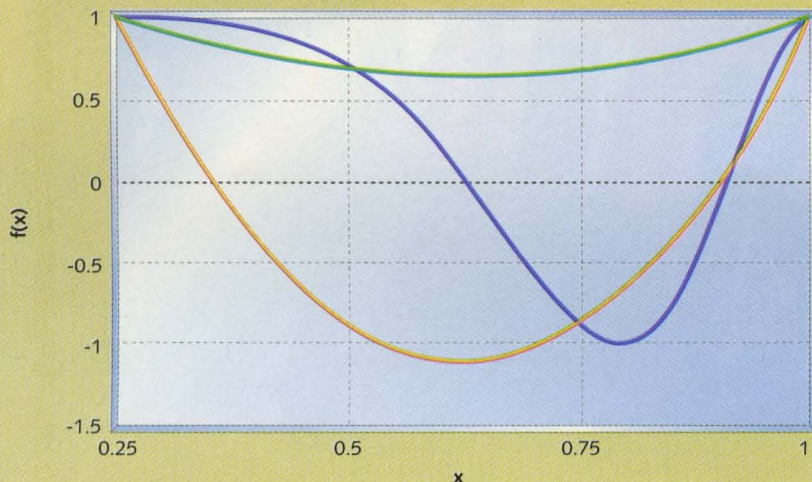
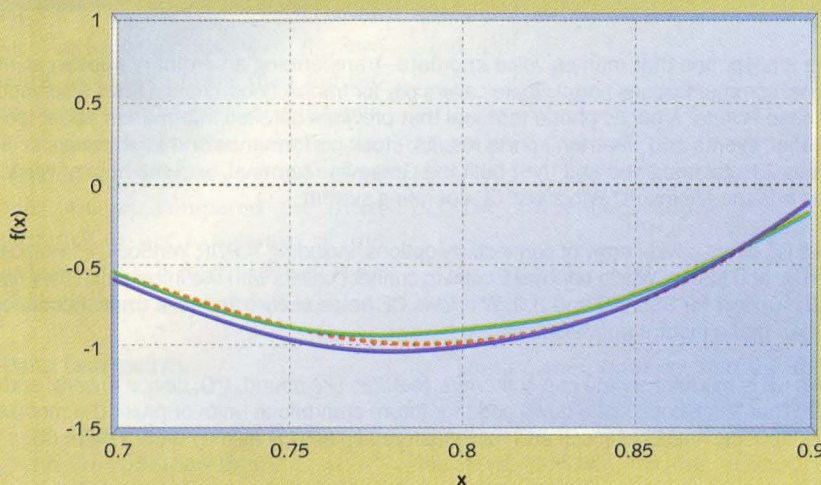
Win the race to market. Power your next generation of intelligent devices with Windows CE.

Visit www.microsoft.com/windowsce/embedded

Where do you want to go today?® **Microsoft**

TABLE 1 The starting values

	P0	P1	P2	P3	P4
x	0	0.25	0.5	0.75	1
y	1	0.995185	0.707107	-0.88192	1

FIGURE 4 A first try

FIGURE 5 Getting closer


$$y_{\min 1} = -0.93911$$

$$y_{\min 2} = -1.032$$

Before leaving this line of inquiry, let me remind you that should the function being fitted just happen to truly be a parabola, we should know it instantly, or almost so, by the fact that

both functions, $f_1(x)$ and $f_2(x)$, produce identical curves.

Can this approach be fooled? In other words, can we come up with a pathological case where fitting two curves, using three of four points for each, gives the same fit even though the function is not a parabola? Off the

top of my head, I can't think of such a case. On the other hand, I'm not yet ready to state categorically that it can't happen, so I'd hate to stake the robustness of our solution on a one-shot result. Maybe we should insist that the result hold for at least two iterations.

Tinkering around

In case you haven't noticed, I've pretty much abandoned, for the moment, the desire to write a production-level C/C++ function that does what we want done. That goal went out the window when I realized that *Numerical Recipes* didn't adequately explain Brent's algorithm. At this point, we're still in the mode of trying things to see what works. As I stated in the last column, I have no doubt that a combination of Golden Section search and quadratic fit can get us the combination of robustness and speed that we seek; I just don't know what that optimal combination is yet.

For the record, I wrote a Mathcad file that alternated between Golden Section and quadratic fits. What this approach does is to take two steps of the Golden Section, generating a total of five points. Then it fits two quadratics, in a manner similar to what I've shown above, through the lowest four points. This worked very well. After seven iterations, I was getting a difference, $y_{\min 2} - y_{\min 1}$, of $3e-8$, which means convergence was at hand. Is this the long-sought perfect search algorithm? Not by a long shot. But it sure isn't bad either, is it?

The thing that seems to make Brent's algorithm so popular is that it will skip all the Golden Section searches, and use quadratics exclusively, if the function is sufficiently well-behaved. I can see that this would certainly make it the fastest-converging method we can devise. I'm not yet convinced that it's robust enough. Alternating between the



©2000 SuperH and Hitachi are registered trademarks of Hitachi Semiconductor of America. All other trademarks are property of their respective owners.

SuperH[®] microprocessors.
Your **road map** to the future. Regardless of where it's headed.



Voice



Video



Data

What's the fastest way to market? On board a fully loaded 200MHz SH-4 microprocessor. Everything you need to design your Internet appliance is mapped out and ready to drive off the lot. You get a chip that integrates the functionality of voice, data, and video in a high-performance, low power consumption environment that achieves 1000MIPS/watt. 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, SH-4 lets you hit the road running. Check out www.superh.com today.

HITACHI
Semiconductor

Built for fast times™

I have a tiger by the tail, and I'm beginning to wish I'd never even heard of Brent's method. But, fortunately or unfortunately, solving the problem of minimization in one dimension seems to be key to everything that follows.

methods is certainly robust; it keeps the search from getting confused by crazy quadratic fits, and it narrows the range of interest as only a bisection-type method can do. But, of course, it's going to take a little longer, worst case. Somewhere in between those extremes is the perfect search algorithm.

Also for the record, I tried the third approach in the bulleted list above: use a least-squares fit. Space doesn't permit sharing this with you this month, but it does present an intriguing approach. As far as I know, no one has suggested this approach to minimum-seeking, but I don't see any fundamental reason why it doesn't bear examining, except the obvi-

ous one that performing a least-squares fit is going to burn CPU cycles. Even so, the computation load is not outrageous. One must sum seven sets of four numbers, and invert a 3-by-3 matrix. The latter operation is going to cost you, if you use a typical, canned, N-by-N matrix inverter. On the other hand, the 3-by-3 case is small enough so that one can write a closed-form version that's quite fast.

After four iterations using this method, I'm already getting quite good agreement between the fitted and actual curves. When you're using a least-squares fit, the best measure of goodness of fit is, of course, the residuals, and even after only four

iterations, the residuals are down to about $1e-3$, and the error in x_{min} about the same. Both seem to be decreasing about an order of magnitude per iteration, which is great. After six iterations, convergence is nearly complete.

Okay I don't really expect an approach based upon least squares to fly, because of the computational complexity involved. Nevertheless, it's an intriguing approach. One thing you have to say for it: it will give us an absolutely unequivocal statement as to whether the function being fitted is truly a parabola. If it is, the least-squares fit gives zero residuals, and that's a difficult result to argue with.

We'll be exploring this problem even more deeply next month. I realize you're probably getting tired of function minimization right now. For that matter, so am I. I have a tiger by the tail, and I'm beginning to wish I'd never even heard of Brent's method. But, fortunately or unfortunately, solving the problem of minimization in one dimension seems to be key to everything that follows. I have so much to go over with you yet, including all the multivariate minimization methods like the Simplex method, steepest descent, modified Newton's method, and so on. With the exception of the Simplex method, which seems to be in a class all its own, all of the best implementations of multivariate searches have a good one-dimensional engine beating in their hearts.

Once we get this method down pat, lots more good stuff is waiting in line, so be patient with me as we beat Dr. Brent into submission. Thanks.

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 holds a PhD in physics from Auburn University. Crenshaw enjoys contact and can be reached via e-mail at jcrens@earthlink.net.

Sounds Great!

Super Multimedia Power in a Fast Embedded Platform



PCM-9574 Features

- Intel Celeron™ or Pentium® III socket 370 processor
- AC97 3D stereo surround sound
- 2X AGP & 3D VGA/LCD
- 10/100 Mbps Ethernet & optional PanelLink
- One CompactFlash socket for CFC
- Software MPEG II playback
- 203mm x 146mm (8" x 5.75")

\$550

- Socket 370 CPU
- AC97 Audio
- 2X AGP 3D Accelerator
- Optional Software Modem



Automation with PCs

Advantech Technologies, Inc.
 Tel: 949-789-7178 Fax: 949-789-7179
 Email: EPCinfo@advantech.com

1-800-866-6008
www.advantech.com/epc

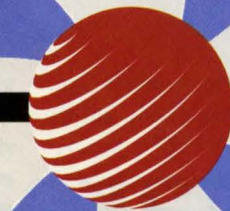
The RTOS for Internet Appliances, Systems-On-A-Chip, and all other Embedded Systems

INSPIRATION



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 sub-systems 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

© 2000 Embedded Power Corporation

Trademarks are the property of their respective holders.

MICHAEL BARR

Software-Based Memory Testing

If ever there was a piece of embedded software ripe for reuse it's the memory test. This article shows how to test for the most common memory problems with a set of three efficient, portable, public-domain memory test functions.

One piece of software that nearly every embedded developer must write at some point in his career is a memory test. Often, once the prototype hardware is ready, the board's designer would like some reassurance that she has wired the address and data lines correctly, and that the various memory chips are working properly. Even if that's not the case, it is desirable to test any onboard RAM at least as often as the system is reset. It is up to the embedded software developer, then, to figure out what can go wrong and design a suite of tests that will uncover potential problems.

At first glance, writing a memory test may seem like a fairly simple endeavor. However, as you look at the problem more closely you will realize that it can be difficult to detect subtle memory problems with a simple test. In fact, as a result of programmer naïveté, many embedded systems include memory tests that would detect only the most catastrophic memory failures. Perhaps unbelievably, some of these may not even notice that the memory chips have been removed from the board!

The purpose of a memory test is to confirm that each storage location in a memory device is working. In other words, if you store the value 50 at a particular address, you expect to find that value stored there until another value is written to that same address. The basic idea behind any memory test, then, is to write some set of data to each address in the memory device and verify the data by reading it back. If all the values read back are the same as those that were written, then the memory device is said to pass the test. As you will see, it is only through careful selection of the set of data values that you can be sure that a passing result is meaningful.

Of course, a memory test like the one just described is necessarily

In the process of testing the memory, you must overwrite its prior contents.

destructive. In the process of testing the memory, you must overwrite its prior contents. Since it is usually impractical to overwrite the contents of nonvolatile memories, the tests described in this article are generally used only for RAM testing. However, if the contents of a non-volatile memory device, like flash, are unimportant—as they are during the product development stage—these same algorithms can be used to test those devices as well.

Common memory problems

Before implementing any of the possible test algorithms, you should be familiar with the types of memory problems that are likely to occur. One common misconception among software engineers is that most memory problems occur within the chips themselves. Though a major issue at one time (a few decades ago), problems of this type are increasingly rare. These days, the manufacturers of memory devices perform a variety of post-production tests on each batch of chips. If there is a problem with a particular batch, it is extremely unlikely that one of the bad chips will make its way into your system.

The one type of memory chip problem you could encounter is a catastrophic failure. This is usually caused by some sort of physical or electrical damage to the chip after manufacture. Catastrophic failures are uncommon and usually affect large portions of the chip. Since a large area is affected, it is reasonable to assume that catastrophic failure will be detected by any decent test algorithm.

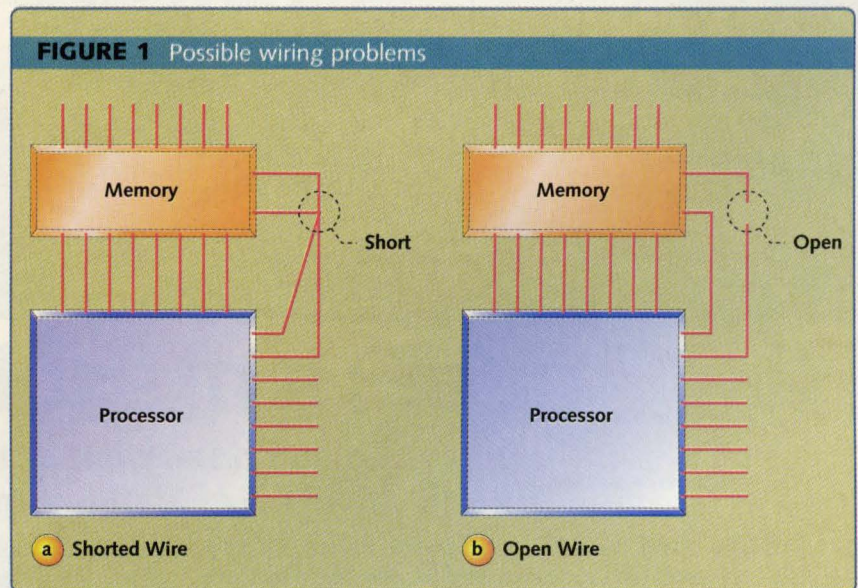
In my experience, the most common source of actual memory problems is the circuit board. Typical circuit board problems are problems with the wiring between the processor

and memory device, missing memory chips, and improperly inserted memory chips. These are the problems that a good memory test algorithm should be able to detect. Such a test should also be able to detect catastrophic memory failures without specifically looking for them. So, let's discuss the circuit board problems in more detail.

Electrical wiring problems. An electrical wiring problem could be caused by an error in design or production of the board or as the result of damage received after manufacture. Each of the wires that connects the memory device to the processor is one of three

ed or damaged in such a way that it is either shorted (for example, connected to another wire on the board) or open (not connected to anything). These problems are often caused by a bit of solder splash or a broken trace, respectively. Both cases are illustrated in Figure 1.

Problems with the electrical connections to the processor will cause the memory device to behave incorrectly. Data may be stored incorrectly, stored at the wrong address, or not stored at all. Each of these symptoms can be explained by wiring problems on the data, address, and control lines, respectively.



types: an address line, a data line, or a control line. The address and data lines are used to select the memory location and to transfer the data, respectively. The control lines tell the memory device whether the processor wants to read or write the location and precisely when the data will be transferred. Unfortunately, one or more of these wires could be improperly rout-

If the problem is with a data line, several data bits may appear to be “stuck together” (for example, two or more bits always contain the same value, regardless of the data transmitted). Similarly, a data bit may be either “stuck high” (always 1) or “stuck low” (always 0). These problems can be detected by writing a sequence of data values designed to test that each data

pin can be set to 0 and 1, independently of all the others.

If an address line has a wiring problem, the contents of two memory locations may appear to overlap. In other words, data written to one address will actually overwrite the contents of

another address instead. This happens because an address bit that is shorted or open will cause the memory device to see an address different than the one selected by the processor.

Another possibility is that one of the control lines is shorted or open. Although it is theoretically possible to develop specific tests for control line problems, it is not possible to describe a general test for them. The operation of many control signals is specific to either the processor or memory architecture. Fortunately, if there is a problem with a control line, the memory will probably not work at all, and this will be detected by other memory tests. If you suspect a problem with a control line, it is best to seek the

advice of the board's designer before constructing a specific test.

TABLE 1 Consecutive data values for the walking 1's test

00000001
00000010
00000100
00001000
00010000
00100000
01000000
10000000

LISTING 1 Data bus test

```
typedef unsigned char datum;      /* Set the data bus width to 8 bits. */

datum
memTestDataBus(volatile datum * address)
{
    datum pattern;

    /*
     * Perform a walking 1's test at the given address.
     */
    for (pattern = 1; pattern != 0; pattern <= 1)
    {
        /*
         * Write the test pattern.
         */
        *address = pattern;

        /*
         * Read it back (immediately is okay for this test).
         */
        if (*address != pattern)
        {
            return (pattern);
        }
    }

    return (0);
}

/* memTestDataBus() */
```

Missing memory chips. A missing memory chip is clearly a problem that should be detected. Unfortunately, due to the capacitive nature of unconnected electrical wires, some memory tests will not detect this problem. For example, suppose you decided to use the following test algorithm: write the value 1 to the first location in memory, verify the value by reading it back, write 2 to the second location, verify the value, write 3 to the third location, verify, and so on. Since each read occurs immediately after the corresponding write, it is possible that the data read back represents nothing more than the voltage remaining on the data bus from the previous write. If the data is read back too quickly, it will appear that the data has been correctly stored in memory—even though there is no memory chip at the other end of the bus!

To detect a missing memory chip the test must be altered. Instead of performing the verification read immediately after the corresponding write, it is desirable to perform several consecutive writes followed by the same number of consecutive reads. For example, write the value 1 to the first location, 2 to the second location, and 3 to the third location, then verify the data at the first location, the second location, and so on. If the data values are unique (as they are in the test just described), the missing chip will be detected: the first value read back will correspond to the last value written (3), rather than the first (1).

Improperly inserted chips. If a memory chip is present but improperly inserted in its socket, the system will usually behave as though there is a wiring problem or a missing chip. In other words, some number of the pins on the memory chip will either not be connected to the socket at all or will be connected at the wrong place. These pins will be part of the data bus, address

P L A N E T



M I C R O C H I P

20 VOLTS

SELF-PROGRAMMING

PICmicro FLASH

IN-CIRCUIT DEBUGGER

TWO-SPEED OSC

10-BIT A/D

Get More in a Flash

Discover the power and flexibility of
an 8-bit PICmicro[®] RISC MCU with FLASH.

Light-up your designs! The newest generation of PICmicro FLASH MCU devices offer the ultimate in programming flexibility. The possibilities are endless when you incorporate self-programming and two-wire In-Circuit Serial Programming[™] over the entire voltage range, *without* external components. These bright PICmicro stars feature an operating voltage from 2V, 10-bit A/D converter with up to 8 channels, RS-485 type



Low-cost
In-Circuit Debugger

USART, up to 256 bytes high-endurance EEPROM data memory and up to 5 MIPS performance. For added flexibility, the PICmicro MCU Migratable Memory[™] path gives you socket compatible OTP, ROM and FLASH MCUs...and a design without limits. Add world-class development tools and technical support and you've got the most complete 8-bit RISC MCU solution with FLASH. Call or visit today for your free PICmicro FLASH PowerPak.

1-888-MCU-MCHP



1-888-628-6247

Explore the Universe of Embedded Control
at www.microchip.com

The Microchip name, logo, PIC, PICmicro and The Embedded Control Solutions Company are registered trademarks and Migratable Memory and In-Circuit Serial Programming are trademarks of Microchip Technology Inc. in the USA and other countries. © 2000 Microchip Technology Inc. All rights reserved. All other trademarks are the property of their respective owners.



MICROCHIP
The Embedded Control Solutions Company[®]

bus, or control wiring. So as long as you test for wiring problems and missing chips, any improperly inserted chips will be detected automatically.

Developing a test strategy

Before going on, let's quickly review the types of memory problems we

must be able to detect. Memory chips only rarely have internal errors, but, if they do, they are probably catastrophic in nature and will be detected by any test. A more common source of problems is the circuit board, where a wiring problem may occur or a memory chip may be missing or improperly

inserted. Other memory problems can occur, but the ones described here are the most common.

By carefully selecting your test data and the order in which the addresses are tested, it is possible to detect all of the memory problems described above. It is usually best to break your memory test into small, single-minded pieces. This helps to improve the efficiency of the overall test and the readability of the code. More specific tests can also provide more detailed information about the source of the problem, if one is detected.

I have found it is best to have three individual memory tests: a data bus test, an address bus test, and a device test. The first two tests detect electrical wiring problems and improperly inserted chips, while the third is intended to detect missing chips and catastrophic failures. As an unintended consequence, the device test will also uncover problems with the control bus wiring, though it will not provide useful information about the source of such a problem.

The order in which you execute these three tests is important. The proper order is: data bus test first, followed by the address bus test, and then the device test. That's because the address bus test assumes a working data bus, and the device test results are meaningless unless both the address and data buses are known to be good. If any of the tests fail, you should work with the board's designer to locate the source of the problem. By looking at the data value or address at which the test failed, he or she should be able to quickly isolate the problem on the circuit board.

Data bus test

The first thing we want to test is the data bus wiring. We need to confirm that any value placed on the data bus by the processor is correctly received by the memory device at the other end. The most obvious way to test that is to write all possible data values and verify that the memory device stores each one successfully. However, that is

RTOS^{Win32} with REAL realtime

Phar Lap SoftwareTM

making our mark on the embedded

WORLD



TNT
EMBEDDED
TOOLSUITE[®]
10.0

- Edit, compile, link and debug with MS Developer Studio
- TCP/IP - Winsock API featuring SNMP, Wininet and MicroWeb Server
- Knowledgeable technical support
- Comprehensive engineering services and training programs
- Realtime GUI
- Extensive sample code and printed documentation
- Small footprint

Call for
a FREE
GUI demo

TEL: (617) 661-1510
FAX: (617) 876-2972
www.pharlap.com
info@pharlap.com



Phar Lap Software, Inc.
60 Aberdeen Ave.
Cambridge, MA 02138

©2000 Phar Lap Software and TNT Embedded ToolSuite are registered trademarks of Phar Lap Software, Inc.
Win32 is a registered trademark of Microsoft Corporation

many processors.
many architectures.
many issues.
one perfect development environment.

ASPEX.

DEVELOP

You know the challenges facing developers today. Mixed processors, multiple processors. Make it smaller, make it cheaper, make it faster, make it better. Let ASPEX give you the tools to meet the challenges. We're experts in embedded development tools and it shows in every aspect of ASPEX.

DEBUG

All this multiprocessing and integration vastly increases the complexity of software debugging. But ASPEX can handle it, no problem. Test drive ASPEX for yourself and experience features our competition hasn't even dreamed of.

DELIVER

So make that miracle happen and meet your time to market goal. Download your free, fully functional evaluation copy of ASPEX at www.allant.com.



Allant Software Corporation 1280 Civic Drive, Suite 206, Walnut Creek, California 94596 USA
Tel: 925.944.9690 Fax: 925.944.9612 e-Mail: sales@allant.com Web: www.allant.com

LISTING 2 Address bus test

```

datum *
memTestAddressBus(volatile datum * baseAddress, unsigned long nBytes)
{
    unsigned long addressMask = (nBytes - 1);
    unsigned long offset;
    unsigned long testOffset;

    datum pattern      = (datum) 0xAAAAAAAA;
    datum antipattern  = (datum) 0x55555555;

    /*
     * Write the default pattern at each of the power-of-two offsets.
     */
    for (offset = sizeof(datum); (offset & addressMask) != 0; offset <<= 1)
    {
        baseAddress[offset] = pattern;
    }

    /*
     * Check for address bits stuck high.
     */
    testOffset = 0;
    baseAddress[testOffset] = antipattern;
    for (offset = sizeof(datum); (offset & addressMask) != 0; offset <<= 1)
    {
        if (baseAddress[offset] != pattern)
        {
            return ((datum *) &baseAddress[offset]);
        }
    }

    baseAddress[testOffset] = pattern;

    /*
     * Check for address bits stuck low or shorted.
     */
    for (testOffset = sizeof(datum); (testOffset & addressMask) != 0;
        testOffset <<= 1)
    {
        baseAddress[testOffset] = antipattern;

        for (offset = sizeof(datum); (offset & addressMask) != 0; offset <<= 1)
        {
            if ((baseAddress[offset] != pattern) && (offset != testOffset))
            {
                return ((datum *) &baseAddress[testOffset]);
            }
        }

        baseAddress[testOffset] = pattern;
    }

    return (NULL);
}

/* memTestAddressBus() */

```

not the most efficient test available. A faster method is to test the bus one bit at a time. The data bus passes the test if each data bit can be set to 0 and 1, independently of the other data bits.

A good way to test each bit independently is to perform the so-called "walking 1's test." Table 1 shows the data patterns used in an 8-bit version of this test. The name of this test comes from the fact that a single data bit is set to 1 and "walked" through the entire data word. The number of data values to test is the same as the width of the data bus. This reduces the number of test patterns from 2^n to n , where n is the width of the data bus.

Since we are testing only the data bus at this point, all of the data values can be written to the same address. Any address within the memory device will do. However, if the data bus splits as it makes its way to more than one memory chip, you will need to perform the data bus test at multiple addresses, one within each chip.

To perform the walking 1's test, simply write the first data value in the table, verify it by reading it back, write the second value, verify, and so on. When you reach the end of the table, the test is complete. It is okay to do the read immediately after the corresponding write this time because we are not yet looking for missing chips. In fact, this test provides meaningful results even if the memory chips are not installed.

The function `memTestDataBus()`, in Listing 1, shows how to implement the walking 1's test in C. It assumes that the caller will select the test address, and tests the entire set of data values at that address. If the data bus is working properly, the function will return 0. Otherwise it will return the data value for which the test failed. The bit that is set in the returned value corresponds to the first faulty data line, if any.

Address bus test

After confirming that the data bus works properly, you should next test the address bus. Remember that address bus problems lead to overlapping mem-

Automatic Unit Testing

by Michael Aivazis

If you've read this column before, you know that we are advocates of unit testing and other forms of error prevention. However, we also acknowledge that there is a major obstacle to performing unit testing: the fact that we are human. As humans, we don't like to take on tasks that are time consuming or that increase our workload. Unit testing does seem tedious at first, and those who perform the process manually are likely to give up entirely because of the time involved.

On the other hand, the fact that we are human is one of the best reasons to automate unit testing. When we perform manual unit testing, we run the risk of making mistakes simply because we are human. After all, manual unit testing is a complicated process because we must spend a good deal of time thinking about what inputs we need to test a class.

Aside from saving time and preventing human errors, the best reason to automate unit testing is to facilitate regression testing. Regression testing is a method of preventing yourself from introducing new errors to code as you make repairs. Developers often think regression testing can only apply to an entire application, but regression testing is in fact very effective at the unit level. When they find errors in a class, it is easy for developers to create a test case that will guard against that error, put the test case in their regression test suite, and run the suite.

Every time you modify your code, you should run your code against your regression test suite to make sure that your code has not "regressed" into previous errors. When you find an error, you can write a test case against it and add it to your suite. As you develop your application, your regression testing suite will grow.

There is no limit on regression testing, and the more you perform it, the greater the benefits for your code. You can even use a script to automatically pull up a class, compile it, and run it against your test suite. You'll get a clear idea of what is going on with the code, and you don't even need to be in the office when the testing occurs.

These are just a few of the benefits of automatic unit testing. Fortunately, it is possible to automate unit testing on any development platform. You can use scripts, or select from a variety of automatic tools. No matter what method you use to automate testing, you will save time as you improve the quality of your applications.

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



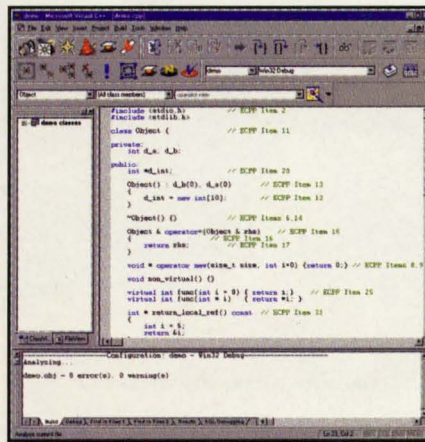
ParaSoft®

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.



CodeWizard enforces over 120 sophisticated C/C++ coding standards.

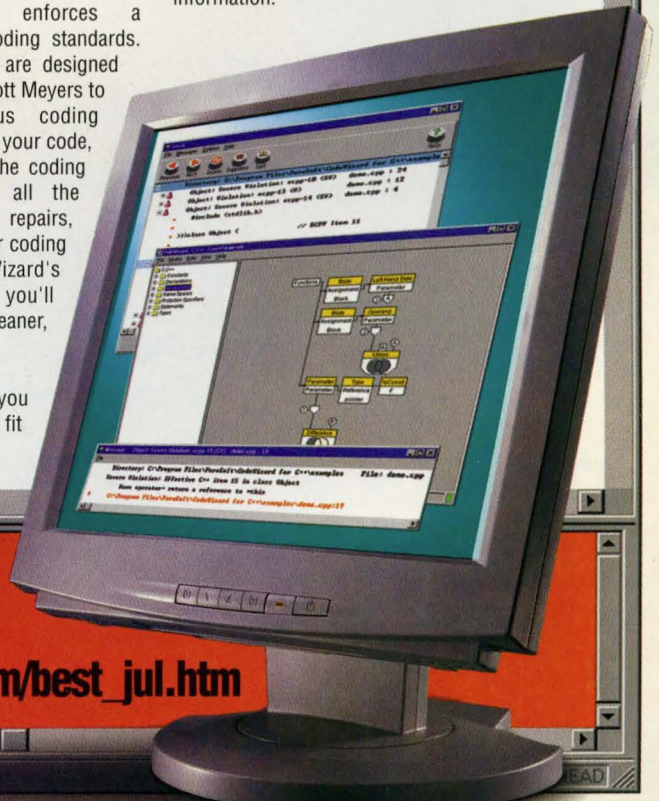
CodeWizard automatically enforces a sophisticated set of C/C++ coding standards. These language-specific rules are designed by industry experts such as Scott Meyers to help you avoid dangerous coding constructs. CodeWizard parses your code, clearly displays violations of the coding standards, and gives 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/prevention.htm, or call (888)305-0041 for more information.



**Download your
FREE DEMO at:**

www.parasoft.com/best_jul.htm

Files 2 | Runt | SQL Debugging |

TABLE 2 Data values for an increment test

Memory offset	Binary value	Inverted value
000h	0000001	11111110
001h	0000010	11111101
002h	0000011	11111100
003h	0000100	11111011
...
0FEh	1111111	0000000
0FFh	0000000	1111111

LISTING 3 Device test

```

datum *
memTestDevice(volatile datum * baseAddress, unsigned long nBytes)
{
    unsigned long offset;
    unsigned long nWords = nBytes / sizeof(datum);

    datum pattern;
    datum antipattern;

    /*
     * Fill memory with a known pattern.
     */
    for (pattern = 1, offset = 0; offset < nWords; pattern++, offset++)
    {
        baseAddress[offset] = pattern;
    }

    /*
     * Check each location and invert it for the second pass.
     */
    for (pattern = 1, offset = 0; offset < nWords; pattern++, offset++)
    {
        if (baseAddress[offset] != pattern)
        {
            return ((datum *) &baseAddress[offset]);
        }
        antipattern = ~pattern;
        baseAddress[offset] = antipattern;
    }

    /*
     * Check each location for the inverted pattern and zero it.
     */
    for (pattern = 1, offset = 0; offset < nWords; pattern++, offset++)
    {
        antipattern = ~pattern;
        if (baseAddress[offset] != antipattern)
        {
            return ((datum *) &baseAddress[offset]);
        }
    }
}

```

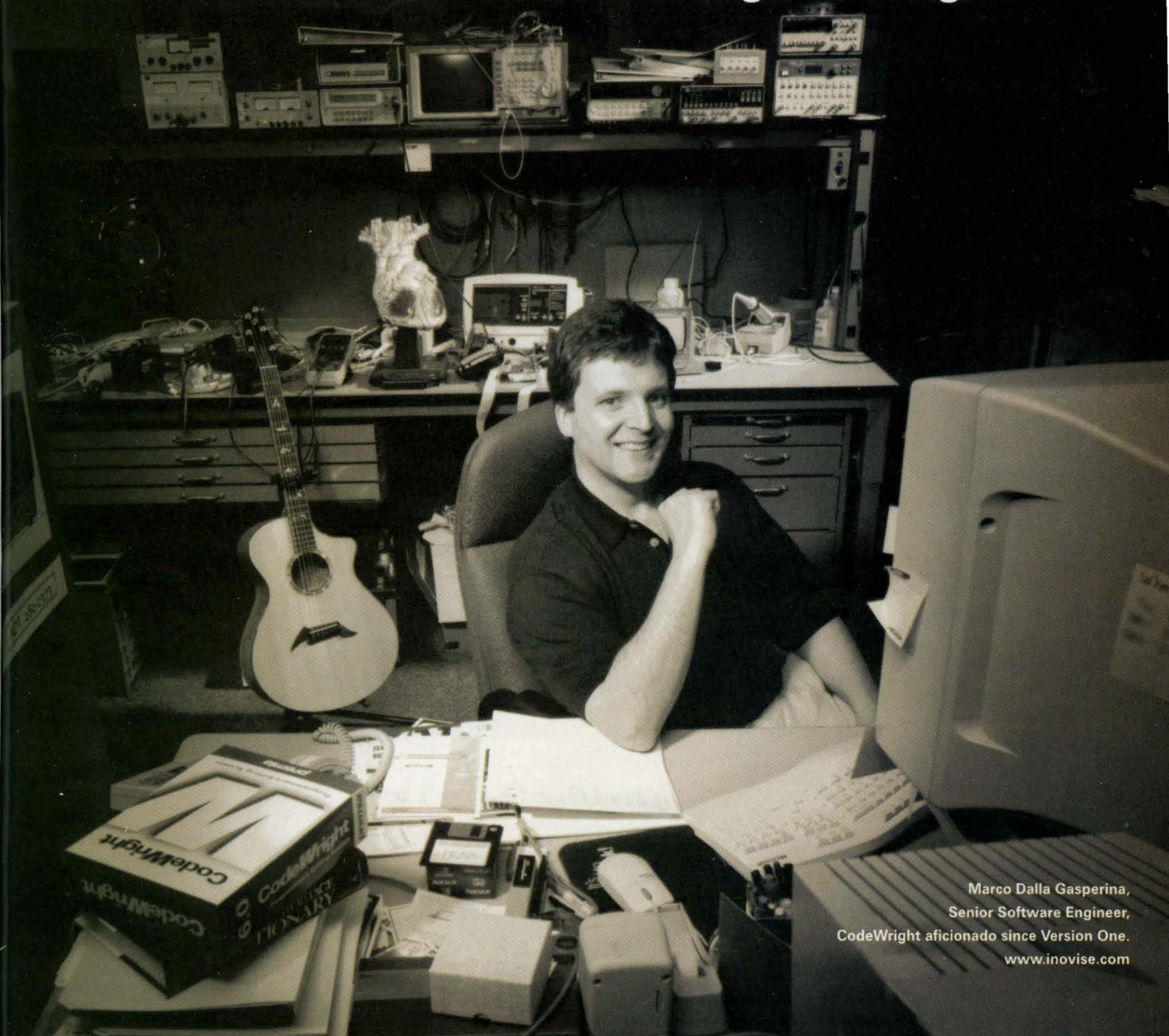
ory locations. Many possible addresses could overlap. However, it is not necessary to check every possible combination. You should instead follow the example of the data bus test above and try to isolate each address bit during testing. You just need to confirm that each of the address pins can be set to 0 and 1 without affecting any of the others.

The smallest set of addresses that will cover all possible combinations is the set of "power-of-two" addresses. These addresses are analogous to the set of data values used in the walking 1's test. The corresponding memory locations are 0001h, 0002h, 0004h, 0008h, 0010h, 0020h, and so on. In addition, address 0000h must also be tested. The possibility of overlapping locations makes the address bus test harder to implement. After writing to one of the addresses, you must check that none of the others has been overwritten.

It is important to note that not all of the address lines can be tested in this way. Part of the address—the leftmost bits—selects the memory chip itself. Another part—the rightmost bits—may not be significant if the data bus width is greater than eight bits. These extra bits will remain constant throughout the test and reduce the number of test addresses. For example, if the processor has 32 address bits, it can address up to 4GB of memory. If you want to test a 128K block of memory, the 15 most-significant address bits will remain constant.¹ In that case, only the 17 rightmost bits of the address bus can actually be tested.

To confirm that no two memory locations overlap, you should first write some initial data value at each power-of-two offset within the device. Then write a new value—an inverted copy of the initial value is a good choice—to the first test offset, and verify that the initial data value is still stored at every other power-of-two offset. If you find a location, other than the one just written, that contains the new data value, you have found a problem with the current address bit. If no overlapping is found, repeat the

"I have a lot more confidence using CodeWright."



Marco Dalla Gasperina,
Senior Software Engineer,
CodeWright aficionado since Version One.
www.inovise.com

CodeWright®

The Programmer's Editing System™

Marco is one of the team of programmers bringing early low cost detection of heart disease to reality. Inovise Medical is developing software that uses waveforms from a standard EKG test to provide a graphical view of the size and location of damage to a patient's heart.

The editor behind the code: CodeWright. When precision counts the most.

Download an evaluation copy at www.starbase.com/embedded or call (800) 491-6784.

\$299
Single User
Multi-User License available
Windows 32-bit platforms



Premia, a Starbase company

procedure for each of the remaining offsets.

The function `memTestAddressBus()`, in Listing 2, shows how this can be done in practice. The function accepts two parameters. The first parameter is the base address of the memory block to be tested and the second is its size, in bytes.

The size is used to determine which address bits should be tested. For best results, the base address should contain a 0 in each of those bits. If the address bus test fails, the address at which the first error was detected will be returned. Otherwise, this function returns `NULL` to indicate success.

Device test

Once you know that the address and data bus wiring are working, it is necessary to test the integrity of the memory device itself. The thing to test is that every bit in the device is capable of holding both 0 and 1. This is a fairly straightforward test to implement, but takes significantly longer to execute than the previous two.

For a complete device test, you must visit (write and verify) every memory location twice. You are free to choose any data value for the first pass, so long as you invert that value during the second. And since there is a possibility of missing memory chips, it is best to select a set of data that changes with (but is not equivalent to) the address. A simple example is an "increment test."

The data values for the increment test are shown in the first two columns of Table 2. The third column shows the inverted data values used during the second pass of this test. The latter represents a decrement test. There are many other possible choices of data, but the incrementing data pattern is adequate and easy to compute.

The function `memTestDevice()`, in Listing 3, implements just such a two-pass increment/decrement test. It accepts two parameters from the caller. The first parameter is the starting address and the second is the number of bytes to be tested. These parameters give the user maximum control over which areas of memory will be overwritten. The function will return `NULL` on success. Otherwise, the first address containing an incorrect data value is returned.

Putting it all together

To make our discussion more concrete, let's consider a practical example. Suppose that we wanted to test a 64K chunk of SRAM at address 00000000h. To do this, we call each of the three test routines in the proper order, as shown in Listing 4. In each case, the first parameter is the base address of the memory block. If the width of the data bus is greater than eight bits, a couple of modifications are required.

Flash ISP/IAP

Now for 16-bit MCUs and DSPs

FLASH PSD

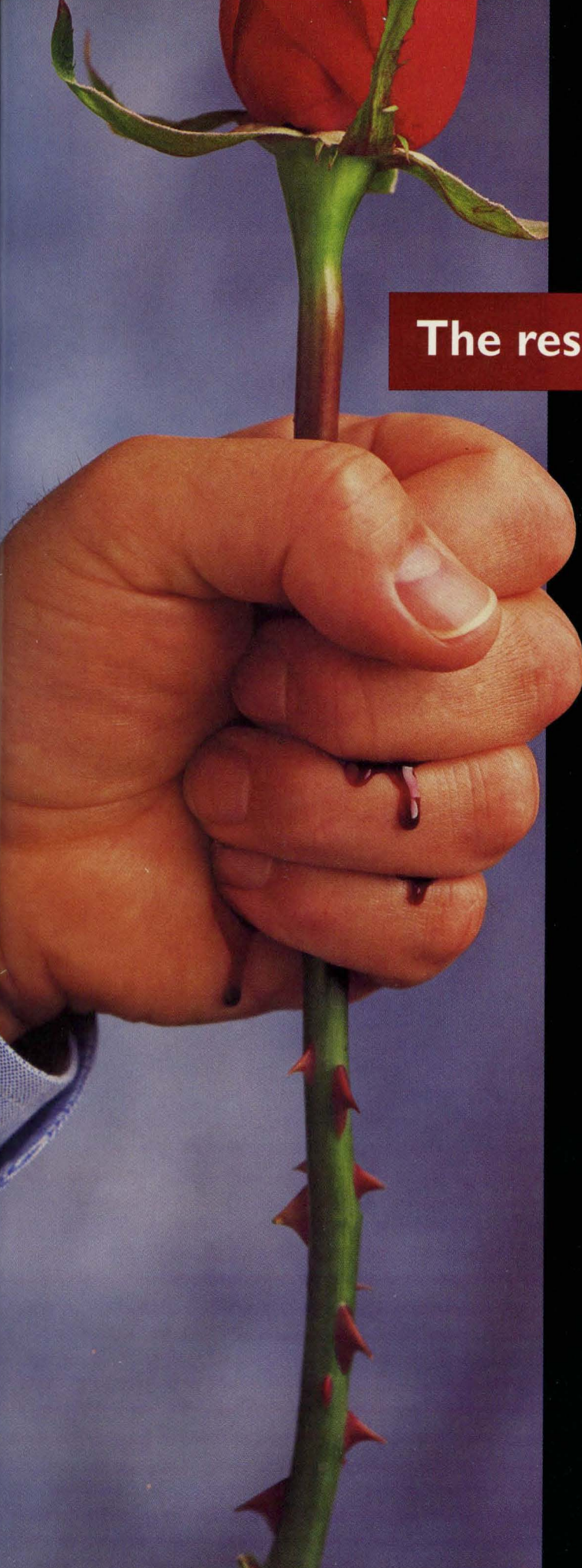
Download Free software today, and configure your embedded design with your mouse!

PSDsoft Express

Waferscale now offers 16-bit versions of our popular *EasyFLASH™* PSDs. The 16-bit PSD4000 series integrates 4Mb of Flash program store, 256Kb of concurrent Flash, 64Kb of Scratch-Pad SRAM, programmable logic, and advanced in-system programming capabilities onto a single chip. The new devices are configurable with the same easy to use, point and click development/programming software... **PSDsoft Express**. The software guides you through the entire design from configuring the MCU interface and selecting the PSD, to programming the code and firmware into the PSD...All in less than 2 hours! Visit the URL below today to learn more about the PSD4000 series, and to download your FREE copy of PSDsoft Express.

www.waferscale.com/esp.html

<p>Waferscale-USA Tel. 800-832-6974 Fax 510-657-5916 info@waferscale.com</p>	<p>Waferscale-EUROPE Tel. 33-1-69320120 Fax 33-1-69320219 wsifrance@wsiusa.com</p>	<p>Waferscale-ASIA Tel. 82-2-761-1281 Fax 82-2-761-1283 james@mail.wsiasia.co.kr</p>
---	---	---



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.



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

No developer was actually harmed or risked injury in the making of this ad.
Rational Rose is a registered trademark of Rational Software Corporation.
Real-time Studio is a registered trademark of Artisan Software Tools, Inc.

LISTING 3, cont'd. Device test

```

    }

    baseAddress[Offset] = 0;
}

return (NULL);

} /* memTestDevice() */

```

LISTING 4 A function to search the ARP cache

```

int
memTest(void)
{
#define BASE_ADDRESS (volatile datum *) 0x00000000
#define NUM_BYTES    (64 * 1024)

    if ((memTestDataBus(BASE_ADDRESS) != 0) ||
        (memTestAddressBus(BASE_ADDRESS, NUM_BYTES) != NULL) ||
        (memTestDevice(BASE_ADDRESS, NUM_BYTES) != NULL))
    {
        return (-1);
    }
    else
    {
        return (0);
    }
}

} /* memTest() */

```

If any of the individual memory test routines returns a nonzero (or non-NULL) value, you might turn on a red LED to visually indicate the error. Otherwise, after all three tests have completed successfully, you might turn on a green LED. In the event of an error, the test routine that failed will return some information about the problem encountered. This information can be useful when communicating with the hardware designer or technician about the nature of the problem. However, it is visible only if we are running the test program in a debugger or emulator.

In most cases, you would simply download the entire suite and let it run. Then, if and only if a memory problem is found, would you need to use a debug-

ger to step through the program and examine the individual function return codes and contents of the memory device to see which test failed and why.

Unfortunately, it is not always possible to write memory tests in a high-level language. For example, the C language requires the use of a stack. But a stack itself requires working memory. This might be reasonable in a system with more than one memory device. For example, you might create a stack in an area of RAM that is already known to be working, while testing another memory device. In a situation such as this, a small SRAM could be tested from assembly and the stack could be created there afterward. Then a larger block of DRAM could be tested using a better

test suite, like the one shown here. If you cannot assume enough working RAM for the stack and data needs of the test program, then you will need to rewrite these memory test routines entirely in assembly language.

Another option is to run the memory test program from an in-circuit emulator. In this case, you could choose to place the stack in an area of the emulator's own internal memory. By moving the emulator's internal memory around in the target memory map, you could systematically test each memory device on the target.

The need for memory testing is most apparent during product development, when the reliability of the hardware and its design are still unproven. However, memory is one of the most critical resources in any embedded system, so it may also be desirable to include a memory test in the final release of your software. In that case, the memory test, and other hardware confidence tests, should be run each time the system is powered-on or reset. Together, this initial test suite forms a set of hardware diagnostics. If one or more of the diagnostics fail, a repair technician can be called in to diagnose the problem and repair or replace the faulty hardware. **esp**

Michael Barr is the technical editor of Embedded Systems Programming. He holds BS and MS degrees in electrical engineering from the University of Maryland. Prior to joining the magazine, Michael spent half a decade developing embedded software and device drivers. His "Connecting..." column is a regular feature of the magazine. Michael can be reached via e-mail at mbarr@cmp.com.

References

1. 128K is 1/32,768th of the total 4GB address space.

Source

This article is adapted from material in Chapter 6 of the book *Programming Embedded Systems in C and C++* (ISBN 1-56592-354-5). It is printed with the permission of O'Reilly & Associates, Inc. The source code for these memory tests is placed into the public domain and is available in electronic form at www.embedded.com/code.html.

Small Footprint... Huge Impact!

Extending data beyond the corporate enterprise. PointBase, the first pure Java data management solution designed to support applications for the mobile wireless device market.

PointBase

Managing data anywhere, anytime.

Meet the demands of the small application environment... test-drive the PointBase data management solution with synchronization...discover the keys to high-power, small-footprint databases...pick-up your FREE 30-day evaluation copy now!

www.pointbase.com/esp

Questions? Give us a call
1.877.238.8798 toll free

Code for the Road

Application Contest!

For details go to www.pointbase.com/contest

ORCHESTRATING EMBEDDED LINUX SOLUTIONS



It's been true in music since the first duet. And today, it's just as true in the world of Linux. A technology may play beautifully, but for a solution to be complete, this technology must also play in harmony. With compelling complementary technologies. With worldwide, world-class consulting, support, and training services. Without one false note.

That's what we at LynxWorks bring to the performance. As Lynx Real-Time Systems, our company developed LynxOS® RTOS, a true virtuoso in developing embedded real-time applications and the closest RTOS to Linux today. Now, with BlueCat™ Linux, we've expanded our repertoire to embedded Linux. Embedded systems designers can use either of these

superb soloists, or tap the natural synergies of both, to speed new product development and more precisely match product and market requirements.

To mark this change, our company has taken a new name. One that integrates our past with our future. And, like a seasoned maestro, we've brought all our resources – experience, technology, and support – with us.

Don't miss a beat. Find out more about LynxWorks. The company that orchestrates embedded Linux solutions. The company that can help you hit new high notes in embedded systems design.

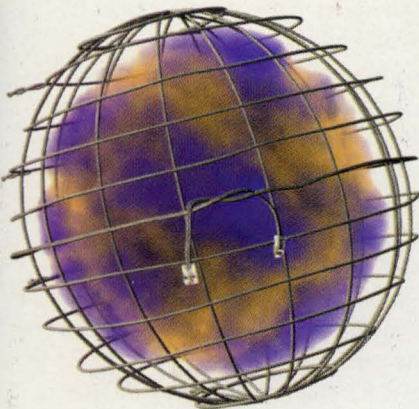
Lynx Real-Time Systems is now



www.lynuxworks.com



Michael Barr



Internet Appliance Design

in this issue

- 57** Bluetooth Basics
- 75** From PROM to Flash
- 95** Embedded Internet Tools

Mid Year's Resolutions

Every system on a TCP/IP network has two addresses, one physical and one logical. The address resolution protocol (ARP) provides a necessary bridge between these two addresses. The ARP protocol and its implementation are the subject of this month's column.

To put this discussion of ARP in the proper perspective, let's talk a little bit about where we've been and where we're headed. If you're a regular reader of this column, you know that my long-term goal is to produce a small, portable UDP/IP stack appropriate for use in all sorts of embedded systems. The complete stack should be finished, documented, and available for downloading from www.embedded.com in just a few short months.

We've already discussed that TCP support is not always required in embedded uses of Internet technologies ("TCP/IP or Not TCP/IP?" April 2000, p. 49). I don't have to convince you that there's no reason to go to the trouble and expense of including software in ROM that your application doesn't actually require. Hence, my decision to include only UDP in this stack. A UDP/IP stack was sufficient for my own use of the Internet protocols to keep a satellite gateway's firmware up to date and send logging messages to a control and monitoring station. That was several years ago, but the technique is just as useful today.

We've also seen how a network stack (UDP/IP or TCP/IP) fits into the broader embedded software framework. Once implemented, the

stack of protocols is simply another API to be called from your application program. The stack is internally dependent upon the API of the underlying operating system and network device driver, but otherwise separate from those pieces of software. It is, in effect, middleware. The ARP protocol is just one component of a TCP/IP or UDP/IP stack.

Lookup, look down

Last month we talked about the unique hardware addresses associated with each device connected to a physical network like Ethernet. I told you that these addresses must be globally unique; no two Ethernet-connected systems may have the same 6-byte hardware address. It turns out that every system on an IP network also has a second address. This logical address is called, as you might expect, the "IP address."

The best analogy I can draw to this two-address phenomenon is that of a toll-free phone number, like the 800- and 888-prefixed numbers in the U.S. Although each toll-free number is unique and can be used to contact a person at a particular physical location, the number itself does not convey any direct information about what the actual phone number is. For a while I had a toll-free number of my own; calls to that number were automatically redirected to the phone in my office. My office phone had an area code, exchange, and four more digits just like any other; my 888-number had nothing in common with it. You can't tell from a toll-free number if it will

reach the phone on your desk or one on the other side of the country.

In much the same way, each node on the Internet (or an intranet) has two addresses: one physical and one logical. Neither of these addresses contains any information about the other. And yet, unlike the phone number example, you need both addresses to communicate with a given system. Usually, your application knows just one address—the IP address—of the remote system. But no network packets can be sent to the remote system without the hardware address as well.

ARP is the Internet's lookup service. Given an IP address (toll-free number), ARP can obtain the hardware address (actual area code and phone number) to which network packets should be sent on the physical network. Similarly, a related reverse-lookup service called RARP can obtain the IP address of a machine given only

its hardware address. ARP is used by every machine on the Internet; the use of RARP is more limited.

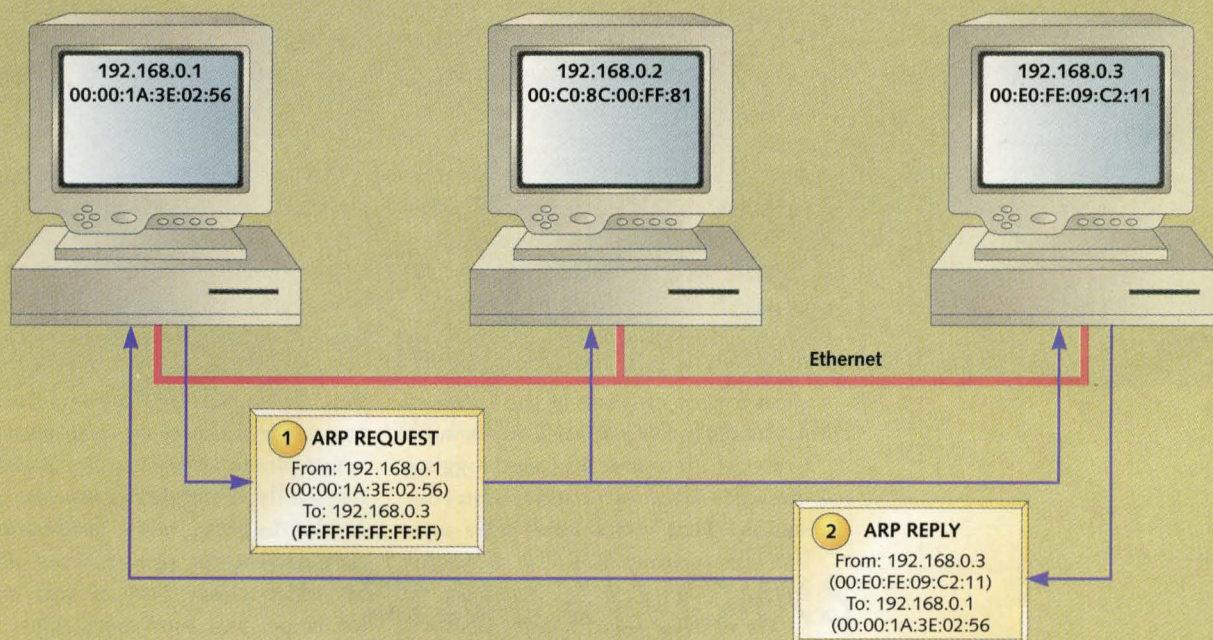
ARP, ARP, and away

Before we go on, I need to let you know that I'm restricting the remainder of this month's discussion to simple networks where all of the connected systems are on the same physical network. In other words, I'm not going to tell you how the hardware addresses obtained with the help of ARP are sometimes white lies, used in conjunction with nodes on the network called bridges, switches, and routers to direct your packets across physical network boundaries. Those details really aren't important to you anyway—unless the embedded system you are building is itself a bridge, switch, or router—since these white lies don't affect your ARP implemen-

tation. (Now just pretend like I didn't open up a whole can of worms of routing issues you hadn't thought about before and let's carry on.)

As specified in RFC 826 (way back in the early days of the Reagan administration), ARP is a general-purpose protocol that can be used to map any type of hardware address to any type of protocol address. However, for most practical purposes, all anyone really cares about using ARP for, these days, is converting the IP address of a remote machine into an Ethernet address. It's the device driver for the Ethernet controller that needs this information. Every time the IP layer passes the network driver a packet to send over the Ethernet, it needs to figure out what Ethernet address, specifically, to send the packet to. So ARP will be at the very bottom of our UDP/IP stack, residing below the IP layer but above the network driver.

FIGURE 1 How ARP works





LINEO™

Transportation Systems

- Radar Controls
- Global Positioning Systems

Consumer Devices

- Internet, Mobile and Entertainment Devices

Retail Business Products

- Credit Card Readers
- P.O.S. Systems

Internet Infrastructure

- Routers
- Modems
- Switches
- Gateways

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

My choice of μ C/OS-II as RTOS also affects my naming convention. Except for well-known names I want to mimic, all of my function and data structure names will begin with the prefix *Net*.

TABLE 1 Portable data types

Data Type	Description
INT8U	An unsigned 8-bit integer
INT8S	A signed 8-bit integer
INT16U	An unsigned 16-bit integer
INT16S	A signed 16-bit integer
INT32U	An unsigned 32-bit integer
INT32S	A signed 32-bit integer

Figure 1 shows how ARP works. In short, the system that needs a hardware address sends an ARP request message out onto the network. Since the sender doesn't know the hardware address of the system it's looking for, this message is broadcast to all systems on the physical network. (On Ethernet, address FF:FF:FF:FF:FF:FF is reserved for broadcast messages.) Included within the ARP request is the IP address (also known as, protocol address) of the target system and both of the sender's addresses. Each system that receives the broadcast ARP request checks to see if its local IP address matches the target protocol address in the ARP request. The one system with that IP address sends an ARP reply directly to the requester. Normal UDP/IP communication can begin only after the requester receives the ARP reply.

Preliminaries

Before I can show my ARP implementation code, we need to discuss a few preliminaries. The first of these is that I'm following the source code conventions of the μ C/OS-II real-time operating system (see Jean Labrosse's *MicroC/OS-II*, R&D Books, 1998). One thing you'll notice as a direct result of this is that my code uses the set of portable, compiler-independent data types shown in Table 1. These types

will be used instead of `char`, `short`, `int`, `long`, and their unsigned counterparts, whenever the size of a field is dictated by a network protocol.

So, for example, the definition of an ARP packet looks like this in my implementation:

```
typedef struct
{
    INT16U  hw_type;
    INT16U  prot_type;
    INT8U   hw_len;
    INT8U   prot_len;
    INT16U  operation;
} NetArpHdr;

typedef struct
{
    NetArpHdr  arpHdr;
    INT8U      sender_hw_addr
        [HW_ADDR_LEN];
    INT32U     sender_ip_addr;
    INT8U      target_hw_addr
        [HW_ADDR_LEN];
    INT32U     target_ip_addr;
} NetArpPkt;
```

The advantage of this should be clear enough. Each field will have the correct size (in bytes and sign) on any platform, provided the fellow doing the port of the protocol stack remembers to redefine the six basic types in Table 1 to match the compiler and underlying hardware. So, for example, a port to an 80186 processor would include the definitions:

```
typedef unsigned char INT8U;
typedef unsigned short INT16U;
typedef unsigned long INT32U
```

My choice of μ C/OS-II as RTOS also affects my naming convention. Except for well-known names I want to

mimic, all of my function and data structure names will begin with the prefix *Net*. That prefix will be followed by the module name—for example, *Arp*—and then a name descriptive of the function itself.

The second thing we need to talk about is the format of IP addresses. You might, for example, refer to your personal workstation by a dotted-decimal number such as 207.221.32.136. This is your workstation's IP address, but in a human-readable string format. For reasons you can easily understand, the protocol stack doesn't much like to deal with strings. Rather, it prefers to deal with numbers. Therefore, the protocol stack treats your IP address as a big-endian 4-byte unsigned integer. (The preceding string IP address would be treated as 0x8820DDCF.)

Just as string addresses are hard for the protocol stack to manipulate, big-endian 32-bit integers are hard for people (even programmers) to interpret. So one of the first things I did on this project was to write a pair of functions for converting string IP addresses to 32-bit integers and vice versa. Following a naming convention with which I was familiar I called these `inet_addr()` and `inet_ntoa()`, respectively. Their prototypes are as follows:

```
INT32U inet_addr(char const * str);

void inet_ntoa(INT32U ip_addr,
    char * buf);
```

The only thing to be aware of when using these is that the second function requires the caller to reserve 16-bytes of space for `buf` in advance of the call. That is the maximum length of a dotted-decimal IP address, including the null terminator for the string.

'Nuff said

With those preliminaries out of the way, we can now begin to look at the code within the ARP module. It should be obvious from Figure 1 that

TimeSys Linux/RTTM

The Power of Real-Time in Pure LinuxTM



Enjoy the freedom of Linux and the robustness of a true real-time operating system...

Brought to you by TimeSys, the leading provider of tools and services for building predictable real-time systems.



www.timesys.com
1-412-681-6899 1-888-432-TIME

LISTING 1 A function to send ARP request and reply packets

```

#define ETHERNET      1

#define PROTO_IP      0x800
#define PROTO_ARP     0x806

#define HW_ADDR_LEN   6
#define IP_ADDR_LEN   4

#define ARP_REQUEST   1
#define ARP_REPLY     2

#define ARP_PACKET_LEN    sizeof(NetArpPkt)

const INT8U broadcast[HW_ADDR_LEN] = { 0xFF,0xFF,0xFF,0xFF,0xFF,0xFF };
int
NetArpSnd(INT16U operation, INT8U * target_hw_addr, INT32U target_ip_addr)
{
    NetArpPkt * pArpPkt;

    /*
     * Allocate a network buffer.
     */
    pArpPkt = (NetArpPkt *) NETMemGet(NET_ARP, ARP_PACKET_LEN);

    /*
     * Fill in the ARP header fields.
     */
    pArpPkt->arpHdr.operation = htons(operation);
    pArpPkt->arpHdr.hw_type   = htons(ETHERNET);
    pArpPkt->arpHdr.prot_type = htons(PROTO_IP);
    pArpPkt->arpHdr.hw_len   = HW_ADDR_LEN;
    pArpPkt->arpHdr.prot_len = IP_ADDR_LEN;

    /*
     * Fill in the address fields.
     */
    memcpy(pArpPkt->target_hw_addr, target_hw_addr, HW_ADDR_LEN);
    pArpPkt->target_ip_addr = target_ip_addr;

    memcpy(pArpPkt->sender_hw_addr, local_hw_addr(), HW_ADDR_LEN);
    pArpPkt->sender_ip_addr = local_ip_addr();

    /*
     * Broadcast the request over the network.
     */
    return (NetPhySnd(broadcast, htons(PROTO_ARP), (unsigned char *) pArpPkt,
                     ARP_PACKET_LEN));
} /* NetArpSnd() */

```

there will be at least two functions: one for the sending of ARP requests and replies and another for receiving these packets and processing them. I've called these functions `NetArpSnd()` and `NetArpRcv()`, respectively.

The implementation of `NetArpSnd()` is shown in Listing 1. This function would typically be called by the IP module above or the network driver below, after it had been determined that the hardware address of the target system was not known. In that case, the call might look something like this:

```

NetArpSnd(ARP_REQUEST, broadcast,
          inet_addr("207.221.32.136"));

```

In addition, `NetArpSnd()` may also be called by the `NetArpRcv()` function, which we'll see next, in order to send an ARP reply to a remote system that requested the hardware address of the local system. In that case, the first parameter will be `ARP_REPLY` and the second and third parameters will be the `local_hw_addr()` and the `local_ip_addr()`, respectively.

The implementation of `NetArpRcv()` is shown in Listing 2. This implementation conforms to the recommendation, in RFC 826, that any valid ARP packet directed to a local system, whether it be an ARP request or reply, be examined for useful hardware addresses. If the packet is an ARP request, this function is responsible for invoking `NetArpSnd()` to send the reply.

Cacheing in

If you looked closely at the code in Listing 2, you probably noted a call to a mysterious function named `NetArpAddEntry()`. Obviously, we don't want to broadcast an ARP request onto the network and wait for a reply before sending each and every IP packet. This could seriously offset the performance of communications across the entire network. So we need a way to keep track of the hardware



**What may appear to be
a nice little embedded solution now
just may rear its ugly head later.**

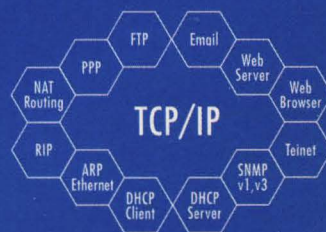


PROVEN SOLUTIONS : Market needs and competitive situations will continue to change. That's why we have provided solutions over the last ten years that are modular so you can quickly adapt and easily meet these challenges with a minimum of engineering overhead.

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 won't rear up and bite you in the end. So follow the lead of companies like ARM, Intel, Ericsson and Greenhills 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.

LISTING 2 A function to handle incoming ARP packets

```

int
NetArpRcv(int n, NetArpPkt * pArpPkt)
{
    int  retval = NET_ERROR;

    /*
     * If the ARP packet is addressed to this system, process it.
     */
    if ((pArpPkt->arpHdr.hw_type == htons(ETHERNET)) &&
        (pArpPkt->arpHdr.prot_type == htons(PROTO_IP)) &&
        (pArpPkt->target_ip_addr == local_ip_addr()))
    {
        /*
         * Add or update the sender's ARP cache entry.
         */
        NetArpAddEntry(pArpPkt->sender_hw_addr, pArpPkt->sender_ip_addr);

        /*
         * Process the ARP message.
         */
        switch (ntohs(pArpPkt->arpHdr.operation))
        {
            case ARP_REQUEST:
                /*
                 * Reply to the request.
                 */
                NetArpSnd(ARP_REPLY, pArpPkt->sender_hw_addr, pArpPkt->sender
                    _ip_addr);
                retval = NET_SUCCESS;
                break;

            case ARP_REPLY:
                /*
                 * We've already updated the ARP cache as necessary.
                 */
                retval = NET_SUCCESS;
                break;

            default:
                /*
                 * Unsupported operation (RARP?).
                 */

                retval = NET_ERROR;
                break;
        }
    }

    return (retval);
}

/* NetArpRcv() */

```

addresses we've already learned about. To do this, the ARP module typically includes a cache of the most recently used hardware addresses.

An ARP cache can be implemented in a variety of ways and there are no strict rules about doing it. In fact, nothing in the standard specifically precludes a system from asking for the hardware address of a system each time a packet is to be sent to it; so an ARP cache is not strictly necessary.

Because I want my stack to be small and simple and because I only aim to support IP-Ethernet address pairs, I've employed a pretty basic strategy for hardware address tracking. The first element of this strategy is the ARP cache itself, which is defined as follows:

```

typedef struct
{
    INT32U  ip_addr;
    INT8U   hw_addr[HW_ADDR_LEN];
} NetArpTblEntry;

NetArpTblEntry
gArpCache[NET_ARP_CACHE_SIZE];

```

where the size of the cache, `NET_ARP_CACHE_SIZE`, is a configuration option. Each entry in the cache takes up 10 bytes of RAM, so you want to limit this as much as possible. The correct number of entries depends heavily on your application. If you'll only be communicating with one other system, a cache containing just one entry would be sufficient.

Before using the ARP cache, it should be flushed or initialized with meaningless records. This can be done with the help of the function shown in Listing 3, `NetArpFlush()`. The idea here is simply to ensure that none of the data in the cache be interpreted as a valid address pair.

One important thing to note about flushing the cache is that address pairings can become "stale" over time. For example, imagine that a system you're communicating with has a hardware

WE CALM THE SEAS OF TECHNOLOGY



SO YOU
CAN RIDE
THE WAVE
OF SUCCESS

WHY PACIFIC SOFTWARES ?

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

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

Royalty Free or Royalty based licensing

End-to-End support from the Protocol Specialists

Ported and Tested Compatibility with any Processor,

Kernel or RTOS

Flexible and Scalable solutions to protect your investment

Demonstrated reduction of time to market

Leading the industry with next generation products and solutions

World-wide presence

...and, most importantly, we have Happy Customers!

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

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

California Corporate Headquarters:

tel: 800.541.9508 fax: 805.499.5512

European Headquarters:

tel: 44.1494.432.735 fax: 44.1494.432.728

Japan/Asia-Pacific Headquarters:

tel: 81.3.5669.7722 fax: 81.3.5669.7723



Pacific Softworks
our wave... your Success.

LISTING 3 A function to flush the ARP cache

```

void
NetArpFlush(void)
{
    int i;

    /*
     * Fill the ARP cache with null records.
     */
    for (i = 0; i < NET_ARP_CACHE_SIZE; i++)
    {
        gArpCache[i].ip_addr = 0;
        memcpy(gArpCache[i].hw_addr, broadcast, HW_ADDR_LEN);
    }
}

/* NetArpFlush() */

```

LISTING 4 A function to search the ARP cache for a hardware address

```

INT8U *
NetArpLookup(INT32U ip_addr)
{
    int i;

    /*
     * Search the ARP cache for a matching record.
     */
    for (i = 0; i < NET_ARP_CACHE_SIZE; i++)
    {
        if (gArpCache[i].ip_addr == ip_addr)
        {
            /*
             * Found a match, return the hardware address.
             */
            return (gArpCache[i].hw_addr);
        }
    }

    /*
     * Not found, return a pointer to the broadcast address.
     */
    return (broadcast);
}

/* NetArpLookup() */

```

failure. A new system is substituted for it and given the same IP address as the old one. But this new system will have a different hardware address. The result? Your IP packets will be sent to a non-existent hardware address and will be received by no one. You won't be able to communicate with the replacement system unless and until it happens to send you an ARP request. (**NetArpRcv()** automatically updates the ARP cache when either an ARP request or an ARP reply is received.)

One way to prevent such problems from occurring is to periodically flush the cache. The network device driver might, for example, flush the cache every 20 minutes. The next IP packet sent from the local application code to each IP address will trigger an ARP request and a fresh ARP reply. Since embedded systems tend to run for long periods of time without a reset, you should plan for the worst case and flush the cache from time to time.

A cache of hardware addresses isn't very useful without a way to do lookups. That's the purpose of the **NetArpLookup()** function in Listing 4. The guts of this routine should make perfect sense. The ARP cache is searched, linearly, until an IP address matching the one provided as a parameter is found. If a match is found, a pointer to the associated hardware address is returned. Otherwise, a pointer to the broadcast address is returned. The caller must check this return value to see if a call to **NetArpSnd()** is necessary.

Listing 5 shows the implementation of the final function in the ARP module. **NetArpAddEntry()** simply adds a new IP-Ethernet address pair to the ARP cache. If the IP address is already in the cache, that record is updated. If it is not already in the cache, the first available slot is filled in. If, for some reason, the entire ARP cache is filled, the cache is flushed and the new address pair is inserted in the very first location. The position of the entry in the cache is returned, though the caller should have no particular use for it.

BIG FEATURES SMALL FOOTPRINT



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.

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



*Netpeer™ is delivered as a library.

U S SOFTWARE®
EMBEDDED EXCELLENCE

7175 NW Evergreen Pkwy. Suite 100 Hillsboro, OR 97124

TEL: 503-844-6614 • FAX: 503-844-6480 • EMAIL: info@ussw.com • www.ussw.com

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.

■ 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

800-356-7097

WWW.USSW.COM

LISTING 5 A function to add an entry to the ARP cache

```

int
NetArpAddEntry(INT8U * hw_addr, INT32U ip_addr)
{
    int i;

    /*
     * Look for a place to insert the entry into the ARP cache.
     */
    for (i = 0; i < NET_ARP_CACHE_SIZE; i++)
    {
        if ((gArpCache[i].ip_addr == ip_addr) ||
            (gArpCache[i].ip_addr == 0))
        {
            /*
             * Found existing or new slot in cache.
             */
            gArpCache[i].ip_addr = ip_addr;
            memcpy(gArpCache[i].hw_addr, hw_addr, HW_ADDR_LEN);

            return (i);
        }
    }

    /*
     * The ARP cache is full! Clear the cache and use first slot.
     */
    NetArpInit(n);
    gArpCache[0].ip_addr = ip_addr;
    memcpy(gArpCache[0].hw_addr, hw_addr, HW_ADDR_LEN);

    return (0);
} /* NetArpAddEntry() */

```

Still searching...

I still haven't made up my mind about the embedded platform I'm ultimately going to test this UDP/IP stack on. The problem is that I want to balance two different sets of needs—mine and yours. As the developer of the stack, I need a toolset that is easy to work in and debug with. Likewise, I'd prefer a hardware platform that is easy to understand and explain. As a user of the stack, however, you may only want to evaluate its performance on the reference platform, then port it to something of your own design. So

I'm trying to find an inexpensive (under \$200) single-board computer that is widely available and will continue to be. It'd also be nice if all the tools you'd need to make minor configuration changes and recompile the stack were available for free or as demo versions.

While I consider my hardware and tool options, I'm developing the UDP/IP stack on a PC. Only the network driver will be hardware specific, so it's relatively easy to do the bulk of the development as though it were a PC application. I've got the whole stack up and running on my

development PC already, with separate client and server threads passing messages back and forth over UDP/PI. The stack itself is reentrant, so both threads can share those modules. Underneath the stack, I've implemented a dummy network device driver that assigns each thread unique Ethernet and IP addresses and its own private ARP cache. It then routes packets from one thread to the other as though there were an actual network between them.

By this time next month, I should have some client and server apps, the UDP/IP stack, and my dummy network driver running on μ C/OS-II. Since the book describing that RTOS includes a DOS port on floppy disk, it should be easy enough to accomplish the port of my existing Win32 stack—or so I hope. At that point, I'll probably have a complete set of source code worth sharing with you. Then you can quite literally play along at home if you choose, for the small price of a PC and about \$120 (\$70 for the μ C/OS-II book and \$50 for the Borland C compiler). Who knows, maybe that's the only platform you'll ever want for playing with this stuff. But I'll still ultimately need to test my code on a network-connected embedded system.

Next month I'll tell you everything you ever wanted to know about the IP layer and share my implementation of it with you. In the meantime, enjoy this month's feature articles on Bluetooth and flash memory, and try to stay connected... **esp**

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

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



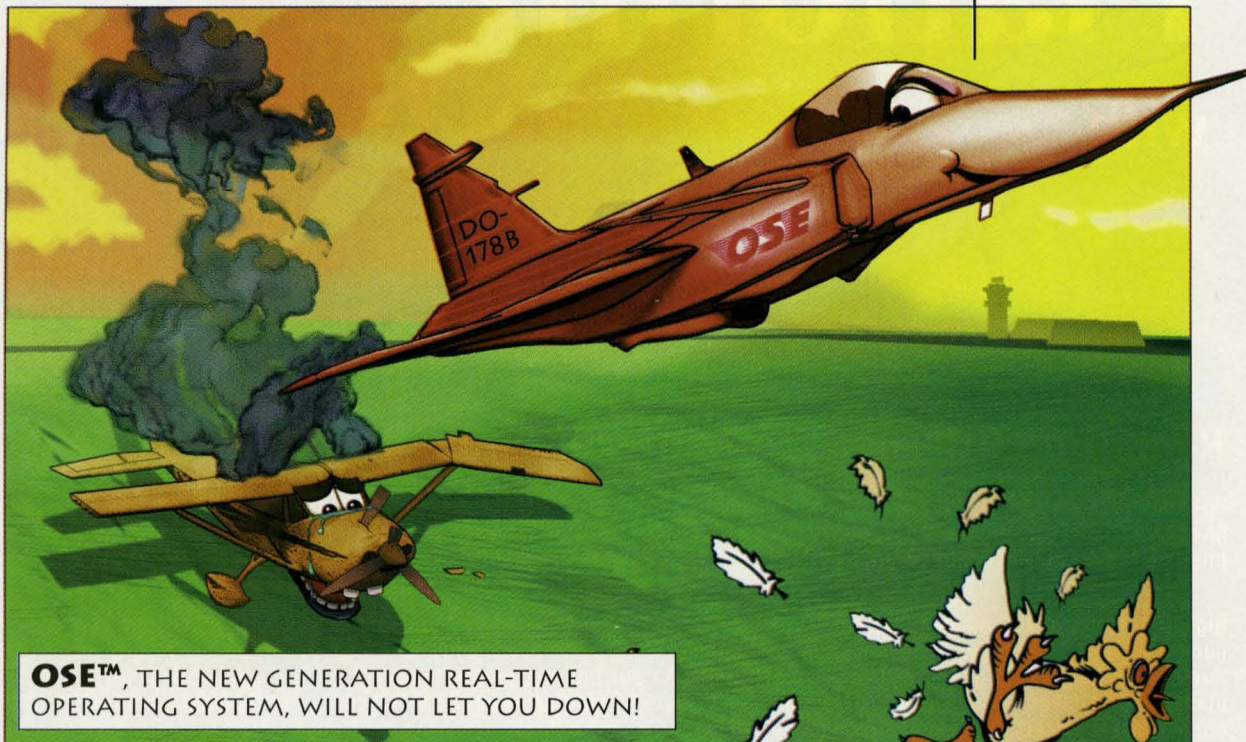


COME SEE US AT THE EMBEDDED SYSTEMS CONFERENCE SUMMER.
Boston - Hynes Convention Center. July 11-12th.

COME VISIT US!

OSE

— Conventional RTOS? ...sorry!



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

ENEAS OSE SYSTEMS

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

Bluetooth Basics

Since the release of the Bluetooth v. 1.0 specification last year, the race has been on to provide Bluetooth-enabled products. Here is an overview of Bluetooth technology, its unique capabilities, and its advantages over competing wireless technologies. It's followed by an in-depth discussion of the Bluetooth link layer and Service Discovery Protocol (SDP).

Bluetooth is an open standard for wireless connectivity with industry backers mostly from the PC and cell phone industries. Not surprisingly, its primary market is for data and voice transfer between communication devices and PCs. In this way, it's similar in purpose to the IrDA protocol. Bluetooth, however, is a radio-frequency (RF) technology utilizing the unlicensed 2.5GHz Industrial-Scientific-Medical (ISM) band. Target applications include PC and peripheral networking, hidden computing, and data synchronization such as for address books and calendars. Other applications could include home networking and home appliances of the future such as smart appliances, heating systems, and entertainment devices.

Why Bluetooth?

Bluetooth attempts to provide significant advantages over other data transfer technologies, such as IrDA and HomeRF, which are vying for similar markets. Despite comments from the Bluetooth Special Interest Group (SIG) indicating that the technology is complementary to IrDA, it is clearly a competitor for PC-to-peripheral connection. IrDA is already popular in PC peripherals, but is severely limited by the short connection distance of 1m and by the

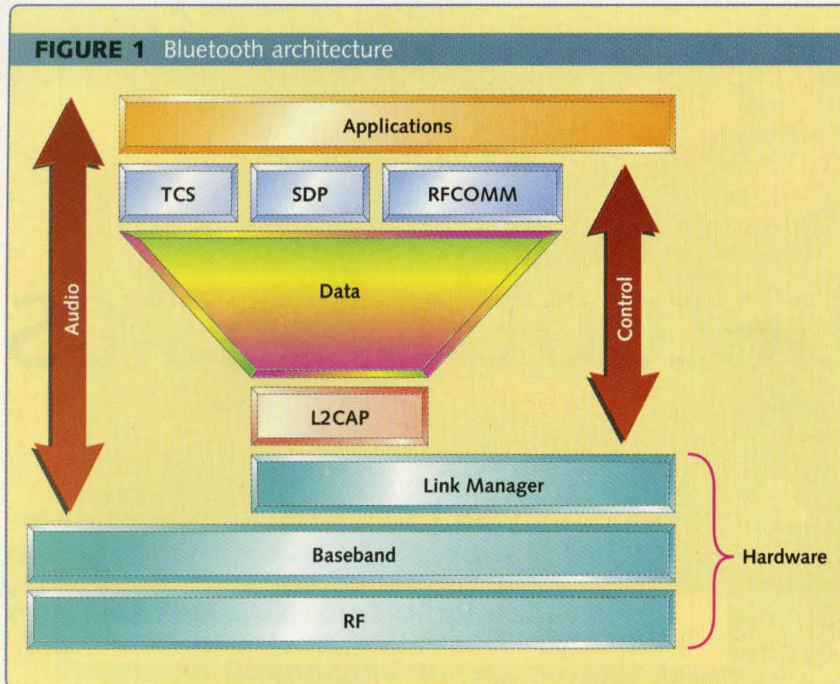
line-of-sight requirement for communication. This limitation eliminates the feasibility of using IrDA for hidden computing, where the communicating devices are nearby but not visible to one another.

Due to its RF nature, Bluetooth is not subject to such limitations. In addition to wireless device connections up to 10m (up to 100m if the transmitter's power is increased), devices need not be within line-of-sight and may even connect through walls or other non-metal objects. This allows for applications such as a cell phone in a pocket or a briefcase acting as a modem for a laptop or a PDA.

Bluetooth could also be used in home networking applications. With increasing numbers of homes having multiple PCs, the need for networks that are simple to install and maintain is growing. Wireless connections circumvent the hassle of adding wiring to existing residences. Competing technologies for this market include HomeRF and IEEE 802.11. The main drawback to these technologies is their cost, which is in both cases greater than \$100 per node.

Bluetooth is designed to be low cost—eventually under \$10 per unit. On the flip side, however, is the limited connection distance, and even more damaging, the transmission speeds. Bluetooth supports only 780Kbps, which may be used for 721Kbps unidirectional data transfer (57.6Kbps

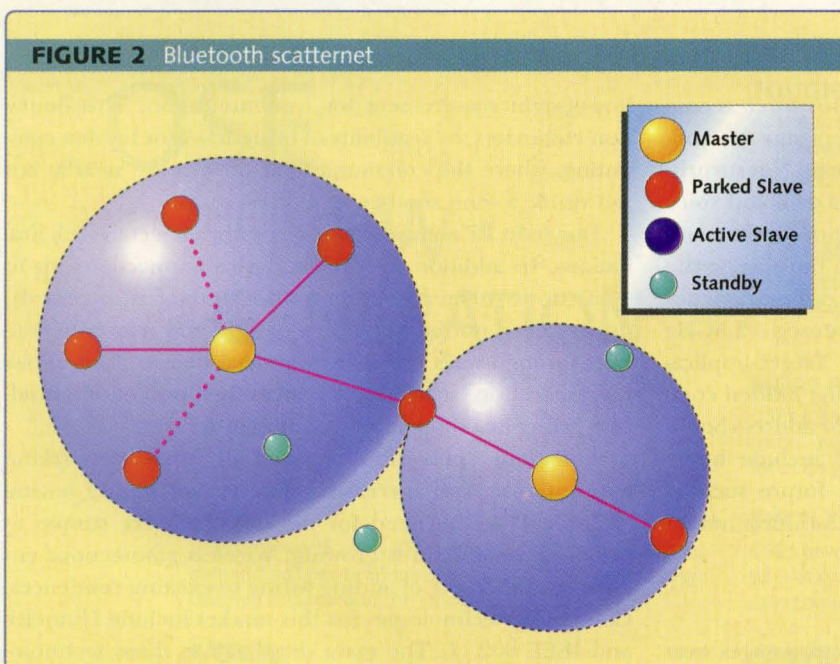
The Bluetooth Core specification has over 1,000 pages and covers everything from the radio transceiver to various interfaces to third-party protocols.



capable of supporting one asynchronous data channel and up to three synchronous voice channels, or one channel supporting both voice and data. This capability combined with ad hoc device connection and automatic service discovery make it a superior solution for mobile devices and Internet applications. This combination allows such innovative solutions as a mobile hands-free headset for voice calls, print to fax capability, and automatically synchronizing PDA, laptop, and cell phone address book applications.

Architecture overview

The Bluetooth Core specification has over 1,000 pages and covers everything from the radio transceiver to various interfaces to third-party protocols. This includes descriptions of technology implemented in silicon, in hardware, and in software. A diagram of the Bluetooth protocol architecture is shown in Figure 1.



Link control hardware. Bluetooth link control hardware, either integrated as one chip or as a radio module and a baseband module, implements the RF, Baseband, and Link Manager portions of the Bluetooth specification. This hardware handles radio transmission and reception as well as required digital signal processing for the baseband protocol. Its functions include establishing connections, support for asynchronous (data) and synchronous (voice) links, error correction, and authentication. The link manager firmware provided with the baseband CPU performs low-level device discovery, link setup, authentication, and link configuration. Link managers on separate devices communicate using the Link Management Protocol, which utilizes the services of the underlying link controller (baseband). The link control hardware may also provide a Host Controller Interface (HCI) as a standard interface to the software.

Network topology. Bluetooth devices are generally organized into groups of two

return direction) or for up to 432.6Kbps symmetric data transfer. These rates are comparable to the 1Mbps to 2Mbps supported by HomeRF and, although live digital video is still beyond the capability of

any RF technology, are perfectly adequate for file transfer and printing applications.

Finally, Bluetooth's main strength is its ability to simultaneously handle both data and voice transmissions. It's

why would you want to network a thermostat?



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

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.



e-smart™



device networking software that makes products better
go to www.emware.com/thermostat3 or call toll-free 1.877.436.9273

Bluetooth operates in the unlicensed ISM frequency band that is generally cluttered with signals from other devices.

A diagram of a Bluetooth scatternet is shown in Figure 2.

Bluetooth operates in the unlicensed ISM frequency band that is generally cluttered with signals from other devices—garage door openers, baby monitors, and microwave ovens, to name just a few. To help Bluetooth devices coexist and operate reliably alongside other ISM devices, each Bluetooth piconet is synchronized to a specific frequency hopping pattern. This pattern, moving through 1,600 different frequencies per second, is unique to the particular piconet. Each frequency “hop” is a time slot during which data packets are transferred. A packet may actually span up to five time slots, in which case the frequency remains constant for the duration of that transfer.

Baseband state machine. Piconets may be static or may be formed dynamically as devices move in and out of range of one another. A device leaves standby (the low power, default state) by initiating or receiving an inquiry or a page command. An inquiry may be used if the address of a targeted device is unknown; it must be followed by a page command. A page command containing a specific DeviceAccessCode is used to connect to a remote device. Once the remote device responds, both devices enter the connected state, with the initiating device becoming the master and the responding device acting as a slave.

Once in the connected state, the slave device will synchronize to the master’s clock and to the correct frequency hopping pattern. At this point, link managers exchange commands in order to set up the link and acquire device information. The master will then initiate regular transmissions in order to keep the piconet synchronized. Slaves listen on every master-transmit time slot in order to synchronize with the master and to determine if they have been addressed.

FIGURE 3 Host controller interface

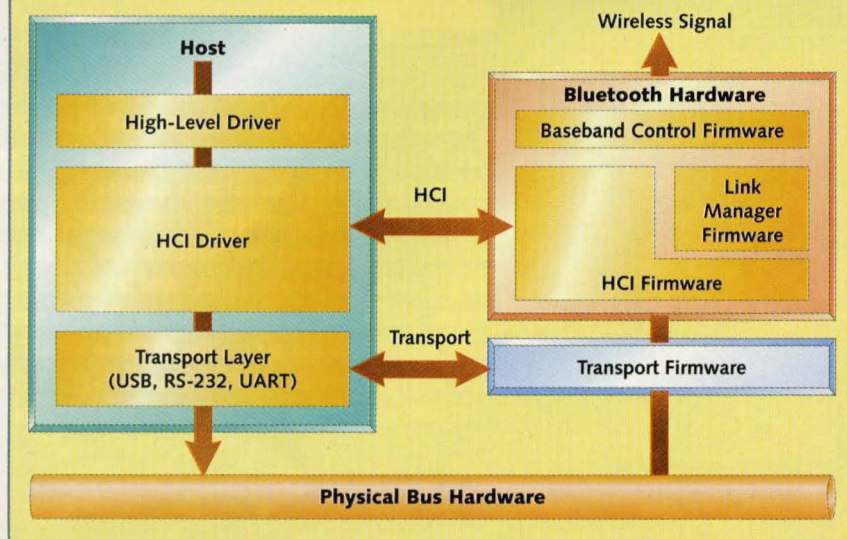
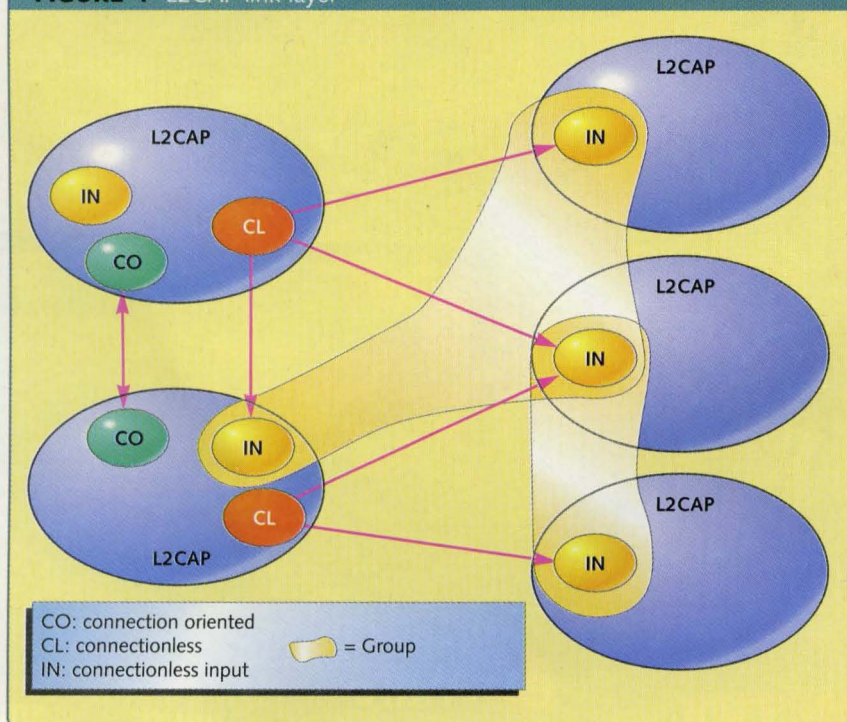


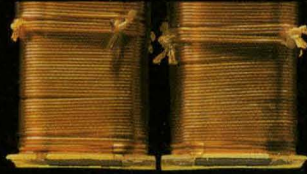
FIGURE 4 L2CAP link layer



to eight devices called *piconets*, consisting of a single master device and one or more slave devices. A device may additionally belong to more than one

piconet, either as a slave in both or as a master of one piconet and a slave in another. These bridge devices effectively connect piconets into a *scatternet*.

Can your Java™ make this ball float?



Small AND
fast Jbed can.



Visit us at
JavaOne



And every swing is a home-run!

Jbed

The embedded
real-time Java™
Virtual Machine

Jbed runs faster than other Javas because it always compiles, never interprets. Its small memory footprint allows low-cost hardware with less memory. You save time and cost because Jbed offers the productivity of the Java™ language to the embedded world. **Included features are:**

- Integrated development environment (IDE)
- Deadline-driven/time-triggered scheduling with admission testing
- Full support of the Java™ language as specified by Sun Microsystems
- Incremental Garbage Collector compliant with hard real-time
- Reflection, serialization
- Dynamic loading of Java™ byte code
- All standard communication protocols supported
- RTOS and JVM (Java™ Virtual Machine) in one!

esmertec inc., Technoparkstrasse 1, CH-8005 Zürich, Switzerland
www.esmertec.com, info@esmertec.com, Tel +41 1 445 37 30, Fax +41 1 445 37 34

esmertec

your partner for embedded real-time Java™

Synchronous connection-oriented (SCO) links are typically used for voice transmission.

Each active slave is assigned an active member address (AM_ADDR) and participates actively on the piconet, listening to all master time slots to determine if it is being addressed by the master. In addition, there are three lower-power slave states: sniff, hold, and park. A master can only transmit to devices in sniff mode during particular sniff-designated time slots. Therefore, these devices listen only during these special time slots and sleep the rest of the time. A slave in hold mode, alternately, does not receive any asynchronous packets and listens only to determine if it should become active again. Finally, a device in park mode not only stops listening, but gives up its active member address. It is only a

member of the piconet in that it remains synchronized with the frequency hopping pattern.

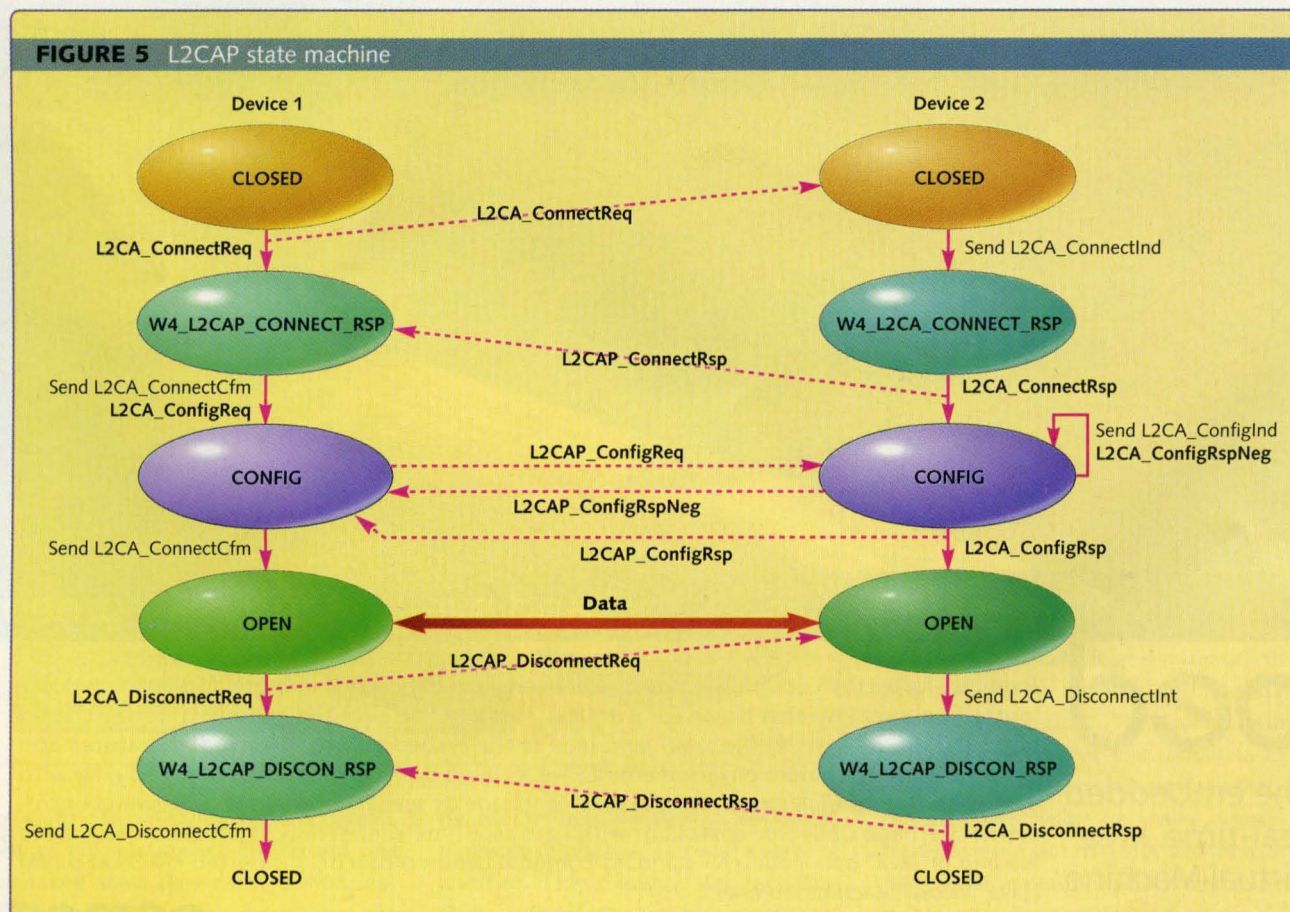
Baseband links. The Bluetooth baseband provides transmission channels for both data and voice and is capable of supporting one asynchronous data link and up to three synchronous voice links (or one link supporting both). Synchronous connection-oriented (SCO) links are typically used for voice transmission. These are point-to-point symmetric connections that reserve time slots in order to guarantee timely transmission. The slave device is always allowed to respond during the time-slot immediately following a SCO transmission from the master. A master can support

up to three SCO links to a single or multiple slaves, but a single slave can support only two SCO links to different masters. SCO packets are never retransmitted.

Asynchronous connectionless (ACL) links are typically used for data transmission. Transmissions on these links are established on a per-slot basis (in slots not reserved for SCO links). ACL links support point-to-multipoint transfers of either asynchronous or isochronous data. After an ACL transmission from the master, only the addressed slave device may respond during the next time-slot, or if no device is addressed, the packet is considered a broadcast message. Most ACL links include packet retransmission.

Link manager. The baseband state machine is controlled largely by the link manager. This firmware—general-

FIGURE 5 L2CAP state machine



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®, RTX®, 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

I-BROWZR, 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.

I-BROWZR 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.

I-BROWZR is compatible with EBSnet's RTIP but can be standalone.

ERTFS-DOS COMPATIBLE FILE SYSTEM

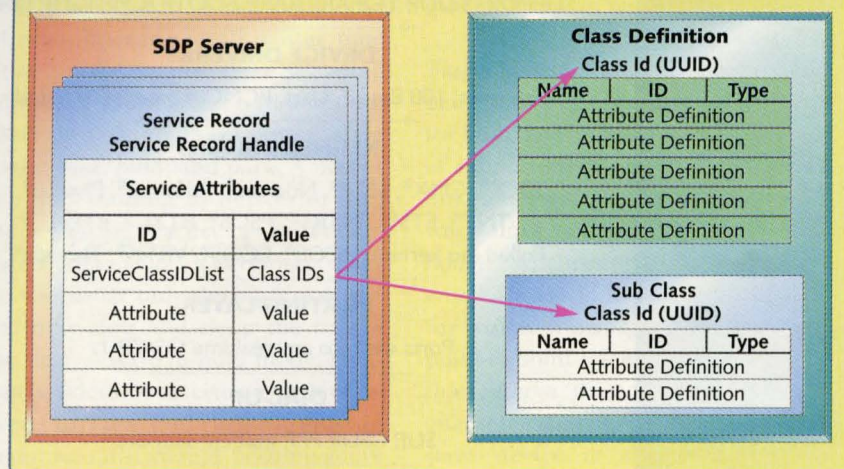
Comprehensive, portable, high performance
DOS compatible file I/O for embedded systems.

ebsnet
INC



Link managers communicate with each other using the Link Management Protocol (LMP), which uses the underlying baseband services.

FIGURE 6 Service attribute definition



ly provided with the link control hardware—handles link setup, security, and control. Its capabilities include authentication and security services, quality of service monitoring, and baseband state control. The link manager controls paging, changing slave modes, and handling required changes in master/slave roles. It also supervises the link and controls handling of multi-slot packets.

Link managers communicate with each other using the Link Management Protocol (LMP), which uses the underlying baseband services. LMP packets, which are sent in the ACL payload, are differentiated from logical link control and adaptation protocol (L2CAP) packets by a bit in the ACL header. They are always sent as single-slot packets and are a higher priority than L2CAP packets. This helps ensure the integrity of the link under a high traffic demand.

Host controller interface (HCI). Some link controller hardware may include an HCI layer above the link manager. This firmware layer is used to isolate the Bluetooth baseband and link manager from a transport protocol such as

USB or RS-232. This allows a standard host processor interface to Bluetooth hardware. An HCI driver on the host is used to interface a Bluetooth application with the transport protocol. Currently three transport mechanisms are supported: USB, RS-232, and UART. The HCI layer is illustrated in Figure 3. Using HCI, a Bluetooth application can access Bluetooth hardware without knowledge of the transport layer or other hardware implementation details.

Software protocols. The remaining Bluetooth protocols are implemented in software. L2CAP, the lowest layer, provides the interface to the link controller and allows for interoperability between Bluetooth devices. It provides protocol multiplexing, which allows support for many third party upper-level protocols such as TCP/IP and vCard/vCalendar. In addition, L2CAP provides group management—mapping upper protocol groups to Bluetooth piconets, segmentation and re-assembly of packets between layers, and negotiation and monitoring “quality of service” between devices.

Several Bluetooth protocols inter-

face to the L2CAP link layer. SDP provides service discovery specific to the Bluetooth environment without inhibiting the use of other service discovery protocols. RFCOMM is a simple transport protocol providing serial data transfer. A Port Emulation Entity is used to map the communication API to RFCOMM services, effectively allowing legacy software to operate on a Bluetooth device. Telephony Control Protocol Specification (TCS) is provided for voice and data call control, providing group management capabilities and Connectionless TCS, which allows for signaling unrelated to an ongoing call. Both point-to-point and point-to-multipoint signaling are supported using L2CAP channels, although actual voice or data is transferred directly to and from the baseband—bypassing L2CAP—over SCO links.

Bluetooth also supports IrDA Object Exchange Protocol (IrOBEX), a session protocol defined by IrDA. This protocol may run over other transport layers, including RFCOMM and TCP/IP. For Bluetooth devices, only connection-oriented OBEX is supported. Three application profiles have been developed using OBEX. These include synchronization functionality for phone books, calendars, messaging, and so on; File Transfer functionality; and Object Push for business card support.

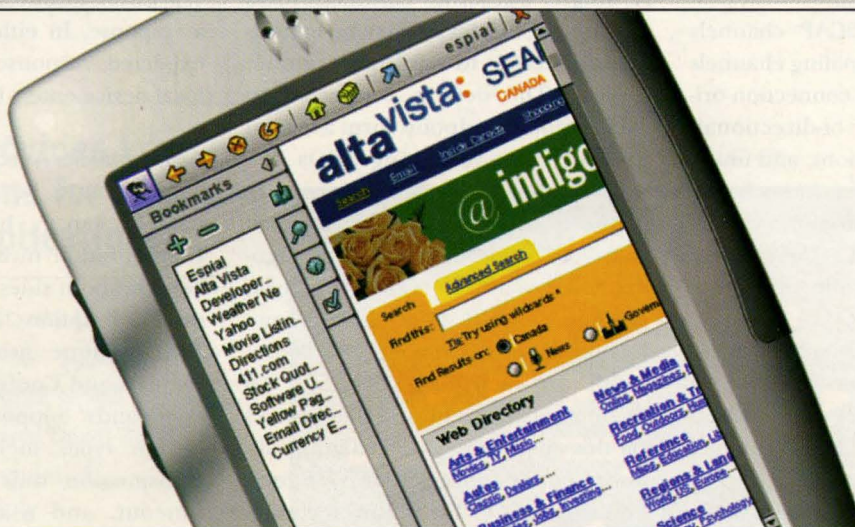
Finally, Bluetooth may be used as a Wireless Application Protocol (WAP) bearer. The specification outlines the interoperability requirements for implementing this capability.

Logical Link Control and Adaptation Protocol (L2CAP)

The L2CAP link layer operates over an ACL link provided by the baseband. A single ACL link, set up by the Link Manager using LMP, is always available between the master and any active slave. This provides a point-to-multipoint link supporting both asynchronous and isochronous data transfer.

National Delivers

Industry's First 2.0



Vision available. Hardware not included.



BUILD A SMART FUTURE™

espial.com

An L2CAP connection-oriented channel endpoint may be in one of several possible states, with data transfer only possible in the OPEN state.

L2CAP provides services to upper-level protocols by transmitting data packets over L2CAP channels.

Three types of L2CAP channels exist: bi-directional signaling channels that carry commands; connection-oriented channels for bi-directional, point-to-point connections; and unidirectional connectionless channels that support point-to-multipoint connections, allowing a local L2CAP entity to be connected to a group of remote devices.

Channels. Figure 4 shows L2CAP entities with various types of channels between them. Every L2CAP channel includes two endpoints referred to by a logical channel identifier (CID). Each CID may represent a channel endpoint for a connection-oriented channel, a connectionless channel, or a signaling channel. Since a bi-directional signaling channel is required between any two L2CAP entities before communication can take place, every L2CAP entity will have one signaling channel endpoint with a reserved CID of 0x0001. All signal channels between the local L2CAP entity and any remote entities use this one endpoint.

Each connection-oriented channel in an L2CAP entity will have a local CID that is dynamically allocated. All connection-oriented CIDs must be connected to a single channel, and that channel must be configured before data transfer can take place. Note that the channel will at that point be bound to a specific upper-level protocol. In addition, a quality of service (QoS) agreement for the channel will be established between the two devices. QoS is negotiated for each channel during configuration and includes data flow parameters such as peak bandwidth, as well as the transmission type: best effort, guaranteed, or no traffic.

Connectionless channels are unidirectional and used to form groups. A single outgoing connectionless CID on a local device may be logically connected to multiple remote devices. The devices connected to this outgoing endpoint form a logical group. These outgoing CIDs are dynamically allocated. The incoming connectionless CID, however, is fixed at 0x0002. Although multiple outgoing CIDs may be created to form multiple logical groups, only one incoming connectionless CID is provided on each L2CAP entity. All incoming connectionless data arrives via this endpoint. These channels do not require connection or configuration. Therefore, any required configuration information, such as upper-level protocol, is passed as part of the data packet.

Channel state machine. An L2CAP connection-oriented channel endpoint may be in one of several possible states, with data transfer only possible in the OPEN state. Initially, an endpoint is CLOSED, indicating that no channel is associated with the CID. This is the only state in which a baseband is not required and it is the state an endpoint will default to if the link is disconnected. Figure 5 illustrates the state interactions between two L2CAP entities on either end of a connection-oriented channel.

Connection. In order to open a channel, the channel endpoint must be connected and configured. A connection occurs when either the local L2CAP entity requests connection to a remote device or an indication has been received indicating that a remote L2CAP entity is requesting connection to a local CID. In the first case, the request has originated from the upper-level-protocol, has been passed on to the remote device, and the local entity

enters the W4_L2CAP_Connect_RSP state to await a response. In the latter case, the indication is recognized as a connection request, the request has been passed on to the upper layer, and the local entity enters the W4_L2CA_Connect_RSP state to await a response. In either case, when the expected response is received, the local device enters the CONFIG state.

Configuration. A connection-oriented channel must be configured before data can be transmitted. Configuration involves a negotiation between both sides of the connection until all options are agreed upon. This is done using Configuration Request and Configuration Response commands. Supported configuration option types include a maximum transmission unit (MTU), a flush timeout, and a quality of service (QoS) agreement. The MTU option reflects the largest L2CAP packet payload the local device can handle. The flush timeout determines the amount of time the link controller will attempt to transmit a L2CAP segment before flushing the packet. Finally, the QoS is used to negotiate a flow specification for a single transmission direction. L2CAP implementations are only required to support "Best Effort" service but "No traffic" or "Guaranteed" service may also be negotiated. Other parameters in the flow specification include the token rate, token bucket size, peak bandwidth, latency, and delay variation. The requesting device indicates all of the non-default options it will accept, to which the responding device either agrees or provides alternate settings. This process continues until all options are agreed upon. This configuration is for a single transfer direction, however, and the process must then be repeated for the opposite transfer direction. After all configuration parameters have been determined, both L2CAP entities enter the OPEN state, at which point data transfer may begin.

National Delivers Industry's First 2.4GHz Integrated ISM Transceiver

LMX3162 Provides Low Cost Solutions for Emerging Wireless Applications

- System RF Sensitivity to -93dBm; RSSI Sensitivity to -100dBm
- System Noise Figure 6.5dB (typ.)
- 70μA (max.) Power Down Mode
- Works with Unregulated 3.0V - 5.5V Supply
- Two Regulated Voltage Outputs for Discrete Amplifiers
- High Gain (85dB) Intermediate Frequency Strip

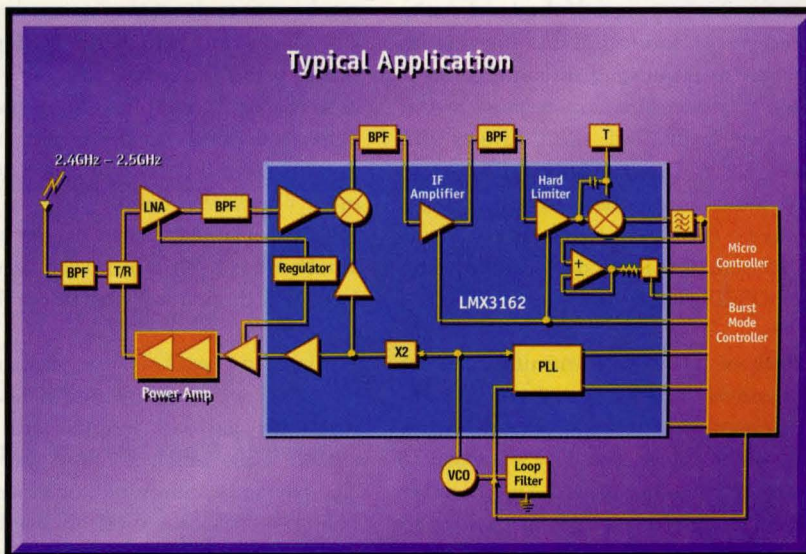
Ideal for 2.4GHz Voice/Cordless Applications, Wireless Networking Proprietary Protocols, and Products Utilizing Bluetooth, HomeRF, 802.11b, or 2.4GHz US DECT


For More Information on LMX3162:

www.national.com
1-800-272-9959

Evaluation Board Available

Free CD-ROM Data Catalog
Available at freecd.national.com



©1999 National Semiconductor Corporation. National Semiconductor and  are registered trademarks of National Semiconductor Corporation. All other trademarks are the property of their respective owners. All rights reserved.



Bluetooth.



National
Semiconductor®

Service discovery protocol provides a means of determining what Bluetooth services are available on a particular device.

Disconnection. To close a channel, one L2CAP entity must send a disconnection request to the other. If an entity receives a disconnect request from the upper-level protocol, it passes the request onto the remote device and the local entity enters the `W4_L2CAP_DISCONNECT_RSP` state to await a response. If the local entity receives an indication that the remote device is requesting disconnection, it sends a disconnection request to the upper layer and then enters the `W4_L2CA_DISCONNECT_RSP` state to await a response. In either case, when the expected response is received, the local device enters the `CLOSED` state.

Packets. Data is transmitted across channels using packets. A connection-oriented channel uses packets with a 32-bit header followed by a payload of up to 65,535 bytes. The header includes a 16-bit length of payload to use for integrity check and the 16-bit destination CID. The payload contains information received from or being sent to the upper-level protocol. Connectionless channel packets also include a header but always use 0x0002 for the remote CID. In addition, the header is followed by a 16-bit (minimum) protocol/service multiplexer (PSM), which is used to indicate the upper-level protocol the packet originated from. This allows for packet re-assembly on the remote device. The PSM field is not required for connection-oriented channels since they are bound to a specific protocol during connection.

L2CAP supports segmentation and reassembly of packets between upper-level and lower-level protocols. Packets from the upper-level must not exceed a maximum transmission unit (MTU) negotiated during configuration. These upper-level packets are segmented into protocol data units

(PDUs) that are small enough for the lower-level protocol. Depending on the protocol stack implemented, PDUs may be baseband packets or blocks of data supported by a host controller interface. In either case, all PDUs for a packet must be sent over the baseband before another packet can be sent to the same remote device. The baseband will send PDUs in order, using retransmit and timeouts as necessary, but notification of packet delivery depends on the implementation. If reliability is required, L2CAP must check the length field of the L2CAP packet header to determine if a packet has been transmitted successfully and then notify the upper level protocol.

Signaling packets are similar to data packets. They include a header with the payload length and the remote CID, which is always 0x0001. The payload contains one or more signaling commands used to establish and remove connections or to initiate activity over a link. Each command includes a command code, an identifier used to match requests with corresponding responses, a data length, and zero or more bytes of data. The commands are either requests or responses with every request requiring a corresponding response with the same identifier. Signaling commands provide connection-oriented service primitives and include requests and responses related to connecting, configuring, and disconnecting channels, as well as echo (ping) and information (MTU) requests and responses. In addition, a Command Reject command is used to respond to an invalid or unrecognized command.

Service discovery protocol (SDP)

SDP provides a means of determining what Bluetooth services are available on a particular device. A Bluetooth

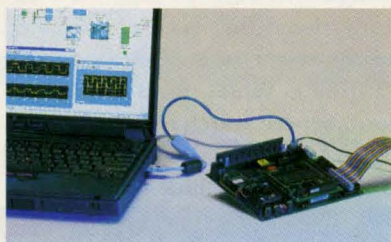
device may act as an SDP client querying services, an SDP server providing services, or both. A single Bluetooth device will have no more than one SDP server, but may act as a client to more than one remote device. SDP provides access only to information about services; utilization of those services must be provided via another Bluetooth or third-party protocol. In addition, SDP provides no notification mechanism to indicate that an SDP server, or any specific service, has become available or unavailable as may occur when the services available on a device change, or when a device comes in or out of RF proximity. This would be a common occurrence in a network supporting mobile devices. The client may, of course, poll a server to detect unavailability, but other means are required for detecting a server or service that has recently become available.

Service records. In SDP, a service may provide information, perform an action, or control a resource. SDP servers maintain service records to catalog all available services provided by the device. Each service is represented by a single service record with a dynamically allocated service record handle that is unique within the server. A special service record, with a service record handle of 0x00000000, is provided to describe the SDP server itself and its supported protocols. Service attributes within a service record describe and define the supported service including the service type, a service ID, the protocols supported, the service name, a service description, and so on. These attributes are composed of a 16-bit ID and a variable length value. Attribute values in turn include a header field, with a data type and data size, and a data field. A range of data types are supported: null, unsigned integer, signed twos-complement integer, Universally Unique Identifier (UUID), text string, Boolean, data element sequence (set), data element alternative (select one),

NEW
PC-IN-THE-LOOP
PROTOTYPING
PRODUCTS

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

With the new PC-in-the-Loop™ products, you can accurately test the



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

performance of real-time embedded system designs on a low-cost PC long before your final target hardware is available. Simulink code generation products convert block diagrams to real-time C code for rapid prototyping. You can

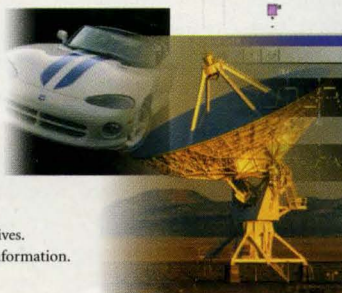
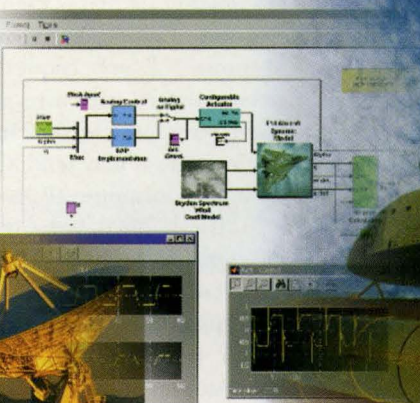
then go directly to your embedded target with our production-quality code.

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

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

SIMULINK®

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



The
**MATH
WORKS**
Inc.

Visit www.mathworks.com/esp
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.

© 2000 The MathWorks, Inc.

With the bulk of the work developing the Bluetooth specification complete, the Bluetooth SIG is now working on improvements and analyzing feedback from the industry.

and URL. The interpretation of this data is dependent on the attribute ID and the service class to which the service belongs.

Service classes. Each service belongs to a service class which defines all attributes by ID, intended use, and the format of the attribute value. A sub-class of another service class may define more specific attributes, in which case the sub-class inherits the attributes of the super-class. The `ServiceClassIDList` attribute of a service record lists the class IDs for all classes to which the service belongs. A class ID is a UUID that identifies a specific service class. If a service belongs to more than one class, the `ServiceClassIDList` will list class IDs starting from the most specific subclass and ending with the most general super-class, effectively defining all service attributes. The relationship between the service record attributes and the service classes is illustrated in Figure 6.

Discovering services. The purpose of SDP is to discover, not access, services. Two processes are supported: searching and browsing. Searching is based on UUIDs. A service record is returned by a search only if all of the UUIDs in the service search pattern are found within service record attribute values. It does not matter in which attribute a UUID is found, or whether the UUID is only one element in a list, as long as all the search pattern UUIDs are contained somewhere among the attribute values for the service.

Searching is useful if the client is looking for specific capabilities, but for a client to identify all services offered by a remote device, the SDP "browsing" mechanism must be used. Browsing is accomplished via the search capability by using a special service attribute supported by all service

classes: `BrowseGroupList`. This attribute contains a UUID list of "browse groups." At the top-level of the browse hierarchy, this will be the "PublicBrowseRoot." Searching on the PublicBrowseRoot UUID will return all services that may be browsed at the top-level, as well as `BrowseGroupDescriptor` service records describing lower-level browse groups. `BrowseGroupDescriptor` service records include a `GroupID` attribute with the UUID for the browse group it is describing. To browse the services belonging to this group, the client must read the `GroupID` attribute of the `BrowseGroupDescriptor` and then search on this group ID. Services belonging to this group will include the group ID in their `BrowseGroupList` attribute, and will therefore be returned by the search.

Protocol. SDP is a packet-based protocol utilizing a request-response architecture. The SDP packet is referred to as a protocol data unit (PDU), which includes a header followed by a variable number of parameters. The length of the parameter field is specified in the header, as is the type (PDU ID) which may indicate a request or response for searches or attribute queries. The header also includes a transaction ID that is used to match a request with a corresponding response. Every request must be acknowledged with a response. If for some reason the server cannot handle the request, it may send a response of type `ErrorResponse` (PDU ID 0x01).

It is possible that the response may be too large to fit into a single PDU. To accommodate this, a continuation state parameter is supported by most PDUs. In a response, this parameter indicates the number of bytes that are outstanding. The client may then resend the original request—with a new

transaction ID—but with the continuation state parameter set. This alerts the server to send the continuation of the response. At what point a response is split is determined by the server.

Three categories of transactions (PDU IDs) are supported: service search transactions, service attribute transactions, and service search attribute transactions. Service search transactions are used to request a list of service record handles for service records which have attributes containing all of the UUIDs in a service search pattern. There is no mechanism to request all the service records, although browsing is supported as already described. Service attribute transactions are used to request specific attribute values from a service record. Service search attribute transactions combine the service search and the service attribute transactions, which allows getting specific attribute values for all service records that match a service search pattern.

Bluetooth today and tomorrow

With the bulk of the work developing the Bluetooth specification complete, the Bluetooth SIG is now working on improvements and analyzing feedback from the industry. In addition to their work investigating improvements in speed, security, noise immunity, and so on, the SIG continues to develop Bluetooth profiles. These are defined in a separate Bluetooth Profiles Specification, also in excess of 1,000 pages.

Bluetooth profiles. The continuing effort to define profiles provides Bluetooth with a significant advantage over competing technologies. Profiles, defined by the Bluetooth SIG, are intended to ensure interoperability between Bluetooth applications and devices from different manufacturers. These profiles define the roles and capabilities for specific types of applications. Different profiles may encompass different layers and protocols and



Stone Age

For every age there is a tool

BSQUARE's off the shelf development tools get you to market faster.

Whether you're developing for Windows® 2000, NT®/NT Embedded, CE or 98, with BSQUARE's BlueWater Systems development toolkits, it'll take you no time at all. You'll save valuable time and development costs, as well as quickly create high-performance device drivers that work – the first time.

We've helped many companies get great products to market faster, learn more by calling us at 888-820-4500 or visiting www.bsquare.com.



Device
Driver
Toolkits

Digital Age

© 2000 BSQUARE Corporation. All rights reserved. BSQUARE is a registered trademark of BSQUARE Corporation. All other names, product names, and tradenames are trademarks or registered trademarks of their respective holders.

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™



www.bsquare.com

Along with the continuing development of profiles, the Bluetooth SIG is working on other fronts to improve the specification.

to different degrees. In addition to requirements for interoperability, protocols may define required services to other applications or to end users.

All Bluetooth devices must support the Generic Access Profile at a minimum. This profile defines device discovery, connection procedures, and procedures for various security levels. Some Bluetooth-specific user interface requirements are described as well. Another universal profile, although not required, is the Service Discovery Access Profile which defines how a service discovery application on a device determines the services on other remote devices as well as the protocols and associated parameters required to access them. Quite a number of additional profiles have been

defined including TCS-, RFCOMM-, and OBEX-related profiles. Some of these require the implementation of others, and all of them require the implementation of the Generic Access Profile. So much work has been done developing profiles that a separate article would be required to adequately cover them all.

What's next

Along with the continuing development of profiles, the Bluetooth SIG is working on other fronts to improve the specification. This includes addressing other interoperability issues and developing support for faster data rates, a crucial factor in the future success of the technology. Bluetooth security capability is also an

area of concern for potential Bluetooth adopters. On the implementation front, work is being done to integrate the silicon for the RF and baseband chips, as well as achieving a small enough implementation to be practical in small portable devices.

At the time of this writing, Bluetooth RF/baseband modules were selling well above the \$5 per unit target. After an impressive marketing push that resulted in Bluetooth nearly becoming a household name, designers are beginning to give Bluetooth a more critical look and are voicing their concerns. Although it is well positioned for success in the emerging wireless networking market, the key will be in Bluetooth's evolution.

The Bluetooth SIG must overcome the outstanding barriers to wide-scale implementation and must keep the technology up to date with the ever-increasing consumer demands for price and performance. If they can accomplish that, the Bluetooth wireless world they envision may well come to be.

esp

After earning a BS in electrical engineering from the University of Rochester, Rebecca Spaker has spent the last eight years designing embedded hardware and software, with recent work focused on hardcopy and digital imaging-related applications. For the last four years she has worked for Qestra Corp. as a senior consultant. Currently, she works with Qestra's eAppliances practice group, providing consulting and development services for internet-enabled devices. Her e-mail address is rspaker@qestra.com.

References

- Specification of the Bluetooth System—Core v1.0A. July 26, 1999.
- Specification of the Bluetooth System—Profiles v1.0A. July 26, 1999.
- www.bluetooth.com. The official Bluetooth Web site.
- www.bluetooth.net. Digianswer A/S Bluetooth Web site.
- www.palopt.com.au/bluetooth. Palo Pacific Technology Bluetooth Web site.

1 The SBC One Stop Shop 1

PC Compatible SBCs



LCD Touch Screens



PC/104 Add-Ons



Custom Applications



Custom Display Solutions





SBC Microcontrollers
A/D · D/A · PWM



Microcontroller Add-Ons



Microprocessor Training Systems



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

1984-1998
OVER
14
YEARS
OF SERVICE

IF YOU USE:

ATI Nucleus™

Embedded Power RTX™

Express Logic ThreadX™

Kadak AMX™

Precise MQX™

Wind River VxWorks™

Your own custom RTOS

ON:

ARM

StrongARM

ARC

PowerPC

picoJava

X86

Contact MetaWare

for a free evaluation copy

www.metaware.com/eval

OEM

PowerPC

ARM

ARC

StrongARM

picoJava

SOC

X86

64-bit

32-bit RISC

If time to market is important to you...

If performance, quality, and excellent customer service matter to you...

then you need to evaluate the MetaWare tools!



MetaWare®
INCORPORATED

2161 Delaware Avenue, Santa Cruz, CA 95060 USA • Toll-free 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

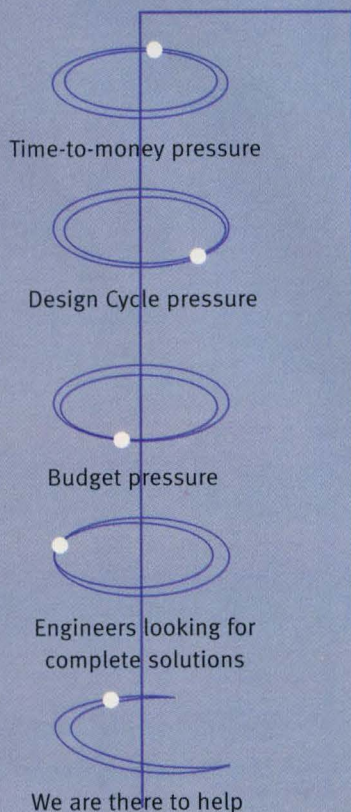
MetaWare is a registered trademark of MetaWare Incorporated. All other company and product names are trademarks or registered trademarks of their respective companies.

We put you in a very COMFORTABLE position with our 8-BIT MCUs.



{PERPETUAL THINKING PROCESS}

INFINEON CYCLES



HAS DESIGNING your 8-bit application and time-to-money pressures got you stressed out? Relax. Now working on 8-bit microcontroller applications couldn't be easier. Why? Infineon's broad C500 family has got all you need to stay cool. Each member is naturally 8051 compatible and features a versatile set of powerful peripherals with enhanced functionality. Plus they're very, very attractively priced. Looking for CAN, Motor Control, USB and OTP solutions? No problem. Need support? You won't after you've checked out our comprehensive set of development and support tools. And DAVE, our digital application virtual engineer, is always there to help. After all, we should know what design engineers want. We're Infineon – the people who never stop thinking of easy ways to get you through the entire design cycle.

www.infineon.com



From PROM to Flash

Moving from PROM to flash isn't easy. Here's a list of required steps and best practices for storing executable code in a flash memory device and making it upgradable in the field.

Designing an embedded system containing flash program memory isn't always as straightforward as it seems. A robust flash-based system needs to deal with such things as in-circuit reprogramming and code execution from RAM—somewhat esoteric topics not associated with PROM. Although several past articles have touted flash memory and its various advantages, many have lacked the details necessary for an engineer to fully understand how to implement flash as a better alternative to PROM. I endeavor to change that by presenting an example from my own experience converting a PROM-based embedded system to flash. I will emphasize software design, as therein lies most of the difficulty.

Although I'll discuss an approach specific to the example system I'll present, the techniques should be applicable to most any embedded product being designed with flash program memory. The scheme I developed is but one possible solution to the flash reprogramming problem. I don't claim that it is the best or the only solution, just that I felt that it was a workable approach when considering the specific constraints of the product I was working on. As the saying goes, your mileage may vary.

It's not my intent to tell you how to write the code to program bytes or erase sectors in a certain vendor's flash chip—that's easy once you have the chip's data sheet. After I present an overview of the characteristics of flash and the design goals for my specific project, I'm going to discuss the tough stuff—mainly an algorithmic approach, but also some tips for working with flash that may not be entirely obvious.

Flash overview

Does anyone doubt the superiority of flash over PROM anymore for program memory storage? Flash offers all of the capabilities of PROM and adds some of its own, all at a reasonable price. The most notable advantage of flash pro-

The downside of flash is that to achieve in-circuit reprogrammability, the software design effort is usually fairly complicated.

Easier flash reprogramming variations

Although the PROM-to-flash conversion example that I will discuss later is one of the more difficult flash variations due to system and cost constraints, some easier alternatives are available. A type of flash called "simultaneous read/write" flash can be used in low-voltage (typically 3.3V and lower) designs. This type of flash allows erasing or programming of a sector during execution from a different sector, and thus is well suited to boot block code designs. On the subject of different voltage levels, some older flash required an external high programming voltage, typically 12V, to be applied to the chips during erasing or programming. More modern flash eliminates this requirement.

Another easier design variation is to have a small boot PROM or a small second flash in addition to the main flash chip that will be reprogrammed. This allows execution from the smaller device while the larger flash is being erased or programmed, although the added cost and logistical complexity of the two-device alternative may be prohibitive to a design.

A non-cost-sensitive, time-critical system that is designed to run from high-speed RAM also simplifies the flash reprogramming chore. In this case, the flash is only used as non-volatile program storage when the product is turned off. During the boot process, the code is copied to RAM and normal execution continues from there. Since the flash is then inactive with regard to code execution, erasing and reprogramming it is simplified.

A compiler and linker that support relocatable code can aid the flash programming effort, as the same code can execute either from the flash or from RAM once code is copied there from the flash. Although it can be tricky to use emulators with program counter relative code, the technique is often used to accomplish flash reprogramming.

gram memory is its reprogrammability, specifically its *in-circuit* reprogrammability.

Everyone benefits with a system that is in-circuit reprogrammable. The end user of a flash-based product can obtain feature updates or bug fixes and easily apply them to his device, using nothing more than a standard PC, a serial cable, and a user-friendly interface. The engineer can (perhaps?) rest a little easier, knowing that if a bug survives through the testing process, it's not terribly burdensome to fix in the field. And the bean counters may benefit too, because in some instances the use of flash may even speed time to market—once the hardware and a certain core set of software features are complete, the product can be shipped and additional software features added in the future as time permits.

The downside of flash is that to achieve in-circuit reprogrammability, the software design effort is usually fairly complicated. Most types of flash have the requirement that during programming or erasing, no other accesses are allowed. If, for instance, an instruction fetch is attempted from flash during this time, the chip simply fails to respond. The implication here is that code execution needs to take place from some other memory device during reprogramming of the flash. The resource normally tapped for this task is RAM.

The majority of flash chips available to the embedded system designer today are byte- or word-writable, sector- or chip-erasable devices. This means that although code can be written to program individual bytes or words, when data is to be erased it must be done on whole sectors or even the entire chip at once. A sector is a fraction of the total chip density, for example, a 512K flash chip might have eight uniform 64K sectors.

A robust embedded system with flash program memory as part of its design needs to be protected from corruption that could render the device unbootable, and therefore unusable. Such a situation might arise if a power loss occurs right after the reprogramming software, executing from RAM, has just erased the entire flash chip in preparation of writing new program data.

Some flash chips contain "boot block" sectors that are typically a normal sector chopped up into smaller-sized pieces and placed at one end of the flash chip's memory space. The advantage of this configuration is that a small bit of code can be placed in the boot block that will never be erased. The idea here is that this code will contain routines for erasing and reprogramming the remainder of the flash chip. Because some or all of the boot sectors are never erased, the embedded device is protected from power-fail corruption during programming—a program is always present to boot the system and reinitiate the reprogramming process.

The smaller size of the boot block sectors allows the engineer to tailor the size of the boot code and eliminate wasted space. For example, a flash chip architected with 64K sectors might have a boot block consisting of one 16K sector, two 8K sectors, and one 32K sector at one end of its address space. This allows the creation of a 16K, 24K, or 32K boot program, without reserving an entire 64K sector for the boot code if not needed. Vendors produce different flash chips with the smaller boot block sectors at either end of the flash to meet the needs of different processors that have their reset vectors at either high or low memory addresses. See part A of Figure 1 for an example of a flash chip's architecture.

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

By far the hardest part of implementing in-circuit reprogrammable flash program memory is deciding upon a software approach to the problem.

FIGURE 1 Flash sector architecture

A AMD Am29F800BB
Sector Architecture

00000	Sector 0
03FFF	16 KB
04000	Sector 1
05FFF	8 KB
06000	Sector 2
07FFF	8 KB
08000	Sector 3
0FFFF	32 KB
10000	Sector 4
1FFFF	64 KB
20000	Sector 5
2FFFF	64 KB
30000	Sector 6
3FFFF	64 KB
40000	Sector 7
4FFFF	64 KB
50000	Sector 8
5FFFF	64 KB
60000	Sector 9
6FFFF	64 KB
70000	Sector 10
7FFFF	64 KB
80000	Sector 11
8FFFF	64 KB
90000	Sector 12
9FFFF	64 KB
A0000	Sector 13
AFFFF	64 KB
B0000	Sector 14
BFFFF	64 KB
C0000	Sector 15
CFFFF	64 KB
D0000	Sector 16
DFFFF	64 KB
E0000	Sector 17
EFFFF	64 KB
F0000	Sector 18
FFFFF	64 KB

B Author's Logical
Boot Block

00000	Sector 0
03FFF	16 KB
04000	Sector 1
05FFF	8 KB

Legend:

- Physical Boot Block Sectors
- Normal Sectors

Note: Addresses shown are byte addresses

factory. Most of all I wanted an understandable design, and to me that meant that it shouldn't be overly complex, and should be written in C with as little assembly language as possible.

The system I was converting was a Motorola 68332-based design in a moderately cost-sensitive product. Details of the system relevant to the flash effort included 64KB of RAM, one RS-232 port, 99% C code, 5V logic levels, a reset vector in low memory, and a movable interrupt vector table. The compiler/linker I was using would not produce relocatable code. A background debug mode (BDM) emulator was available for debugging.

The flash selected was an AMD Am29F800BB, a 5V, 512K x 16 or 1M x 8 device, which would be used in 16-bit mode. This flash is a boot sector device that has 19 sectors making up its 1MB address space. The second "B" in the chip name indicates that its boot sectors are at the "bottom" of the address space. The memory map for this chip is depicted in part A of Figure 1. The physical boot sectors (sector numbers 0 through 3) are sized so that the boot code can be 16KB, 24KB, or 32KB. If not used as part of a "logical" boot block, any remaining "physical" boot block sectors can be used as normal sectors.

Algorithm design

By far the hardest part of implementing in-circuit reprogrammable flash program memory is deciding upon a software approach to the problem. I thought about this for quite a while and managed to stumble up some blind alleys before arriving at a feasible strategy.

I knew that to be able to reprogram the flash I would have to have the code conducting the programming execute from the RAM of the embedded product. But where would that code come from? And how would I get it into RAM and start it executing? Running from RAM is not normally done in embedded systems, and consequently the tools that an engineer utilizes

Example conversion

Recently I was given the task of converting a PROM-based embedded product to flash program memory. Before beginning work, I summarized the goals of the redesign based on the constraints I had to work within. I needed to replace the product's PROM chip with flash, and allow for user-friendly in-circuit reprogram-

ming via an RS-232 serial port connection to a standard PC. The product should be protected from corruption in the event of a power failure during programming, and it shouldn't create any undue difficulties during debugging. Also, it needed to be self-contained, that is, the code to conduct the reprogramming needed to be embedded in the device as shipped from the



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

Only \$3,200*

LogicWave

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

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



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

www.agilent.com/find/bi

1-800-452-4844, Ext. 6918



Agilent Technologies

Innovating the HP Way

The strategy that I settled on placed three independent programs in the memory space of the flash chip.

(compilers, linkers, and emulators) are not usually cooperative when asked to do this. Ideally, the tools want to work with one large block of code

destined for one program memory device.

I also knew that to protect the product from corruption in the event

of a power failure during reprogramming, I would need to reserve some sectors in the flash for permanent code that could boot and restart the programming if it was somehow interrupted. This would have to be the first few sectors in the flash as 68332 designs require the reset vector to be at byte location 0x000004.

The strategy that I settled on placed three independent programs in the memory space of the flash chip. Each of these programs would have a different memory map for code and data storage.

A "boot loader" would run on power-up and initialize the system in a rudimentary way—just enough so that RAM and certain I/O are available. It decides which of the other two programs to transfer control to, based upon input from the user, and starts the selected program running. The memory map of the system when the boot loader is running can be found in part A of Figure 2.

The "application" is the program that runs to perform the function that the embedded system was designed to do. In the PROM version of the product, the application was the only program present. The memory map for the running application is shown in part B of Figure 2.

For lack of a better way of referring to it, I decided to call the program that runs from RAM the "RAM executive." The RAM executive conducts the reprogramming of flash based on data packets received on the embedded device's serial port from the PC-based reprogramming utility. The RAM executive is placed in the logical boot block of the flash along with the boot loader to protect it from accidental loss. For the memory map of the system when the RAM executive has control, see part C of Figure 2.

Details

Looking at the physical boot sector architecture of the flash chip in Part A of Figure 1, I decided to make my logical boot block 24KB, as shown in Part

Adaptable In-Circuit Emulators

We change so you don't have to.

iC1000
8 bit
Emulator



iC2000
8,16,32 bit
Emulator



iC3000
BDM, JTAG
Emulator



winIDEA
Software
Toolset



Support for
68HC05 08 09 11 12 16,
68K, 683xx, MCORE,
MPC, 8051, 80x86, C500,
Z80/180, CR16, PIC, ST7,
and many more.

Preserve your investment
in time and capital with
universal emulators from
iSYSTEM.

A complete line, from low-cost
BDM/JTAG to high-end
full-function ICES.

Support for over 400 micro-
controllers with a simple swap
of the POD or software setup.

Driven with an intuitive
development environment
integrating all popular
compilers.

Thousands in use world-wide.

Call for your free demo CD.



iSYSTEM
www.isystem.com

America	1 (888) 543-5300
Europe	49 (8131) 70610
Scandinavia	46 (40) 459571
Asia Pacific	82 (2) 2645-0386

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.

Complete solutions

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



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.

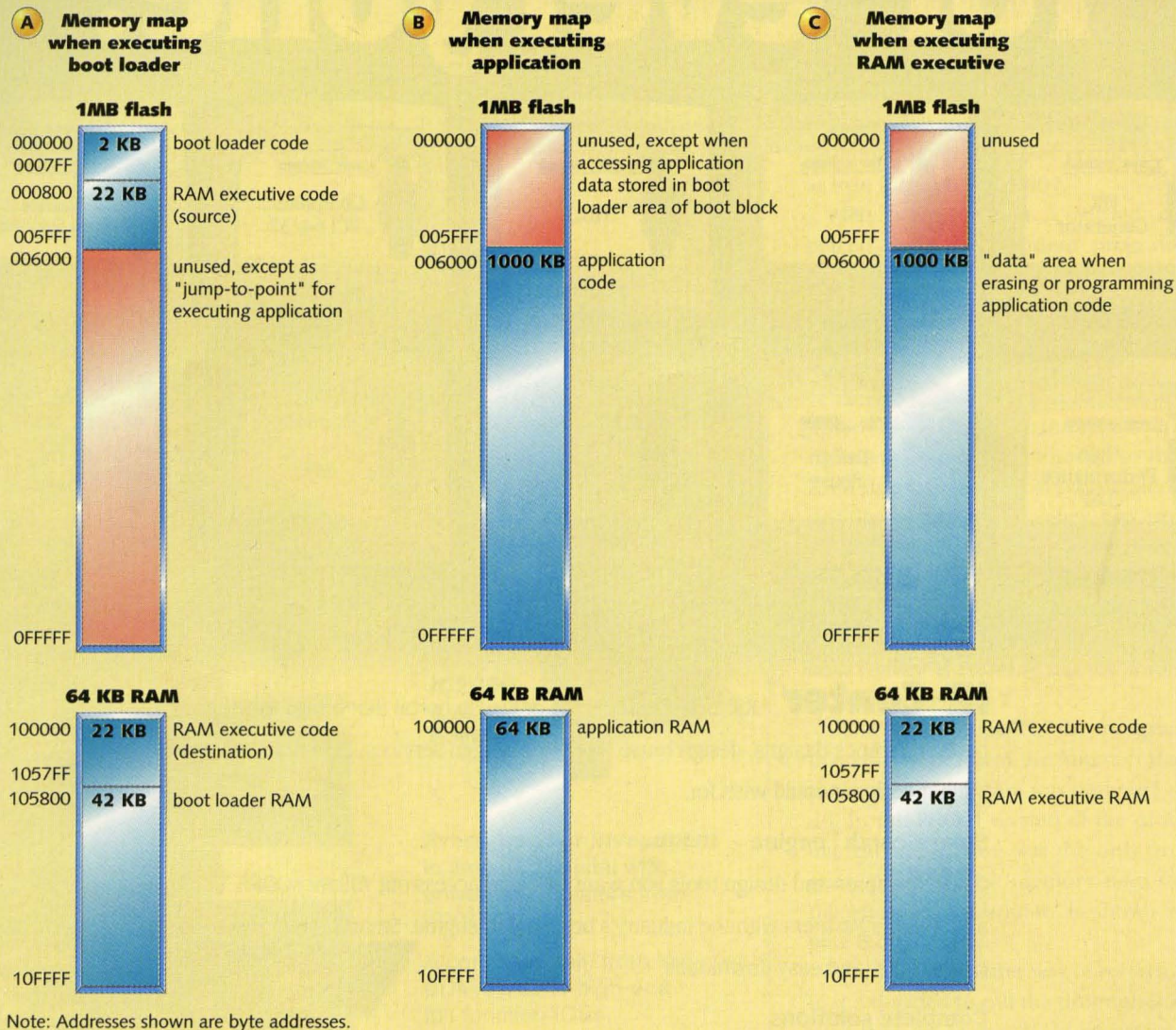
Visit www.xilinx.com/ipcenter today and get the IP that clicks for you.



The Programmable Logic CompanySM

www.xilinx.com

FIGURE 2 Example system memory maps



B. I tried to err on the pessimistic side, figuring that I could always pull back to 16KB if the combined size of the boot loader and the RAM executive ended up less than that. In actuality, I ended up staying with 24KB, as the boot loader and RAM executive together weighed in at roughly 20KB. As part A of Figure 2 shows, the boot loader was assigned 2KB and the RAM executive 22KB of the boot block's code space.

With respect to actual coding, I decided to start where the embedded system starts when it powers up—the boot loader. The boot loader does

very little. First, it configures and enables the RAM chip select, initializes its own zero and non-zero RAM variables, and sets up a small stack. Then it determines what to do next: either run the application program or the RAM executive. It does this by enabling the I/O ports to read the product's keypad. If the 1, 4, and 7 keys are held down simultaneously when the device powers up, the boot loader copies the RAM executive code data from its known location in flash to RAM the boot loader itself is not using. It then jumps via the reset

vector to the RAM copy of the RAM executive:

```
if (scan_keypad() ==
    SCAN_CODE_FOR_1_4_7_COMBINED)
{
    char * pRamExecSrc =
        (char *)0x000800;
    char * pRamExecDest =
        (char *)0x100000;

    /* copy the RAM executive
       embedded in the flash boot
       block to RAM not being used
       by the boot loader: */
}
```


Got another date with
the old logic analyzer tonight?



COMPUTING
TELECOM
VIDEO



TLA 600 — starting at \$7,000* You keep looking, but you can't find the problem. Time for a state-of-the-art, affordable logic analyzer that provides 500 ps timing simultaneous with up to 200 MHz state on every channel through the same probe? And a Windows interface? That's detail, accuracy, and ease that'll get you home in time for dinner. For a free Solutions Brochure, call 800-426-2200 x3039 or visit www.tektronix.com/TLA600

*TLA 600 Series starts at \$7,000 (manufacturer's suggested retail price). © 2000 Tektronix, Inc. All rights reserved.
Tektronix and the Tektronix logo are registered trademarks of Tektronix, Inc. All others are properties of their holders.

Tektronix®

When the RAM executive starts running, it also makes no assumptions about what the boot loader has done previously and executes its own startup code.

```
while (pRamExecSrc <=
(char *)0x005FFF)
{
    *pRamExecDest++ =
    *pRamExecSrc++;
}

/* make the jump to the RAM
copy of the RAM executive
via its reset vector: */

asm("move.l (0x100004),%a0");
asm("jmp (%a0)"); /* RAM exec */
}
```

Since the boot loader uses RAM at address 0x105800-0x10FFFF for its own use, that leaves RAM at 0x100000-0x1057FF unused to serve as the receptacle for the RAM executive code. And since the highest code address for the boot loader is 0x0007FF, the flash-based copy of the RAM executive code resides at 0x000800-0x005FFF. Again refer to part A of Figure 2.

If the special 1-4-7 key combination is not held down on power-up, the normal application code is run:

```
else
{
    /* make the jump to the
application via its
reset vector: */

    asm("move.l (0x006004),%a0");
    asm("jmp (%a0)"); /* app */
}
```

Because the RAM executive and the application are independently debuggable, each has a vector table at its code starting address. For the former, this address is 0x100000; for the latter, 0x006000. The reset vector is the four bytes at byte offset 4 in the vector table. This explains the use of 0x100004 and 0x006004 in the code

example above. That's really all there is to the boot loader.

The application program, aside from originating at an address that is non-zero, is exactly as it was before when the product was PROM-based. The application code makes no assumptions about what the boot loader might have initialized, and simply executes as if it were the only program in the device.

When the RAM executive starts running, it also makes no assumptions about what the boot loader has done previously and executes its own startup code. The RAM executive initializes the device's serial port, and waits for the PC-based reprogramming software to communicate with it. The idea here is that the PC program will instruct the RAM executive to erase the non-boot block sectors of the flash containing the old application program, then send the address and data information to allow the RAM executive to program the flash with a new application. Since no program is executing from the flash, the RAM executive is free to treat it as a normal memory device, with no access restrictions other than those imposed by the chip maker. Once the programming of the new application code is complete, the RAM executive forces a hard reset of the system, causing the boot loader to again gain control and start the new application program running. At this point, reprogramming has been accomplished. Any further detailed discussion of the RAM executive or the PC-based reprogramming software is beyond the scope of this article.

Debugging the three programs

Since all three programs—the boot loader, RAM executive, and application—are each self-contained, the act of debugging them is simplified. The

emulator cooperates reasonably, as it's not asked to do anything it's not accustomed to doing, such as running relocatable code.

Loading the boot loader code into the target system's emulation RAM and running it without a well-placed breakpoint would not be smart, however, because it will try to jump to a region of emulation memory that contains random data. When using the emulator to test the boot loader, the RAM executive and the application do not exist. To make sure the boot loader is doing its job, I load its code and place a breakpoint on the following line discussed before:

```
if (scan_keypad() ==
SCAN_CODE_FOR_1_4_7_COMBINED)
```

Running at full speed to this breakpoint, I can then step-over (not step-into) the calling of the `scan_keypad` function. If I do this while holding down the appropriate keys on the keypad, I can test the boot loader's ability to load and run the RAM executive. Although the `while` loop will copy garbage data to the RAM executive's destination in RAM, setting another breakpoint on the line

```
asm("jmp (%a0)"); /* RAM exec */
```

allows me the chance to load the code and symbol file for the RAM executive, being careful to not reset the CPU. From there, a single step takes execution to the first instruction in the newly-loaded RAM executive, with full C source-level debugging capacity.

The boot loader's execution path to the application program is tested in a similar manner. This time the special key combination is not held down, and a single breakpoint is needed on the line

```
asm("jmp (%a0)"); /* app */
```

When the emulator breaks on this instruction, the application program debug file can be loaded, and then run at full speed.

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

I'm the kind of engineer that dislikes duplication of code. Whenever I find myself writing the same code in two different places, I seek a way to combine the sections if it's reasonable to do so.

Once I determined that the boot loader was sending execution where it needed to go, that is, either to the start of code through the RAM executive's

reset vector or the application's reset vector, I forgot about the boot loader until time for final integration and test.

The RAM executive and the application program are also debugged independently of each other and the boot loader. The emulator software is used to prepare the appropriate chip select registers to allow the loading of these programs. For instance, since the RAM executive needs to run from a RAM chip that is disabled upon reset, that chip select has to be manually enabled to allow loading of the code. Also, the program counter at reset needs to be overridden in each case, otherwise the reset vector present in the flash chip will be used.

Code sharing among the three programs

I'm the kind of engineer that dislikes duplication of code. Whenever I find myself writing the same code in two different places, I seek a way to combine the sections if it's reasonable to do so.

Realizing that the three programs making up my flash scheme would have similar if not identical code in certain areas, I decided to have them utilize some of the same source files. This way, the boot loader and the application could share the same keypad reading routine, the RAM executive and the application programs could share very similar serial port handling code, all three programs could share similar startup code, and so on.

The vehicle by which this sharing is made possible is C's preprocessing directives. By creating the tags `BUILD_BOOT_LOADER`, `BUILD_RAM_EXECUTIVE`, and `BUILD_APPLICATION`, I was able to achieve my code-sharing goal.

For example, most of the serial port interrupt service routine is common to both the application and the RAM executive. The application code needs to understand how to process received data when the product is configured to be in Modbus (an industrial networking protocol) mode, but the RAM executive has no such need. To allow for this, I did the following in the source code file `serial.c`:



Our inspiration.

**Powerful tools. Great productivity.
Integrated price.**

Paradigm introduces all the tools you'll need for x86 integration in one package.

Paradigm C++ is alone in offering a complete integrated development environment that includes all the tools you need to get your x86 embedded application jump started. Editing, project management, debugging, compiler, assembler, version control and more, all fully integrated into the powerful Paradigm C++ development environment.

If you are tired of wasting time on non-integrated tools, then Paradigm C++ is where you want to be. Download a copy of Paradigm C++ from <http://www.devtools.com/pcpp> and see the future of x86 development tools today.

Paradigm

Paradigm Systems

3301 Country Club Road
Suite 2214
Endwell, NY 13760

1-800-537-5043

Phone 607-748-5966

Fax 607-748-5968

info@devtools.com

<http://www.devtools.com>



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

Download
**FREE
DEBUGGER!**



Embedded Systems
Conference
San Jose, CA
Sept. 26-28, 2000

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>

Having three separate programs instead of one makes it more of a task to get the programs into the flash chip.

```
INTERRUPT sci_handler(void)
{
    if ((RIE == ENABLED) &&
        (RDRF == RECEIVE_FULL))
    {
        /* receiver interrupt */

        /* get received data: */
        short RxData = RDR;

#ifdef BUILD_APPLICATION
        if (ModbusEnabled)
        {
            /* process RxData as part
             of a Modbus packet... */
        }
        else
#endif
        {
            /* process RxData as a
             standalone character... */
        }
    }

    /* ... */
}
```

Then, to compile the file for the application program, I used a command-line similar to this:

```
cc serial.c \
    -DBUILD_APPLICATION=1 \
    -DBUILD_RAM_EXECUTIVE=0
```

When compiling the RAM executive, the complimentary command-line looks like this:

```
cc serial.c \
    -DBUILD_APPLICATION=0 \
    -DBUILD_RAM_EXECUTIVE=1
```

This way, the application and RAM executive get slightly different serial routines compiled and linked in. The boot loader has no need to handle serial interrupts—in fact the boot loader doesn't even use interrupts—so `serial.c` is omitted from the file list in the

boot loader's portion of the make file, and the `BUILD_BOOT_LOADER` tag is not used in `serial.c`.

Code sharing has the advantage that any change affecting more than one of the three programs only needs to be done one time and in one place. The downside is that a change affecting all three programs triples the testing effort to validate the changed section.

Getting the three programs into flash

Having three separate programs instead of one makes it more of a task to get the programs into the flash chip. With PROM, typically one large program is burned into the device, beginning at the lowest address in the chip and extending upward as far as needed to accommodate the current size of the program.

With flash the situation is not so simple, as there are three separate S-record files that make up the flash chip's address space. The application program is like the PROM's single program, except that its starting address is no longer the lowest address of the chip—the boot loader program receives that honor.

When the RAM executive's program code is in place in RAM, its starting address is 0x100000, so the linker outputs an S-record file with this address as a starting point. But there is a problem with using this S-record file as the linker creates it—it can't be burned into flash because the address range of its records (0x100000-0x1057FF) doesn't coincide with the memory range of the flash (0x000000-0xFFFFF). From the memory map (see part A of Figure 2) it's clear that when embedded in the flash, the RAM executive must reside at 0x000800-0x005FFF. To achieve this, an S-record relocation program is used to change the addresses of the data bytes in the

file without impacting the data bytes themselves. An example of such a relocation utility is P&E Microcomputer Systems' `MOVE_S19.EXE` (www.pemicro.com/support_center/downloads/utilities/move_s19_downloads.exe).

Once the RAM executive has been relocated, it and the boot loader are combined into one S-record file—no problem results since their address ranges do not overlap. The new S-record file represents the boot block code and can be delivered to the flash vendor or distributor for burning into blank flash chips. Those chips, once soldered to the printed-circuit board of the embedded product, will then be ready to serially receive the data from the application program's S-record output via the PC connection.

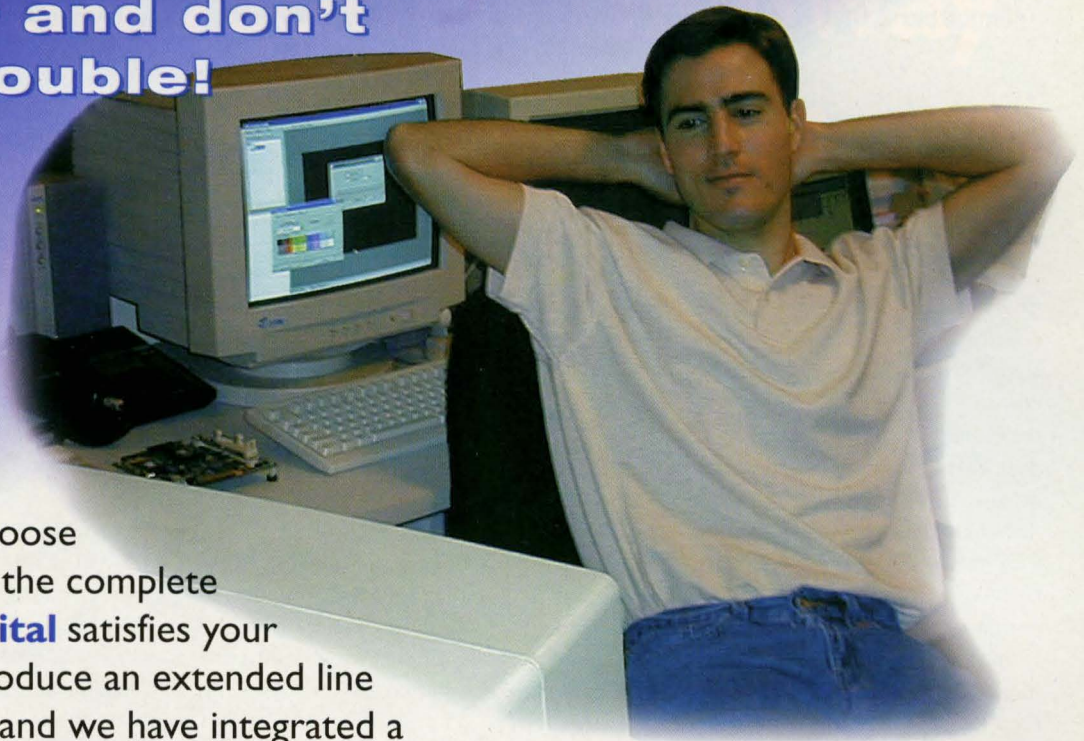
For application code that is fairly mature and stable, the three S-record files can be combined into one and provided to the flash vendor, thus avoiding the time-consuming process of serially-uploading the application program on the product production line. An alternative to this is to solder on blank flash chips and program all three programs at once on the production line using the BDM port of the CPU and either a custom-written or a commercial flash-programming PC software package.

To allow end users to update their devices with new application code, the S-record for the current application code can be placed on the company Web site or e-mailed to customers who have requested bug fixes or have paid for feature updates.

It goes without saying (but I'll say it anyway) that the boot loader and the RAM executive need to be fully tested before being released to the flash vendor or the production line. Bugs in these programs can prevent field reprogramming of the embedded product, the whole point of this exercise. If this makes you squeamish, it is possible to design the system so that the RAM executive can be uploaded to RAM via the serial link, before the application data is sent.

Most Programmers Want ...

Integrated tools that work together and don't cause trouble!



It makes sense to choose the 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 MBytes/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 (200KB). 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

A potential disadvantage to this approach is that the emulation is no longer pure, that is, code developed on the emulator may not run the same way when moved to flash.

The boot loader still needs to be bug-free, however.

Emulator issues with flash

The product upon which I developed my flash programming scheme had a connector to allow for RAM expansion. Extra RAM was used either by the customer for data storage (it was battery-backed), or in-house for emulation RAM during code development using a BDM emulator.

When the embedded product was PROM-based, the boot chip select was connected to both the PROM and the emulation RAM. Since both devices cannot share a single chip select, this meant that when emulating, the socketed PROM had to be removed and a special jumper had to be soldered into a RAM board to be dedicated from that point on as an emulation RAM board. This had the advantage that if I had working code running with the emulator, that code was guaranteed to run when I burned a PROM, since the PROM and emulation systems were identical, except for the physical program storage devices.

Emulating on the flash version of the product was a different story, however. In that case the flash could not be taken out of circuit because it was a surface mount component permanently soldered to the PC board, so any emulation RAM could not share the flash's chip select. The problem could have been solved using moveable jumpers, but a better and less error-prone solution was to use the power of the emulator to configure chip select registers upon receiving a reset signal. Normally, when a reset signal is received, all chip selects except the boot chip select are disabled. If I used the emulator to write into the registers to disable the boot chip select and enable the emulation RAM's chip select

upon reset, I could trick the system into allowing me to load code into the emulation RAM and run it as if it were located in the boot device (the flash).

A potential disadvantage to this approach is that the emulation is no longer pure, that is, code developed on the emulator may not run the same way when moved to flash. This occurs because of the artificial situation created with the emulator by mucking with the chip select registers. Fortunately, most problems that result from this are easily found and corrected, usually right after the flash is programmed from working emulator code and the system fails to power up correctly.

Writing to flash from application code

The application program in the product I was converting to flash required that a few words of calibration data be stored in non-volatile memory. This data was specific to the printed circuit board, so each product sold contained slightly different calibration information. Once calibrated during final checkout of the product, this data never changed. In the days of PROM, the calibration data was stored in a serial EEPROM. With the advent of flash, however, it was decided that this data, along with a product serial number, should be stored in fixed, unused locations high in the boot loader's area of the flash boot block. The data would be programmed into flash from the product's setup mode.

In this situation, the application program is running from the flash when the designated locations need to be programmed. Again, executing the instructions to program the flash from RAM is the only way, but this time execution has to come back to flash and resume where it left off. My

solution to this problem consisted of several steps.

First, I created a C function called `ProgramFlashWordFromApp`. This function manipulates only data passed to it as arguments (no global variables) and calls no other functions. The importance of this will become clear later. Writing the function was straightforward, as was debugging it to make sure it programmed some test locations in the flash correctly. A much-simplified version (just to illustrate the concept) of the `ProgramFlashWordFromApp` function is:

```
#define FLASH \
    ((volatile short *)0x000000)

void ProgramFlashWordFromApp(
    long WordOffset,
    short WordData)
{
    FLASH[WordOffset] = WordData;

    /* code to unlock the flash,
    wait until the location has
    been programmed, do error
    checking, return success or
    failure, etc. omitted here
    to keep the machine code
    example small */
}
```

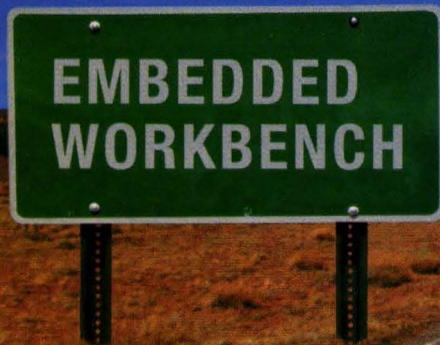
After debugging but while still running the emulator, I examined my C function in the disassembly window:

```
ProgramFlashWordFromApp:
4E560000    LINK.W A6,#0x0
202E0008    MOVE.L (0x8,A6),D0
222E000C    MOVE.L (0xC,A6),D1
D080       ADD.L D0,D0
2040       MOVEA.L D0,A0
3081       MOVE.W D1,(A0)
4E5E       UNLK A6
4E75       RTS
```

Next I copied the machine code text for the function from the disassembly window into my favorite source code editor. After a little text manipulation, I had a group of hex words that I could comma-delimit, put curly braces

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 breakthrough 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

Moving a system from PROM to flash program memory is a worthwhile effort but not one to be undertaken lightly.

around, and name as a C array called ProgramFlashWordFromAppViaRam:

```
/* NOT const ! */ short
ProgramFlashWordFromAppViaRam[] =
{
    0x4E56, 0x0000, 0x202E, 0x0008,
    0x222E, 0x000C, 0xD080, 0x2040,
    0x3081, 0x4E5E, 0x4E75
};
```

I placed this new C array in the flash source code file. I now had a RAM-based C array containing a machine language subroutine to program a word into the flash. I had to resist the urge to put a "const" qualifier on this array of constants—doing so would have resulted in the array being placed in flash. That would have been very undesirable; it needed to be in RAM so that it could execute from there.

A side note to the purists: I could have written the subroutine in assembly language, but I didn't for a few reasons. First, the actual function is considerably longer than the one I show here, and I'm lazy. Second, the C compiler can do a better job at writing the assembly than I can unless I really try, and I didn't want to go to the trouble. Third, did I mention that I'm lazy?

Next I analyzed the assembly code of the function to determine how it accessed the arguments that were pushed onto the stack before the call. At the point in the application code where I needed to write data into flash, I pushed my address and data arguments onto the stack manually using some inline assembly, and coerced the compiler into calling the first location in my RAM-based array as if it was a normal function:

```
asm("move.w %0,-(%sp)" ::
```

```
"m"(Data));
asm("clr.w -(%sp)");
asm("move.l %0,-(%sp)" ::
    "m"(Offset));
asm("jsr "
    "ProgramFlashWordFromAppViaRam");
```

Mission accomplished—the flash could now be written to from the application code.

I commented out the original C function, but I did not delete it, as it was the only copy of the source code for the machine code array. Also, whenever I use a technique such as this (which isn't often!), I try to comment it well. I've been on the maintenance end of things too often to not do this.

Had I used any global variables in my original C function, this approach would not have been practical. The linker takes the liberty of moving the addresses of global RAM variables around in memory as code is under development. By using only stack-based variables in my function, the linker can move globals around all it wants and my machine language subroutine remains valid. With simple code that uses only stack-based data and makes no function calls, even a C compiler that cannot produce a *completely* relocatable program often produces *locally* relocatable code. The fact that the machine code generated from my C function used only relative branching was key in making this technique viable.

One of the caveats to using this method is that interrupts must be disabled during the ProgramFlashWordFromAppViaRam "function" call. Since all of the interrupt service routines reside in the flash, they are unavailable during this method of flash programming. This was not a problem in the product I was working on since the data bytes that are written in this man-

ner are programmed while the product is in "setup" rather than "run" mode, when the system is performing no time-critical tasks. A related caveat is that this method should probably not be used to write a lot of data into flash, since running with interrupts disabled for an extended period of time will adversely effect a real-time system.

An emulation issue to be aware of when using this technique is that the flash chip select cannot be disabled and its address range overlaid with emulation RAM. To be able to read or write data, the flash chip must be present in the memory map during emulation. An easy way to accomplish this is to locate the emulation version of the program at a normally unused memory range and reenable the flash chip select at its normal address.

Let there be light

Moving a system from PROM to flash program memory is a worthwhile effort but not one to be undertaken lightly. Significant development time and planning need to be dedicated to such a change. Deciding on an algorithmic approach that works and is easily debugged can be challenging. It's my hope that this article's example and ideas on how to make the conversion have made the task seem less daunting, and have shed some light on the more arcane aspects of in-circuit flash reprogramming.

esp

Scott Sumner is currently a senior project engineer working on automotive test systems at the Detroit Engineering Center in Rochester Hills, MI. He has been involved with designing embedded and PC-based systems since graduating from Lawrence Technological University with a BSCS in 1988 and a BSEE in 1991. He welcomes contact and can be reached via e-mail at sasumner@bigfoot.com.

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®

© 2000 I-Logix Inc. Java is a registered trademark of Sun Microsystems, Inc.

I-Logix®

REAL TOOLS FOR THE REALLY HARD STUFF

We don't have to remind you how important it is to get your design into the market quickly and efficiently. But we would like you to know that our wide range of SoC verification tools excel at doing just that.

There's ModelSim®, for example, the leading tool for mixed language simulation. XRAY®, the only product to support debugging of multiple cores. Seamless® Co-Verification Environment, the industry standard by far. And a host of other hardware, software, IP and consulting solutions designed to help you increase productivity, reduce steps and slash verification time across every SoC development stage.

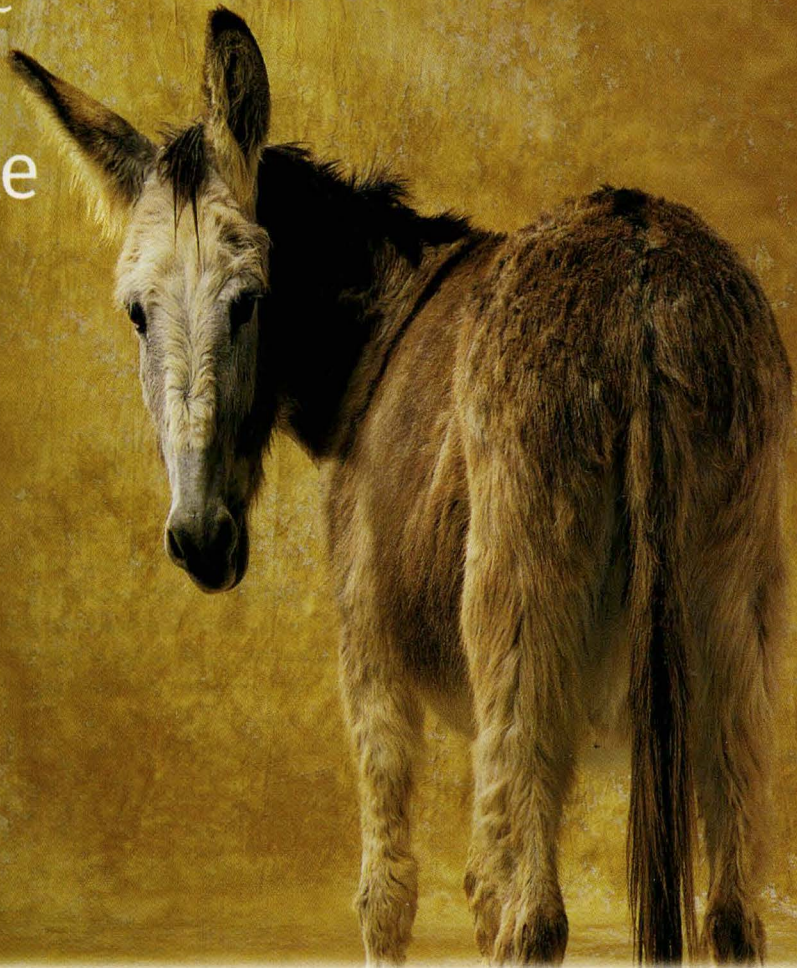
So be sure to start with our tools on the front end. Then you won't have to worry about the other one. For more information, call **1-800-547-3000**, or visit www.mentor.com/soc to get our new online SoC newsletter.

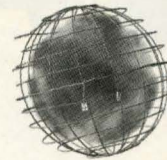
SoC VERIFICATION
TOOL SET

Mentor
Graphics®

Our SoC verification tools

can save the
only thing
more valuable
than your
time.





Embedded Internet Tools

Real-time Linux OS

Linux/RT is a real-time version of the Linux operating system. The Linux R/T distribution includes subsystems which designers can combine to handle a variety of application requirements, from small footprint to full-featured architectures. The Resource Kernel subsystem supports fixed-priority scheduling (with 256 priorities) and priority inheritance. It also supports high resolution timers and clocks, as well as a temporal firewall. The Linux/RT distribution is available free for downloading at the company's Web site. It is also available in three different editions for purchase, all of which come with limited installation support. Pricing for these ranges from \$49 to \$199.

TymeSys Corp.

Pittsburgh, PA
(412) 681-6899
www.timesys.com

Internet access package

The USNET internet access package enhances the company's real-time, embedded protocol TCP/IP with Internet and Web enabling technology. This capability may be used on private LAN/WAN networks or on the Internet. It has a small footprint, as well as APIs designed for embedded applications. In a typical Web-enabled application running an HTTP server on a 16-bit target, the USNET protocol stack, together with USENET IAP, requires less than 45K of code space. Dynamic update support is another component of the product. Dynamic page update allows web pages to be updated without rebuilding the application and re-downloading the target. Users at any remote site can create new pages and submit them to the Web server. Updating is done by the File

Upload form feature of the browser. New web pages can contain text, images, Java applets, and JavaScript among other items. Version 2.56 of USNET is available now, with pricing from \$7,500.

US Software

(800) 356-7097
Hillsboro, OR
www.ussw.com

Software platform for internet appliances and entertainment devices

BeIA is a software platform for appliances that deliver information and entertainment over the Internet. It allows Internet appliances to be configured as information devices such as personal online portals or financial management monitors. The adaptive configuration of BeIA also extends to entertainment devices such as movie or music download and playback systems. BeIA's multi-layered architecture is designed to offer application developers design flexibility. At its foundation is a set of system functions, which talk directly to the individual hardware designs. Layered above the core functions are an integrated Internet browser and a Java virtual machine. The next level of the BeIA platform focuses on streaming media services. At the final level of the BeIA architecture lie application services and integration services. BeIA is available now.

Be Inc.

Menlo Park, CA
(650) 462-4100
www.be.com

Real-time scheduler for Linux

A real-time scheduler is now available for the Linux kernel. It's integrated

into the standard Linux kernel, but with enforced hard prioritization. This scheduler executes before the standard Linux scheduler and it examines and dispatches only the highest-priority real-time entity that is ready to run. If no real-time entity is available for execution, or none has been specified as real-time, then scheduling falls through to the standard scheduler. The real-time scheduler is available now in source form on the company's Web and FTP sites.

MontaVista Software, Inc.

Sunnyvale, CA
(408) 328-9200
www.hardhatlinux.com

Wireless Internet appliance

The Internet Appliance Reference Platform (IARP) is a portable wireless Internet appliance used to browse the Internet and to send and retrieve e-mail. It's based on the Linux or Windows CE operating systems. It features either Intel StrongARM or LinkUp Systems ARM microprocessors. The 640 x 480 high contrast, VGA/SVGA LCD display with integrated touch screen allows the operator to use a built-in stylus for data entry and signature capture or, if the user prefers, a pop-up keyboard for alphanumeric input. Audio output is provided by means of a built-in speaker, which supports user feedback sounds and allows Internet audio to be downloaded and played. The supplied cradle functions as a battery charger and connects to a full keyboard and mouse for e-mail use.

Accelent Systems, Inc.

Akron, OH
(330) 864-2300
www.accelent.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.

HI-TECH Software
7830 Ellis Rd.
Melbourne FL 32904
Ph 800 735 5715
Fax 407 722 2902

www.htsoft.com





Dan Saks

Arrays as Function Parameters

As I explained last month, many C and C++ programmers don't quite understand how arrays really work.¹ Much of the confusion stems from the type conversion rule that quietly turns expressions of array type into pointer types.

For example, in a subscripting expression such as `x[i]`, `x` must be an expression of type "pointer to `T`" for some type `T`. If `x` was declared as a "pointer to `T`," it remains as such. If `x` was declared as an "array of `T`," the compiler converts `x` to a "pointer to `T`" before compiling the rest of the expression. The compiler doesn't actually change `x` into a pointer; it creates a temporary object of type "pointer to `T`" whose value is the address of `x[0]`. Then it uses that temporary object for the rest of the expression.

Although both C and C++ let you declare objects of array type, it's hard to tell arrays from pointers because it seems that the compiler always converts an array to a pointer whenever you use an array in an expression. In fact, the implicit array-to-pointer conversion does not occur in two contexts—when the array is the operand of either the `sizeof` operator or the unary `&` (address-of) operator. Last month, I showed how you can use these operators to tell that an array really is an array rather than a pointer.

The implicit conversion rule is not the only source of confusion about arrays and pointers. More confusion arises from array declarations. Although you can indeed declare objects with array types, sometimes an

array declaration actually declares an object with a pointer type.

Array parameters

A declaration such as:

```
T x[N];
```

declares `x` as an object whose type is

in the same scope is not an error. It's just two declarations for the same function. C lets you declare a function more than once in the same scope.

In C++, we say that both functions have the same signature, which, in this case, is `T *`. (The signature of a function is primarily the sequence of types in the function's parameter list.²) C++ lets you declare a function more than

The rules governing how arrays and pointers really work have been known to cause confusion. Here's some clarification.

"array with `N` elements of `T`," where `N` is some previously declared integer constant, and `T` is some previously declared type. If this declaration declares a global object, a local object in a function body, or the member of a struct, union, or C++ class, it really has array type. However, if this declaration appears (without the trailing semicolon) in the parameter list of a function declaration, as in:

```
int f(T x[N]);
```

then it declares an object whose type is "pointer to `T`." In short, any parameter declared with type "array of `T`" actually has type "pointer to `T`."

C does not permit function overloading, but declaring both:

```
int f(T x[N]);
int f(T *x);
```

once in the same scope, except in class scope.

Since a parameter declared as an array really has a pointer type, the compiler ignores the array dimension. Thus, the array dimension is optional. For example:

```
int f(T x[]);
```

is just another declaration for either of:

```
int f(T x[N]);
int f(T *x);
```

All three function declarations have the same signature, which, once again, is `T *`.

Last month, I explained that you can tell that `v` declared as:

```
int v[10];
```


In both C and C++, array-to-pointer conversions occur only in the first array dimension of a multi-dimensional array.

is an array by using `v` as the operand of the `sizeof` operator. As long as `v` is not a function parameter, `sizeof(v)` yields `10 * sizeof(int)`, not `sizeof(int *)`. However, if `v` is a function parameter, then `sizeof(v)` is indeed `sizeof(int *)`. For example, given:

```
size_t g(int v[10])
{
    return sizeof(v);
}
```

a call to `g` always returns `sizeof(int *)` rather than `10 * sizeof(int)`.

Function `g` appears to accept only arguments of type “array of 10 elements of `int`,” Despite its appearance, `g`’s parameter has type “pointer to `int`” and it accepts any argument of that type. For example, given:

```
int w[20];
```

the call `g(w)` will compile. When `w` appears in the call expression, the compiler converts it from type “array of 20 elements to `int`” to “pointer to `int`,” which exactly matches `g`’s parameter type. Given:

```
int *p = a;
```

the call `g(p)` compiles just as well.

Another way that you can tell that an array parameter is really a pointer is by using the parameter as the operand of the address-of operator. For example, as long as:

```
int v[10];
```

is not a function parameter, `&v` yields a result of type “pointer to array with 10 elements of `int`,” not “pointer to pointer to `int`.” Thus:

```
int (*p)[10] = &v;    /* yes */
```

compiles without error, but:

```
int **q = &v;          /* no */
```

generates at least a warning, if not an error. However, if you declare `v` as a function parameter, then the situation reverses. For example:

```
int **g(int v[10])
{
    int (*p)[10] = &v;    /* no */
    int **q = &v;         /* yes */
    ...
}
```

Multiple dimensions

In both C and C++, array-to-pointer conversions occur only in the first array dimension of a multi-dimensional array. For example, a two-dimensional array converts into a pointer to a one-dimensional array, not a pointer to a pointer to a single element.

More precisely, a multi-dimensional array is really an array of arrays. If you declare an array object (not a parameter) as:

```
T x[M][N];
```

then `x` has type “array with `M` elements of array with `N` elements of `T`,” When you use `x` in an expression (other than as the operand of `sizeof` or unary `&`), the compiler treats it as an expression of type “pointer to array of `N` elements of `T`.” Thus:

```
T (*p)[N] = x;
```

compiles without complaint, but:

```
T **q = x;
```

provokes at least a warning for a questionable pointer conversion.

A parameter declared as an n -dimensional array is actually a pointer to an $n-1$ -dimensional array. For example, a function declared as:

```
void f(T x[M][N]);
```

has the same signature as a function declared as:

```
void f(T (*x)[N]);
```

That signature is `T (*x)[N]`, which is a single parameter of type “pointer to array of `N` elements of `T`.”

Be careful not to confuse a “pointer to an array” with an “array of pointers.” For example, a typical declaration for function `main` looks like:

```
int main(int ac, char *av[]); (1)
```

which declares `av` as an “array of pointer to `char`.” In reality, it has type “pointer to pointer to `char`.” Thus:

```
int main(int ac, char **av); (2)
```

is equivalent to the previous declaration for `main`. However, neither of the declarations above is equivalent to:

```
int main(int ac, char av[][]);
```

Here’s why. The compiler ignores only the first dimension in an array parameter declaration. Only the first dimension is optional. The parameter declaration:

```
char av[][]
```

omits the second dimension. This is a syntax error. Even if it weren’t an error, the compiler would transform the parameter declaration into an equivalent pointer declaration of the form:

```
char (*av)[]
```


This is a "pointer to array of char." The compiler does no more array-to-pointer transformations. One is all you get.

In contrast, the parameter declaration:

```
char *av[]
```

as in (1), declares **av** as an "array of pointer to char," not a "pointer to array of char." After one array-to-pointer transformation, it becomes a "pointer to pointer to char," which is its actual type.

Recapping the rules

In summary, here are the rules that seem to cause all the confusion about arrays and pointers:

- An array declaration not in a parameter list really declares an array
- An array declaration in a parameter list really declares a pointer
- Except when it's the operand of **sizeof** or unary **&**, an expression of array type is converted to an expression of pointer type

When reduced to this short list, it's hard to see why the rules are so confounding. Nonetheless, they are. **esp**

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 and remains an active member. With Thomas Plum, he wrote C++ Programming Guidelines. You can write to him at dsaks@wittenberg.edu.

References

- 1 Saks, Dan "Is an Array Really Just a Pointer?" June 2000, p.15.
- 2 Saks, Dan "Function Signatures and Name Mangling," August 1999, p. 79.

Compact OEM Solutions!

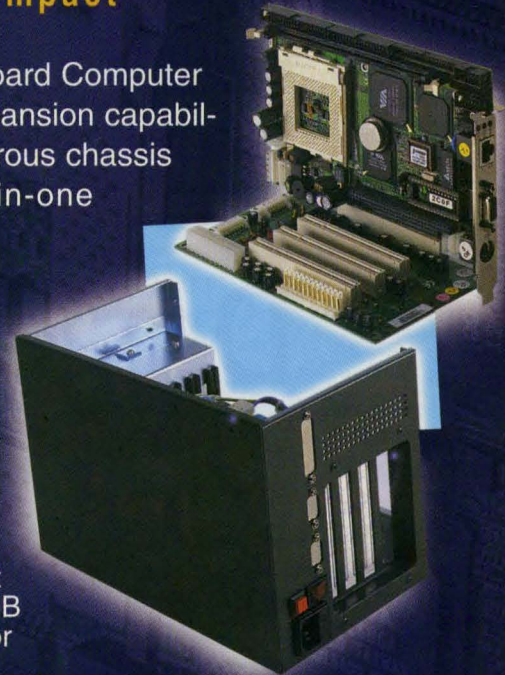
AXIOM™

SBC8260 with Compact Chassis

Half-size PIII Single Board Computer and advanced PCI expansion capability supported by numerous chassis options - it's your all-in-one OEM solution!

SBC8260 Celeron Socket 370 PCI Bus SBC with VGA/Ethernet

- Socket 370 with up to 500MHz processor
- C & T 69000 AGP with 2MB SDRAM
- 10/100 Base-T Ethernet
- DiskOnChip up to 144MB
- 16-bit PC/104 Connector



SBC8261 Socket 370 with Ultimate Performance Based on ISA Bus

SBC8243 486-based SBC with On-board AMD Am486DX5-133 CPU



SBC8252 Pentium ISA Bus CPU Card with VGA

www.axiomtek.com

Axiom Technology, Inc.
(888) GO - AXIOM





Design by Contract for C Programmers

Design by contract is a programming technique that allows the semantics of a class's behavior to be specified. This technique helps you produce more reliable software at the cost of additional formalism. Here's how to apply this traditionally object-oriented technique to procedural languages such as C.

Generally you think of design by contract (DBC) in relation to object-oriented languages such as Eiffel, C++, and Java. However, you can apply the same techniques to procedural languages such as C, or even to assembly. This article explores the application of DBC techniques to the C language, with special consideration to the constraints under which embedded programmers often must work. We'll explore possible implementations of DBC and demonstrate how these techniques can be used to effectively increase software quality.

Design by contract

DBC was pioneered by Bertrand Meyer, the father of the Eiffel pro-

gramming language.¹ Eiffel is a "pure" object-oriented language that directly supports DBC. The crux of the technique demands that each method of a class specify the preconditions that must be met before a particular method is executed, as well as the postconditions that must be met afterward. In addition, class invariants may be specified. Invariants are conditions that must be met before and after any method is executed for that class.

By specifying these three sets of behavior, a *contract* is established between the *supplier object* (the object whose methods are being invoked) and the *client object* (the object invoking those methods). In a nutshell, if the client object will ensure that the preconditions are met, the supplier object ensures that the postconditions and

invariants are met. Let's examine these three types of constraints in detail.

Preconditions. Each method must specify the preconditions that must be true before the method is executed. It is the responsibility of the client object to ensure that the preconditions are met before invoking the relevant method of the supplier object. A common example from the C/C++ world is to check method parameters for NULL pointers.

Postconditions. Each method must specify the postconditions that must be true after the method is executed. It is the responsibility of the supplier object to ensure that the postconditions are met when the relevant method is completed. The postcondi-

tions are a guarantee to the client object that the supplier object is living up to its end of the bargain. Again taking an example from the embedded world, a valid postcondition for pushing a message into a message queue is that any task waiting on that queue become ready to run.

Invariants. Invariant conditions are those conditions that must exist for all methods of a class. Adding an invariant for a class is equivalent to adding the same condition as a precondition and a postcondition for all externally visible methods of that class. The supplier methods must ensure that all class invariants are met before returning control to the client object. For instance, an invariant condition for a set of routines that manipulate a doubly linked list would be that the forward and backward links for each entry in the list must point to another valid list entry or be a NULL pointer.

The benefits of DBC are straightforward and quite powerful. Because an object's behavior may be explicitly specified, a client object need never guess what a supplier object is doing. Since the supplier objects' preconditions, postconditions, and invariants are explicitly defined, the process of writing test drivers for supplier objects tends to be much easier. As a result, DBC techniques generally lead to well-behaved and well-tested objects. In addition, since the supplier object's behavior is explicit, the interactions between objects are also generally more stable and reliable.

As with many design principles, much of this may seem like common sense. In fact, many of you may already be employing techniques like DBC in your embedded software. If you are already working with C++, these tech-

niques may be applied to the methods of a class that you are developing. If C is your language of choice, these techniques may be applied to a set of related functions. For example, a set of functions with high cohesion, such as a device driver, generally operates on the same set of data. As such, you can think of a device driver as a simple "object" and apply the DBC techniques to that object.

For embedded programmers, the amount of extra processing power required to test for preconditions and their ilk may be too expensive as part of normal operating procedure. This implies that the DBC code should be able to be turned on and off. In a like manner, the amount of code space required for DBC tests may also be too costly for memory-strapped systems. In principle, not including the assertions into production code is acceptable because of all the test cycles that you've already run on your system. However, your project management team may have other ideas, usually based on how badly they have been burned in the past. In any case, DBC assertions must be able to be compiled out.

For maximum portability, the code that is executed when a condition is not met must be easily changed. Indeed, this code is normally system-specific. For application layer code, a print statement to the standard error file and a quick `exit()` may be all that is needed. For driver layer code, perhaps a call to the kernel's `panic()` function is required. Other possibilities include logging to an error file, popping up a dialog window, forcing a core dump, or jumping into the debugger.

DBC and C

The standard C library provides the `assert()` macro that may be employed

to gain some of the benefits of DBC. However, the `assert()` macro has limits to its usefulness. According to the standard, `assert()` must be implemented as a macro, and if the test condition is true, a message will be printed to the standard error file and the `abort()` function will be called. Unfortunately, because `assert()` is part of the standard C library, it cannot be tailored for individual use. Our solution will bypass the `assert()` macro and develop our own more powerful framework.

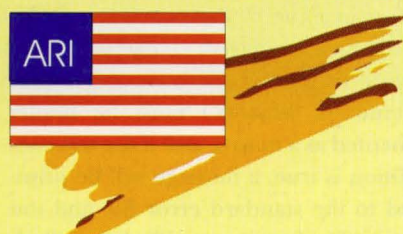
First we'll need some compile-time switches to allow the DBC code to be compiled in or out. For simplicity, let's use `DBC_PRE`, `DBC_POST`, and `DBC_INV`. Each of these compilation switches will turn on or off a specific type of test in the source module that is under compilation.

You may be wondering why we have three separate types of compilation switches. Three separate switches provide the flexibility to check one type of assertion, such as preconditions, without checking the others. We'll also define a DBC compilation switch as a convenience macro that can easily turn on or off the other three switches.

Next we'll need the macros for testing the preconditions, postconditions, and invariants. Predictably, these are named `PRE()`, `POST()`, and `INV()`. The definition for the `PRE()` macro is shown below, and the other two macros follow in a similar vein.

```
#define VALUE(x) x
#if defined(DBC) && defined(DBC_PRE)
    #define PRE(x) { if (testPre) \
        PreFailed(VALUE(__LINE__), \
        __FILE__, #x); }
#else
    #define PRE(x) {}
#endif
```


American Raisonance



"Liberate Yourself"

Introducing American Raisonance. Your new supplier of development tools for the Philips XA, 8051, and ST6 microcontroller families.

XA

Packages available with assembler, compiler, simulator and utilities for the Philips XA. Web distributed for immediate installation. One year technical support included in purchase.

8051

Available individually or combined with XA tools. Source Code compatible with Keil 8051 tools for easy migration to XA.

ST6

Free assembler and simulator. Optimizing compiler including RIDE and utilities for program development.

CAN

For CAN development, American Raisonance also distributes IXXAT CAN products and libraries for CAN-Open and DeviceNet.

Call today for introductory pricing!

877-812-6393

www.americanraisonance.com

This bit of preprocessor chicanery can be easily summed up: if either DBC or DBC_PRE is not defined, the PRE() macro will be expanded out to nothing. If both DBC and DBC_PRE are defined, then PRE(x) will be replaced by the following line:

```
if (testPre) PreFailed("the
    line no.", "the file name", "x");
```

The strings "the line no." and "the file name" will be expanded by the preprocessor to strings that contain the appropriate values. For example, if the file foo.c contained the following macro on line 13:

```
PRE(ptr != NULL);
```

then this line would be expanded out to the following construct:

```
if (testPre) PreFailed("13", \
    "foo.c", "ptr != NULL");
```

One of the requirements stated earlier was that DBC should be able to be turned on and off. However, performing this at compile-time is a rather coarse method of achieving our aims. Control variables, such as testPre, provide flexibility at run-time to allow the testing code to be disabled. This approach allows all of the test code to be compiled into the software. However, if testPre is zero, the PreFailed() function will not be called.

The system specific code for handling test assertion failures resides in PreFailed() (and its brethren PostFailed() and InvFailed()). The contents of this function depend entirely upon what the embedded programmer wants to do when an error is detected. All three functions may call the same code or each may have separate functionality, such as logging to a separate error log. For example, on a UNIX system, the following piece of code may be sufficient for PreFailed():

```
void PreFailed(int line, char
    *file, char *condition)
{
    /* Basically does what the
    * assert() macro does.
    */
    fprintf(stderr, "%s : Failure
        at line %d in file %s\n",
        condition, line, file);
    abort();
}
```

The particular implementation choices are left up to the embedded programmer. A standard implementation is to have the macros defined in a header file, such as dbc.h, and PreFailed() and its brethren defined in a separate C module (dbc.c). This module would be compiled separately and linked to the set of software that is to be tested.

Using DBC

Now that we've examined the framework, let's examine some of the ways that DBC techniques might be used. The easiest and most understandable use is to test preconditions for functions. A simple example would be a routine that generates a new string by concatenating another string a variable number of times. The prototype for such a function would look as follows:

```
char *dupstr(char *string, int
    numberOfDups);
```

To bulletproof this code, two reasonable preconditions are that the input string is not NULL and that the number of duplications is greater than or equal to one. These preconditions may be easily added by use of the PRE() macro:

```
char *dupstr(char *string, int
    numberOfDups)
{
    PRE(string != NULL);
    PRE(numberOfDups >= 1);

    /* body of function */
}
```


Notice that the supplier code is checking the validity of the preconditions even though the client code is responsible for guaranteeing that these preconditions are met. This is an artifact of this particular implementation of DBC in the C language. However, even in languages with explicit support for DBC techniques, such as Eiffel, the supplier object still controls what preconditions must be met, and the preconditions are stored in the supplier source code. By having the supplier object control its preconditions, the overall cohesiveness of the supplier code is enhanced (a good thing).

A good many of you probably use the `assert()` macro to test preconditions. Testing for postconditions is less common, but equally as important in some circumstances. As an example, consider a set of functions that manipulate a stack of strings. This set must minimally include `push()` and `pop()` functions, as well as a counter that keep track of the number of strings currently on the stack. As shown below, a valid postcondition for the `pop()` function is that the number of strings on the stack is always greater than or equal to zero:

```
char *pop(stack_t *stack)
{
    /* body of function */
    POST(stack->stackSize >= 0);
    return (string);
}
```

In C, preconditions and postconditions apply to individual functions. Invariants characterize a class, or set of cohesive functions, as a whole. In the previous example, the postcondition is better characterized as an invariant for the set of functions that manipulate the stack. Let's rework the `pop()` function to use invariants via the `INV()` macro:

```
char *pop(stack_t *stack)
{
    INV(stack->stackSize >= 0);
```

```
/* body of function */
INV(stack->stackSize >= 0);
return (string);
}
```

Because invariant conditions imply that testing must be performed at function entry and function exit, they

are a bit messier than preconditions and postconditions. Maintaining invariants can also be a maintenance problem. Consider a set of 10 functions that must satisfy four invariants. The code would include a minimum of 80 `INV()` macros. Each invariant adds at least 20 `INV()` macros, at the

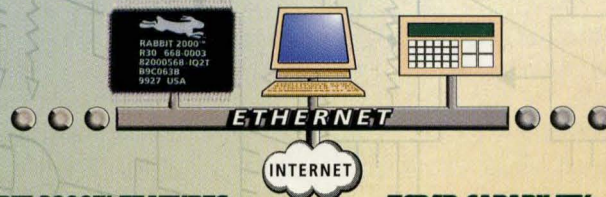
Ethernet & Internet Connectivity

with the

RABBIT 2000™

8 BIT MICROPROCESSOR NOW SUPPORTS TCP/IP

Build your networking application around the high-performance Rabbit 2000™ microprocessor. Full TCP/IP source code is provided. No run time royalties. The industry-proven Dynamic C® software system simplifies development efforts and costs.



RABBIT 2000™ FEATURES:

- Glueless interfacing
- 1 megabyte of code space
- 4 serial ports, 40 parallel I/O
- 7 timers, battery backed timeldate clock

TCP/IP CAPABILITY:

- Transfer data via the Internet
- Transfer files between network nodes (FTP)
- Control remote systems using a web browser via HTTP
- Support provided for TFTP, CGI, SSI protocols
- Send e-mail (SMTP)



Rabbit 2000 TCP/IP Development Kit

Includes: TCP/IP development board (with Rabbit 2000 microprocessor, flash, SRAM, Ethernet hardware, 8 digital I/O), complete Dynamic C® software development system with TCP/IP on CD ROM (not a trial version!), power supply and PC serial cable for real-time debugging.

RABBIT
Semiconductor

For more information or to order, visit

www.rabbitsemiconductor.com

2932 Spafford Street, Davis, CA 95616 • Tel 530.757.8400 • Fax 530.757.8402

entry and exit points for all 10 functions. This is fairly cumbersome and certainly introduces the potential for coding errors.

A reasonable alternative is to define one macro that contains all invariants and use that new macro at all function entry and exit points. This greatly lessens the maintenance issues with the code. If an invariant is added or removed from a set of functions, or if an invariant condition changes, such as a comparison against one instead of against zero, only one macro needs to be modified. In our stack example, the class invariant macro may be as simple as the following:

```
#define STACK_INV() { \
    INV(stack->stackSize >= 0); }
```

Let's rework the `pop()` function once again to use the `STACK_INV()` macro:

```
char *pop(stack_t *stack)
{
    STACK_INV();
    /* body of function */
    STACK_INV();
    return (string);
}
```

Note that the programmer must still have the discipline to insert the class invariant macro at all function entry and exit points. If the tenets of structured programming are followed, namely that functions should have only one exit point, this task is trivial. If a function is designed to have multiple exit points, this task is more laborious and prone to error.

All of this talk about the C programming language may be out of fashion today now that an increasing number of developers are turning to C++. And you are probably wondering if these techniques can be extended to C++ objects. In short, they can be applied to C++ objects up to a point. That point is reached when inheritance comes into play. Design by contract has specific requirements about preconditions and postconditions in a

class inheritance hierarchy. A subclass may keep or weaken the preconditions of an overridden method and it may keep or strengthen the postconditions of an overridden method. Look at the Eiffel Web site for more information (<http://www.eiffel.com>). This type of assertion checking is difficult to perform without assistance from the compiler.

With this in mind, you may want to limit the implementation of the DBC techniques to C++ classes that are either:

- Not derived from any other classes
- Are derived from classes that do not implement DBC semantics

(For those of you that are wondering, Java *does not* support design by contract, although a number of tool vendors—some freeware—offer products with some level of support.)

Put it in writing

Design by contract is a contract between a piece of client software (a client object) and a piece of supplier software (a supplier object). This implies that both the client and the supplier know what the contract is and agree to it. Some preconditions and postconditions the client need never know about and probably does not care about. On the other hand, it is absolutely essential that the client know about other types of preconditions and postconditions. For example, the client software does not want to pass a NULL pointer to a piece of supplier software that forbids it.

How is this contract communicated? As with all great contracts, it has to be in writing. The first step is to include the preconditions, postconditions, and invariants in a standard function header that precedes the function. This allows the documentation of the function to have a decent shot of being updated after someone changes the source code (and someone *will* change the source code).

However, the developers may not always have direct access to the source code, but generally do have access to the header files. Therefore, the developers may not be able to view the preconditions and their cousins.

A generally effective approach to this problem is to include these function descriptions in the header file that contains the function's prototype. This practice ensures that the maintainer of the code has at least some regard for the clients of the code. For those of you with extra time on your hands, a script file can be used to "pull" the function header comment and the function prototype out of the source code file and populate the header file. Scripts facilitate keeping the header file current with the changes to the source code. If a help file or man page exists for the supplier software, or even formal documentation, the preconditions and such should be included right beside the description of the functions' parameters and return value.

Here is a sample function header for the `dupstr()` routine.

```
/*
 * Name: dupstr()
 * Description: This function
 * pushes a string onto the
 * stack.
 * Parameters: string - the
 * string to duplicate
 * numberOfDups - number of
 * times to duplicate
 * Return Value: a new string;
 * the user must free it when
 * done
 * Preconditions: string != NULL
 * numberOfDups >= 1
 * Postconditions: none
 * Invariants: none
 */
```

Conclusion

As with so many of the powerful ideas in the computer science field, design by contract is nothing more than startling common sense. Client software



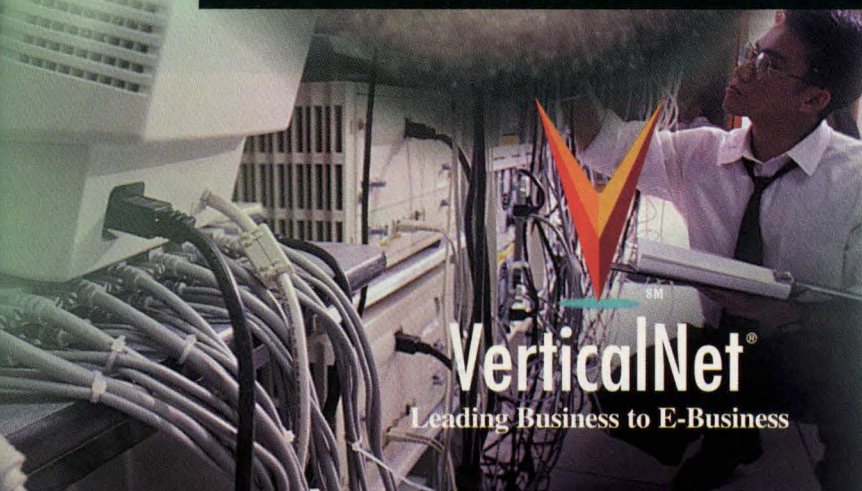
**everything you need to know about
embedded computer design...online**

Showcase your company
for 12 full months with
an online Storefront...
compliments of
Microsoft and VerticalNet!
See site for details!

embeddedtechnology.com gives people
in embedded computer design everything they need:

- Instant buying access to thousands of vendors
and suppliers the world over
- Unlimited sales opportunities through
customized Storefronts, E-Commerce
Centers and other services
- Links to industry auctions, to help you buy
and sell equipment/materials at great prices

www.embeddedtechnology.com

- 
- Breaking news—including regulatory info—
and the latest technology
 - Hot job postings and career opportunities

All delivered on time, on target, and totally FREE.
Become a part of the online community that's 100%
focused on the embedded technology industry.
And while you're there, check out our
sales-building, no-cost Storefront solution.

VerticalNet[®]
Leading Business to E-Business

should understand what service the supplier software is obligated to fulfill, as well as the constraints under which the client software must operate. Conversely, supplier software should understand what services it must provide. Often, just the act of documenting the preconditions, postconditions, and invariants is enough to raise the overall understanding of the roles of supplier and client objects.

The framework presented here is a flexible and powerful implementation of the design by contract techniques for the C language. The assertion mechanism may be independently turned on or off at run-time. Alternately, the assertion mechanisms may be independently included in the source code or excluded from the source code through conditional compilation. Finally, the framework is easily customizable to handle asser-

tion failures in a system-dependent manner.

If you decide to use DBC techniques on a current or future project, you'll be walking down the yellow brick road towards the Emerald City of improved software productivity. Your supplier code will be better tested and more reliable, and the clients of your code will have an improved understanding of their responsibilities (as well as the responsibilities of the supplier code). And for the *coup de grace*, you won't be searching through endless modules of code looking for where the bug occurred. You'll know the file and line number where the error occurred. If that isn't an increase in productivity, then what is? **esp**

Steve Kapp is a consultant and site manager for Embedded Real-Time. He has 12 years of software development experience, mainly concentrated on embedded imaging

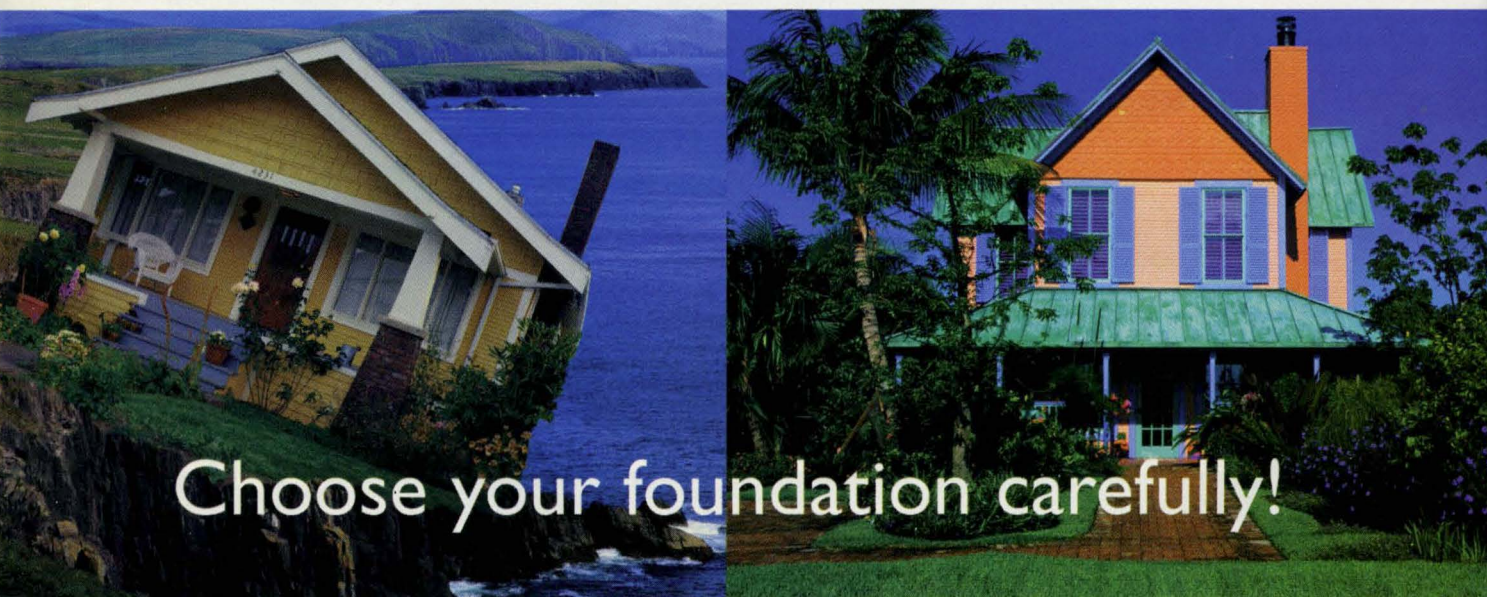
systems. He is currently developing networking software for office automation equipment. Steve can be reached at skapp@emrt.com.

References

1. In fact, "Design by Contract" is a trademark of Interactive Software Engineering, a company founded by Meyer to market the Eiffel programming language.

Resources

- "Building bug-free O-O software: An introduction to Design by Contract," <http://eiffel.com/doc/manuals/technology/contract/index.html>.
- "Explanation of Design by Contract," <http://homepages.munich.neturf.de/Joachim.Durchholz/eiffel-critique/dbc.html>
- Rangaraajan, K., "Does Java Support Design by Contract?" *Dr. Dobb's Journal*, November 1999, p. 113.



Choose your foundation carefully!

Start your DSP product development on a solid foundation with

Blackhawk

by EWA Inc.

DSP Enabling Technologies™



For more information please visit: www.blackhawk-dsp.com or phone: 1-877-983-4514

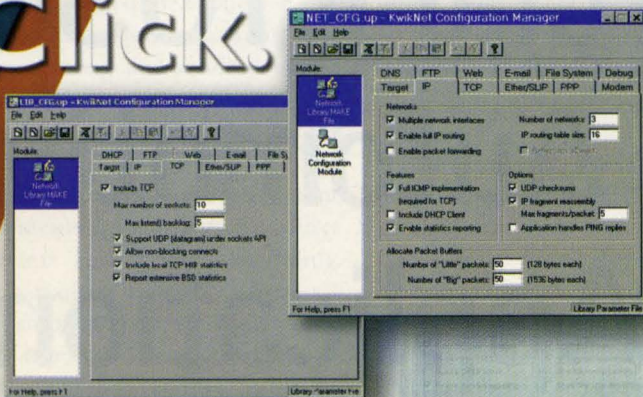
© 2000 EWA

KwikNet™

Adding network features to your embedded products?

KwikNet makes it quicker, easier, and less expensive.

Just
Point and
Click.



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

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

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

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

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

We've already tested it.

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

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

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

KwikNet TCP/IP Stack and Web Server

 **KADAK** RTOS and Network Products

KwikNet and AMX are trademarks of KADAK Products Ltd.

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



Commercial RTOSes for Automotive Applications

Automotive powertrain applications have tough real-time scheduling requirements. Here's an approach for evaluating commercial RTOSes for such applications.

Automotive powertrain applications require stringent real-time scheduling services that present significant challenges for mainstream real-time operating systems (RTOSes). The greatest challenge to the RTOS is providing the required services within a reasonable resource budget (CPU, RAM, and ROM) that is affordable in the resource-constrained automotive applications domain. In order to evaluate the RTOS, determining the real-time processing requirements of the

application and how the RTOS will satisfy the requirements is necessary. A resource-modeling approach permits scaleable evaluations. Fundamental RTOS measurements taken from the application perspective can be combined in a model to provide total RTOS overhead, which can be used for evaluation purposes. Ford Research Laboratories has conducted an RTOS evaluation including 10 RTOSes for the Motorola 683xx and seven RTOSes for the Motorola MPC5xx. Resource models have been developed. This article describes the evaluation approach and the results.

Here we present a general description of automotive powertrain real-time requirements and the corresponding RTOS requirements. Some of the current microprocessors in use are described, with emphasis on the resource constraints. The primary evaluation issues of CPU processing overhead and RAM are described along with the testing methodologies employed. We also define CPU resource overhead models and present the evaluation results. We focus on standard commercial RTOSes that support preemptive multitasking and blocking.

Automotive powertrain application

An automotive powertrain application has a combination of real-time operations that have hard, moderately critical, and soft deadlines. Many of the hard deadlines are handled with "smart" on- and off-chip peripherals. The TPU on the M68332 is an example of a smart peripheral that can handle the hard deadlines. Some hard deadlines and remaining real-time processing are handled by "C" functions whose execution is coordinated within tasks scheduled by the RTOS.

Automotive application real-time requirements are sensitive to resource constraints and proper scheduling. Minor variations in latency can be tolerated since most of the highly time-critical input and output operations are independently handled by the on-chip peripherals, thus being immune to minor latency delays. Of course, significant latency due to scheduling issues is unacceptable.

The RTOS services required for automotive powertrain applications are rather limited compared to other more complex embedded systems. Automotive powertrain applications do not usually require file I/O, distributed processing, dynamic memory allocation, time slicing, network management or user interfaces. The need for event flags, semaphores, mailboxes, and queues is limited.

An automotive powertrain application has real-time processing requirements that will stress the resources available on the microprocessor/microcontroller. The requirements are due to the large number of external interrupts that cause tasks to run. Most of the interrupts are proportional to the engine crankshaft revolution rate or revolutions per minute (RPM).

The task processing associated with the interrupt is significant and can't be handled within an ISR. A second source of real-time stressing is periodic tasks. Many of the core algorithms are scheduled to run periodically at rates varying from every four milliseconds to every one second. About 95% of all RTOS CPU overhead is due to periodic and asynchronous task handling. Additional overhead can be attributed to semaphores, the tick timer, and miscellaneous RTOS services. A typical application has about 10 tasks.

The external interrupts that are proportional to the RPM are the primary application events. The events are associated with specific crankshaft angular positions, which are critical to the algorithms in the associated task. In order to meet the deadlines, a kernel must be very efficient at triggering a task from the ISR. These events are proportional to the RPM and become the predominant source of CPU processing time at high RPMs. Depending on the type of engine, a powertrain controller may have an operating region around zero to 1,000 interrupts per second with a worst case sustained operation at up to 2,000 interrupts per second on larger applications.

The number of interrupts can be approximated with the following equation:

$$N_{isr} = \left((N_{isr} * (N_{cyl} + 1 + N_{rm})) + 60 \right) + N_{comm} + N_m \quad (1)$$

where:

- N_{isr} is the total number of external interrupts resulting in task processing per second
- N_{rpm} is the crank shaft revolution rate

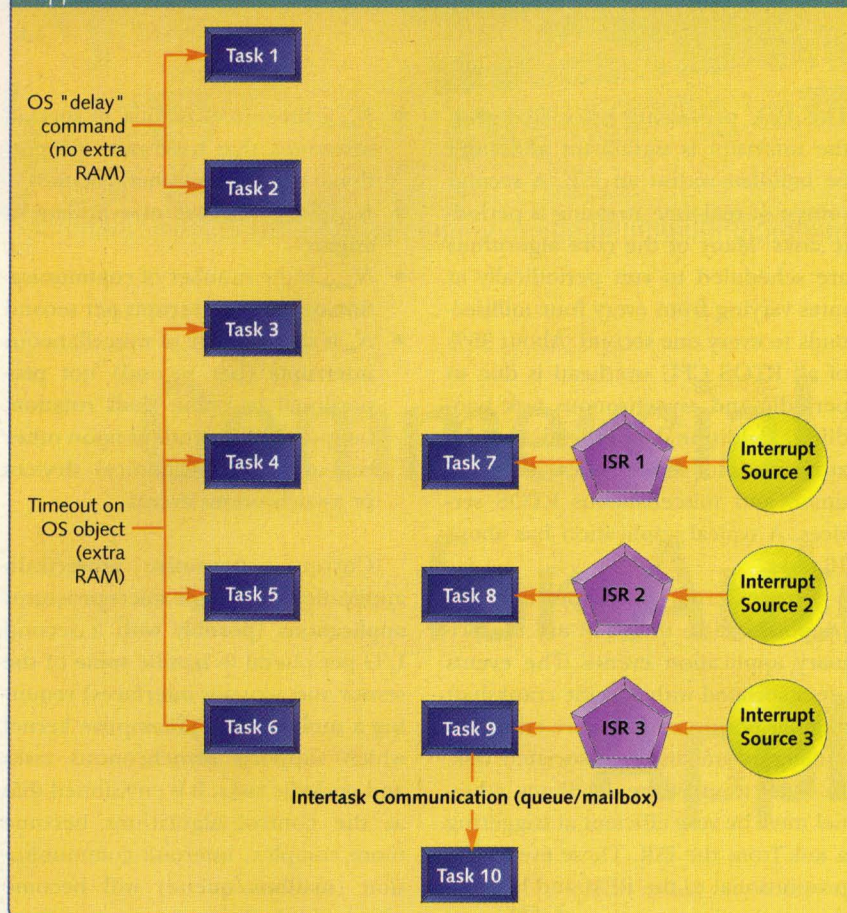
- N_{rm} is the miscellaneous, or special, interrupts that have rates proportional to the crank shaft rotation
- N_{cyl} is the number of cylinders in engine
- N_{comm} is the number of communication-oriented interrupts per second
- N_m is the number of miscellaneous interrupts (per second) not proportional to crank shaft rotation, but possibly dependent upon other rate-oriented mechanical devices or asynchronous events

Current automotive powertrain applications are single microprocessor applications (possibly with a second I/O peripheral to handle some of the sensor and actuator interfaces) requiring a multitasking, preemptive kernel which supports asynchronous tasks and periodic tasks. It is envisioned that as the control algorithms become more complex, intertask communication (mailbox/queue) will become more useful. Currently, an occasional need for resource locking arises.

Example architecture.

Figure 1 illustrates a typical automotive powertrain tasking architecture. This particular example shows three interrupt sources, each of which triggers a separate task to run. These tasks are usually high priority but moderate duration tasks. This architecture also has six periodic tasks. Most kernels have many different mechanisms for causing a periodic task. This example shows two of these methods. The first method involves a "delay" command which is typically the fastest mechanism to accomplish a periodic task. Also, this does not require any extra RAM. A second method, illustrated here, is a timeout on a kernel object such as a mailbox, event flag, or semaphore.

FIGURE 1 Example OS architecture for an automotive powertrain application



Typical automotive powertrain microprocessors and microcontrollers

A typical powertrain control application can be supported with any of several different microprocessors. The two microprocessors considered for this investigation are the Motorola 683xx microcontroller and the Motorola PowerPC 5xx microprocessor.

The 683xx is a non-floating-point microcontroller. It has 2K to 7K of on-chip RAM and 256K of on-chip ROM. The microprocessor runs at 16MHz. The PowerPC 5xx is a floating-point microprocessor with 24K to 32K of on-chip RAM and 256K to 512K of on-chip ROM. The microprocessor can run up to 28MHz.

Because of the cost constraints associated with the automotive industry, adding more resources is not

practical unless absolutely necessary. Based on this, a budget of approximately 1K of RAM is given to the kernel for the 683xx. With the extra floating-point registers and the extra on-chip memory, the RAM allocation for the PowerPC is approximately 5K to 6K.

Evaluation criteria

As we've briefly addressed already, the two major items of this evaluation are RAM and CPU timing overhead. The RAM is a very limited and expensive resource. The CPU timing overhead can significantly impact the overall performance if the kernel uses too much.

Some other evaluation criteria include ROM, kernel features (mailboxes, queues, semaphore, resources,

and so on), ease of use, documentation, support tools, track record of kernel and company and cost. Our primary focus here is resource usage.

Evaluation methodology

RAM measurement methodology. Several possible methods are available to measure the RAM that an operating system uses. These methods include both static and dynamic approaches. For the static approaches, assembly listings or the memory map can be reviewed. Dynamic approaches include tracing the address lines and filling the memory with a special character before running the OS. This evaluation uses a combination of the static and dynamic methods. RAM overhead equations that are a function of the number of kernel objects (tasks, mailboxes, and so on) and kernel configuration can be formulated using the data collected.

CPU processing overhead measurement methodology. Measurement methods are based on external observation from an independent viewpoint. This is accomplished with instrumentation of the software at critical points of interest to set/clear discrete external outputs that can be observed with a logic analyzer. Critical points of interest can be externally observed as signal transitions. Rising and falling edges are used to indicate specific points in the application execution. Continuous toggling of a discrete external output is used to indicate whether or not a task is running. The logic analyzer is programmed to detect edge transitions and calculate time differences. This reduces the level of operator interaction and error. As with instrumentation or measurement of any system, the setting/clearing of discrete external outputs does result in increased overhead in the test. The instrumentation has been introduced as uniformly as possible to each RTOS test suite to minimize variations in the overall analysis.

✓ Royalty Free

✓ Comprehensive
Product line

✓ Focus on Service



All You NEED in an RTOS

www.atinucleus.com



There is
only one conclusion.

Only one RTOS provides all the benefits you need. Accelerated Technology combines a level of service that is unmatched, an affordable pricing model and open source benefits with a vast, tightly integrated embedded product line offering.

Nucleus MNT - Windows-based rapid prototyping environment

Nucleus EDE - intuitive embedded development environment based on Microsoft Developer Studio™

Nucleus PLUS - robust, scalable, multitasking real-time kernel

Nucleus NET - complete TCP/IP networking protocol stack

Nucleus UDB - portable source level debugger

SurroundView and Nucleus ProView - revolutionary profiling tools that introduce a new level of application and OS monitoring

Nucleus WebServ - a tightly integrated, internet-enabling embedded web server

Nucleus WebBrowse - a compact and tightly integrated embedded web browser

Nucleus GRAFIX - portable embedded graphical user interface

These Nucleus embedded products support a wide range of processors:

- MCF5206, 5307 • M-CORE
- PPC40x, 5xx, 60x, 7xx, 8xx, 82xx
- 680x0, 683xx
- ARM6/7/9, AEB, Atmel 40400, CL7110, 7111, 7209, Samsung SNDS100
- SA100-285, SA1100, SA1110 • ARC
- SH1, SH2, SH3, SH4, SH3-DSP, H8S/2000, H8/300H
- IDT RC3081, RC4640/50, R5000, RC32364
- LSI LR 33000, 40xx, 410x, 64008, Lexra 4180, NEC 41xx, 4300, 5000, NKK 4650, Toshiba TX3904, 3927
- x86 RM/PM • TriCore • C167

For additional information on the complete Nucleus product line and how it can greatly diminish your time-to-market needs, royalty-free, please contact us at:

Phone: 1.800.468.6853

Address: 720 Oak Circle Dr. E. Mobile, AL 36609

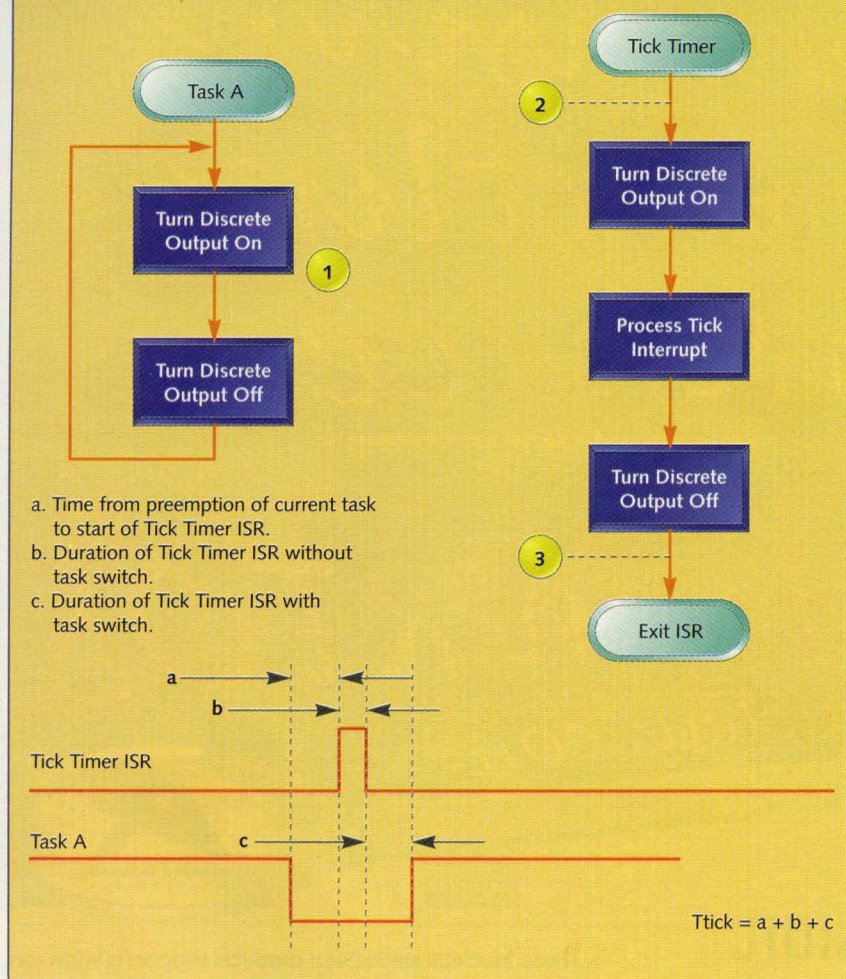
Email: info@atinucleus.com

Internet: www.atinucleus.com



All You Need in an RTOS. Royalty Free.

FIGURE 2 Tick timer RTOS CPU overhead test



Other approaches are available for time measurement of operating systems. It's possible to simply set break points at the same critical points of interest and observe time from an on-board timer chip or clock. However, this approach is manually time consuming and does not easily permit a total overview of all of the interactions in a time sequence representation. The logic analyzer permits several events to be simultaneously tracked to observe relative sequencing of events as well as individual time measurements. An alternative approach to time measurements of operating systems is to establish RTOS interaction loops that execute numerous times and calculate an average time from

the on-board timer at the end. This approach is good for averaging, but does not help to investigate relative sequencing of events or abnormal operations. The logic analyzer can be programmed to support averaging and statistical distributions. Abnormal conditions can be triggered as needed.

RTOS RAM sources

On an operating system, several entities need to use RAM, including the kernel objects like the task control blocks (TCB), semaphores, event flags, queues, mailboxes, memory partitions, timers, and other miscellaneous entities that are vendor specific. Another important source of RAM draw depends on the stack structure

that the kernel uses. A final source that requires RAM are the registers that need to be saved during a context switch.

The RAM for the kernel objects is usually straightforward. Each entity is usually a fixed size (except for variable-length messages), and contains only information that the kernel uses. Except for the contents of messages, this RAM usage is abstracted away from the application programmer.

The RAM for stacks is a much more complex source of RAM than the kernel objects. The stacks contain application information, along with kernel information, and in many cases saved registers which contain application information.

Generally, three types of stacks are used by kernels. These stacks include the task stack, the kernel (or system) stack, and the ISR stack. Most kernels use some or all of these stacks, and some kernels have slight variations on these that lead to additional stacks. For example, one kernel has a stack for each ISR instead of one common stack for all the ISRs to share. This makes it easier to tune each stack, but it does not change the overall ISR stack usage. Conversely, a kernel that only has task stacks will require considerably more RAM for the stacks than a kernel which has both task stacks and an ISR stack. The reason is that when no ISR stack is present, the task stack of every task has to be increased to handle the worst case nesting of interrupts that can occur. Whereas, for the kernels with an ISR stack, allocating enough RAM for this worst case nesting occurs only once instead of for each individual task stack.

The registers saved on a context switch also require some RAM. These registers are usually saved on the task stack, but for some kernels they are saved in the TCB. It should be noted that when calculating the RAM for the worst case nesting, these registers also need to be saved for each nesting.

THREAD **X**

FREE
Evacuation Kit™
for pSOS users
Contact Us
Today

**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

Because Being the Fastest Is Not Enough.

Yes, it's true! ThreadX is the fastest RTOS on the market today. This simple fact can make your application more responsive and better able to handle extreme real-time processing demands. In short, it gives you a more comfortable development environment.

But ThreadX is more than just fast. It is the leading royalty-free, source-code RTOS on the market today, with a host of technical advantages over the competition. Its unique preemption-threshold™ technology has been shown by academic research to improve real-time performance by marrying the best of preemptive and non-preemptive scheduling techniques. ThreadX also has a picokernel design that automatically scales to your needs.

With all these features, plus comprehensive processor support and advanced ThreadX-aware debugger integration, ThreadX can instantly start speeding your application to market.

Call 888.THREADX or visit us at www.threadx.com and find out more about ThreadX, the RTOS that puts you comfortably in the fast lane!



eL
Express Logic, Inc.

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

CPU processing overhead tests

With a kernel, many actions can be timed, such as individual function calls, task switch times between various combinations of tasks and from various kernel actions, interrupt latency times, and so on. These tests were developed and implemented. However, this report will concentrate on three core tests which include measuring the overhead for a tick, for a periodic task, and to trigger a task from an ISR.

Tick timing overhead test. The Tick Timer CPU Overhead Test (see Figure 2) determines the amount of CPU overhead incurred each time the Tick Timer ISR runs without any blocked tasks with timeouts. This measurement is important because it represents a fundamental reoccurring overhead that will always be present in an application that uses an RTOS with time-

out/time management support. Additional tick timer-related overhead can only be incurred when there are blocked tasks with timeouts or when timeslicing is enabled. Additional tests have been specifically established to determine when and how much additional overhead will be incurred.

Overhead analysis. The CPU overhead incurred each time the Tick ISR executes is as follows:

$$T_{\text{tick}} = t_{t_a} + t_{t_b} + t_{t_c} \quad (2)$$

The equation terms (subscript names t_{t_a} , t_{t_b} , and t_{t_c}) correspond to the a, b, and c terms, respectively, in Figure 2. The equation holds regardless of what happens within the ISR. Some RTOSes will invoke the scheduler within the ISR and others will invoke the scheduler upon exit of the ISR. The key point is that the resource

equation is based upon the net impact to the application. If one RTOS increases ISR overhead, a corresponding reduction in other overhead equations (periodics) will occur. However, the magnitudes may not be the same. The resource models will permit the total net effect per a given tasking architecture and stressing model to be evaluated. This approach permits different RTOS implementations and features to be evaluated in a uniform manner.

Periodic timing overhead test. The Periodic Task OS Overhead Test (see Figure 3) determines the amount of CPU overhead that is incurred due to a periodic task and determines the overhead that is in addition to the normal tick timer overhead. When analyzing the data, using the proper weights for each overhead parameter is necessary. The tick timer weight is fixed. The periodic task weight can vary based on the application usage. The two sources of CPU overhead are separated to permit comparative evaluation for multiple application configurations. In other words, it's possible to use the same evaluation data for different Tick Timer rates and different periodic task rates. The combined analysis is described in section "Total RTOS CPU Overhead Analysis."

The periodic task OS overhead test has two major components. The first is due to overhead incurred by the kernel when processing each tick timer service update and a timeout has not occurred. The overhead is the delta increase in the tick timer processing as observed by the current preempted task. The actual delta overhead may occur during the tick timer service update or during the kernel scheduler update. However, it doesn't matter which is the source because the delay and overhead appear the same to the application. The kernel does not reschedule any tasks. The ready list is not altered. The only thing that has actually occurred is that the task that is



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-Free800-348-8051
Phone972-312-1107
FAX.....972-312-1159

Microware customers have more than 250,000
deployed JAVA™ units running on OS-9® today.

We can help you do the same.



Get to market first™

If you're about to jump into the market with a new product, make sure you've got a Java-branded embedded solution that goes the distance. Our customer success rate is more than double the industry average – because our customers use our powerful development tools, proven software components like the OS-9 RTOS, and experienced consultants who help them succeed. Microware's Java implementation is tightly integrated with graphics, networking, plug-in support, non-intrusive garbage collection, memory protection, and development tools for Internet solutions.

Microware's record of success in helping OEMs get their designs to market quickly is legendary. Microware solutions power successful designs for world leaders like Hitachi, Intel, ARM Ltd., MIPS Technologies, Motorola, STMicroelectronics, and Toshiba. That's why embedded systems developers on six continents have turned to Microware for more than 20 years.

Call 888-642-7609 today or visit our web site
at www.microware.com.



blocking on the timeout has had its timeout count decremented. The second component is due to overhead incurred by the kernel when processing the tick timer service update when a timeout does occur. The overhead occurs due to moving the blocked tasks that has just timed out to the ready list. Instead of resuming the previous running tasks, the newly timed-out highest priority task is started.

Overhead analysis. The delta increase in CPU overhead incurred by a single periodic task using a service call with timeout is as follows:

$$T_{pd} = t_{pd_f} + t_{pd_a} + t_{pd_c} + t_{pd_d} - T_{tick} + (t_{pd_a} + t_{pd_b} + t_{pd_g} - T_{tick}) * (n_{delay} - 1) \quad (3)$$

The equation has been prepared for the delay kernel call. The form of the equation is the same if a mailbox, event flag, semaphore, or queue was

used, provided each supports timeouts.

The equation has two major components. The first component is the overhead associated with the actual starting of the periodic task. It only occurs when the timeout has expired and the task has been started. The second component is the delta overhead associated with updating the timeout counters each time the tick timer executes, but does not result in a timeout.

The first component is made up of the following:

$$t_{pd_f} + t_{pd_a} + t_{pd_c} + t_{pd_d} - T_{tick} \quad (4)$$

The periodic task starting overhead is composed of the following:

- t_{pd_a} is the time to leave the currently running task in response to the Tick Timer interrupt

- t_{pd_c} is the time inside of the Tick Timer ISR
- t_{pd_d} is the time to move the periodic task from the blocking state to the ready list and to then start the scheduler to actually start the task
- t_{pd_f} is the time to return to the pre-empted task upon completion of the periodic task

However, since the normally occurring Tick Timer overhead is already accounted for in the "Tick ISR" equation, it is necessary to subtract off the normally occurring tick timer overhead (T_{tick}). This permits the results to reflect the delta increase in overhead due to the periodic task starting. It is important to properly account for the overhead since the periodic task only occurs at specific times and the tick timer occurs on a more frequent, regular basis.

The second reoccurring component is made up of the following:

$$(t_{pd_a} + t_{pd_b} + t_{pd_g} - T_{tick}) * (n_{delay} - 1) \quad (5)$$

The reoccurring periodic timeout counter update overhead is composed of the following:

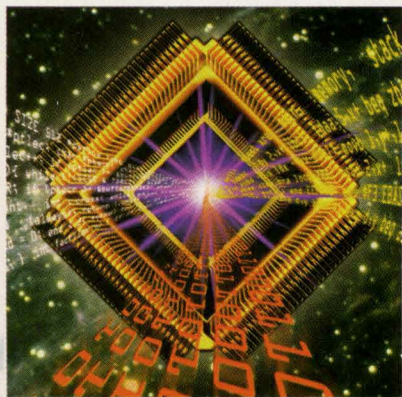
- t_{pd_a} is the time to leave the currently running task in response to the Tick Timer interrupt
- t_{pd_b} is the time inside of the Tick Timer ISR
- t_{pd_g} is the time to return to the pre-empted task upon completion of the Tick Timer ISR

The same "Tick Timer" overhead situation arises and requires the normally occurring tick timer overhead (T_{tick}) to be subtracted.

The second component can occur multiple times if the timeout does not expire. It is therefore necessary to factor in the total number of ticks that will occur that do not result in a timeout ($n_{delay} - 1$).

Asynchronous task timing overhead test. The asynchronous task is fundamental

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



You were right.



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

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 micro-controllers. 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.

to most control systems. It permits external events to be detected via interrupts and processed via tasks. It provides the reactive real-time behavior necessary for a control system.

The asynchronous task OS overhead is easily calculated by making four fundamental measurements. The overhead model is simpler than the periodic task overhead model, and is based on the time it takes to preempt an existing task to service the ISR, invoke the asynchronous task, save the context of the original task, and then restore the context of the original task after the asynchronous task is completed. The time inside of the asynchronous task is not included.

Overhead analysis. The delta increase in CPU overhead incurred by a single asynchronous task processing due to an external trigger is as follows:

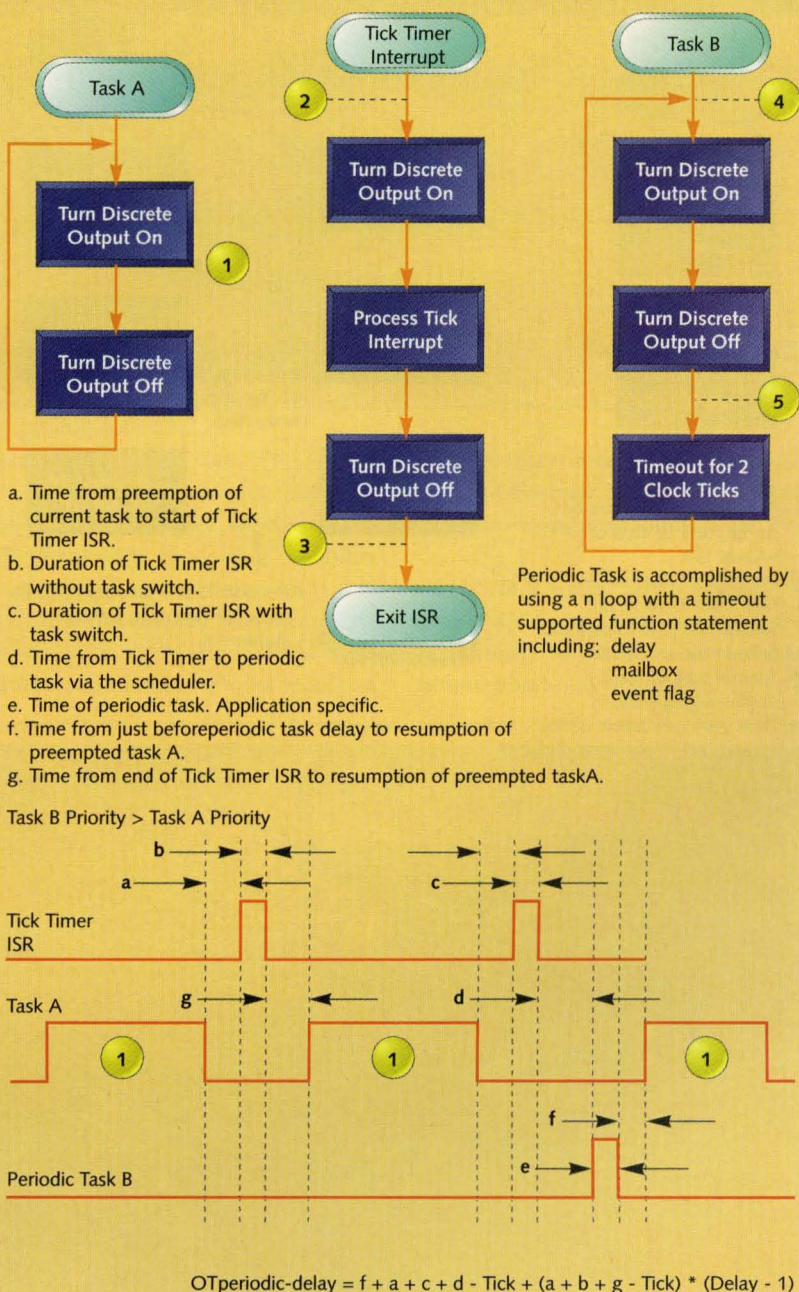
$$T_{aei} = t_{aei_a} + t_{aei_b} + t_{aei_c} + t_{aei_e}$$

The equation has been prepared for a semaphore. The form of the equation is the same if a mailbox, event flag, or queue is used, provided each supports blocking.

Total RTOS CPU overhead analysis. For a given tasking architecture it is possible to formulate a set of first order equations that provide an accurate model of the CPU overhead required. The overhead is evaluated by combining the tick timer, periodic, and asynchronous event RTOS overhead components. Each resource usage component is weighted by the rate of occurrence. The tick timer occurs at a fixed rate (N_{tick}). The periodic tasks will occur a specific number of times per second (N_{pts}) for any particular tasking architecture. The asynchronous events will usually occur at varying rates. However, for any particular situation it is possible to establish the number of events per second (N_{isr}).

It should be noted that the Tick overhead appears in the periodic and tick ISR equations. It should also be

FIGURE 3 Periodic task RTOS overhead using timeout test



noted that it is subtracted from the periodic case to avoid double counting of its effects. The total RTOS CPU overhead is provided in Equation 7:

$$O_{os} = N_{tick} * T_{tick} + N_{pts} * T_{pd} + N_{isr} * T_{aei} \quad (7)$$

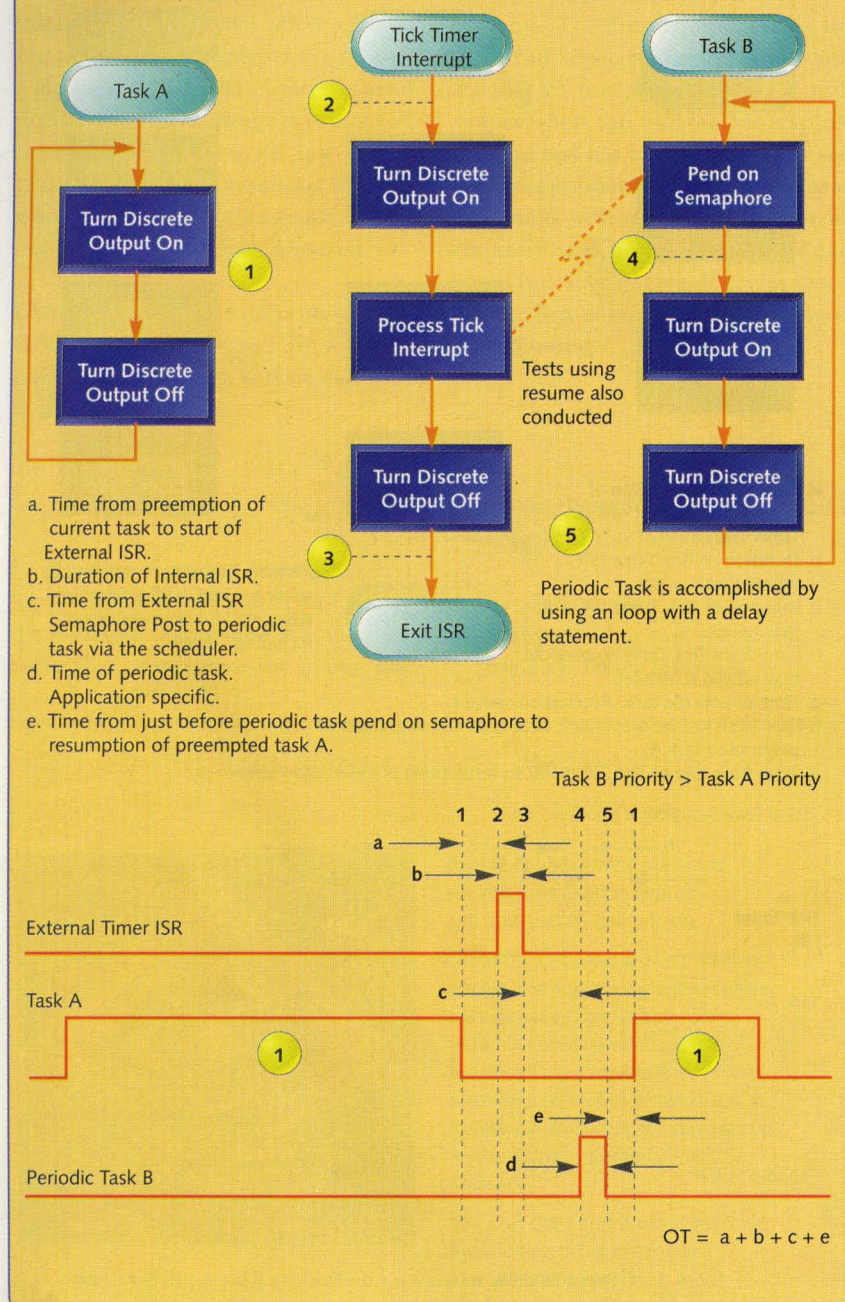
Kernels evaluated

Table 1 shows an evaluation of the kernels from various companies.

Test setup

The test setup includes a PC running the SDS debugger with a BDM connector to the evaluation board. The

FIGURE 4 OS overhead due to asynchronous task



M68332EVS Evaluation System with a 68332 daughter card was used for the 68332 part of the evaluation. For the PowerPC part of the evaluation, the MPC505EVB was used. A Tektronix 3001 GPX logic analyzer with 5ns resolution was used to measure the signal transition times. A Tektronix AWG2005 Arbitrary Waveform Generator was

used to provided an accurate source of interrupts. For the applications, the same compiler and compiler options were used wherever possible.

Hardware

For each microprocessor, the hardware configuration was kept identical for each kernel.

For the 68332, the clock was at 16MHz, and the memory was configured as fast termination with zero wait states.

For the PowerPC 505, the clock was configured at 16MHz, the memory was set to zero wait states, and the "show cycles" were disabled (not documented in the normal Motorola manuals). The following cache configurations were used:

- Cache turned on and allowed to run freely
- Cache turned off
- Cached turned on and locked with non-kernel and non-test code

This last cache configuration is probably most representative of what would happen in a real application, as the kernel will most likely not be in the cache. The application code executed between kernel activity will usually be large enough to displace the kernel code from the cache. The tests were run with both the floating point enabled and disabled. The non-floating-point configuration was performed to show the CPU timing and RAM savings that could be accomplished by not using the floating-point registers in a given task.

Evaluation results

The following sections present a summary of the results for the evaluation. For the 68332, each kernel will be referred to as OS#. For the PowerPC 505, each kernel will be referred to as Ven#. The companies listed in Table 1 are not associated with these identifiers as per our agreement with each company before the evaluation started. Also, only the data for the three core tests described above will be included.

68332 CPU processing overhead results.

For most of the kernels, the most efficient mechanism to accomplish a periodic task was to use the "delay" command. When supported, the

"resume" command was usually the most efficient way to trigger a task from an ISR.

As a general rule, no single kernel is best in every test. Thus, to properly evaluate each, the intended architecture (Figure 1) must be taken into account. The values in Figure 5 are based on Equation 7 with numerous validation points.

The data graphed in Figure 5 assumes that there are 250 periodic task switches per second with a 4ms tick. The 250 periodic tasks switches can be accomplished with many different configurations, from one task every tick, to many tasks running once every few ticks. This "background" activity is responsible for the Yintercept in the graph. The independent variable is the number of interrupts per second.

As Figure 5 shows, most of the kernels have a similar CPU overhead,

TABLE 1 RTOSeS evaluation

Company	Kernel	332	505
Accelerated Technology	Nucleus	x	
CMX	Tiny+	x	
Enea Data	OSE Classic	x	
JMI	JMI	x	
Microtec Research	VRTXmc	x	
U.S. Software	MTASK	x	
Integrated Systems	pSOSelect	x	
Integrated Systems	pSOS		x
Embedded Power	RTXC	x	x
Embedded Power/Motorola	RTEK	x	x
University of Michigan	Emeralds ⁴	x	x
Etnoteam	EOS		x
Express Logic	ThreadX		x
Precise Software Tech.	MQX		x

except for two (OS7 and OS8), which have significantly more overhead. The best kernel, OS2, uses approximately 15% of the CPU processing capability at an operating condition of 1,000 interrupts per second (established by Equation 1). At a high RPM with numerous additional events resulting

in 2,000 interrupts per second the kernels use between 25% and 55% of the CPU processing capability.

68332 RAM results. Figure 6 shows the kernel RAM when varying the number of periodic tasks. Six other tasks are assumed to be in the system. Three of

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
INTUITIVE USER INTERFACE



HIWARE
Innovation for your Success

Hiware USA

5808 Brown Rock Trail
Austin, TX 78749
USA

Ph. (512) 301 4500
Fax (512) 301 0957
info_us@hiware.com

Hiware

Riehenring 175
4058 Basel
Switzerland

Ph. +41 61 690 7500
Fax +41 61 690 7501
info@hiware.com

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_fr@hiware.com

www.hiware.com

FIGURE 5 CPU overhead for 68332

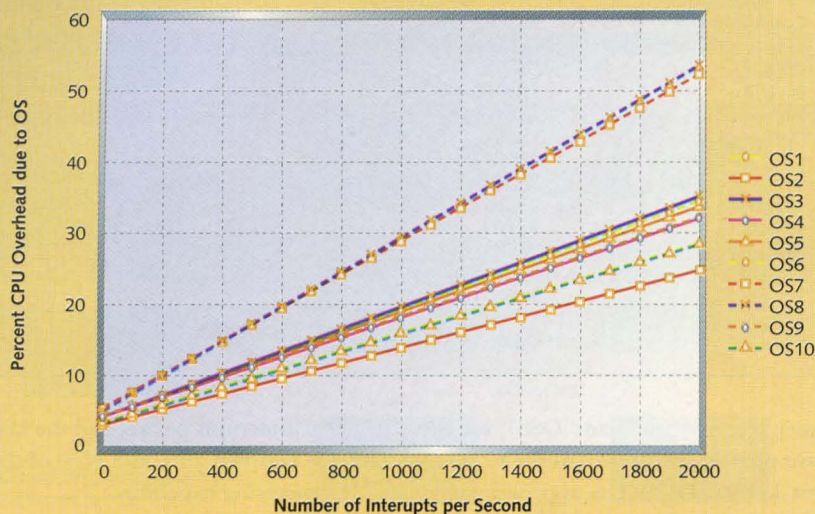
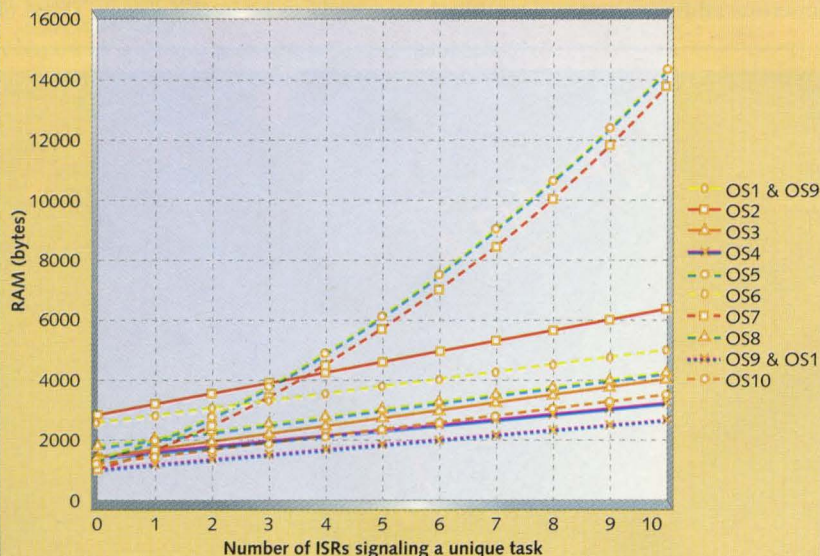


FIGURE 6 RAM for 68332 kernels by varying the number of periodic tasks



these tasks are signaled from an ISR. This data assumes zero task variables for the stack and zero ISR variables for the stack. This means that the application RAM usage is zero and the data is only for the kernel's RAM.

As Figure 6 indicates, there is a large spread on the RAM for each of the kernels. A typical operating scenario with three interrupt sources requires a minimum of approximately 1.5K of RAM

just for the kernel. Since the 6833x has only 7K of RAM, some of these kernels will use that up with a relatively small number of tasks. An interesting point shown here is that while OS2 was the best kernel for CPU performance, it is one of the worst for RAM usage.

Figure 6 has an interesting feature in that some of the curves are not linear. This occurs because the kernels do not have a separate stack for the ISRs. Thus,

each task stack must have room for the worst case nesting of the ISRs. From a RAM usage point of view, having a separate stack for the ISRs is very beneficial.

As shown in Table 2, the kernels require from 1.5K to 3.8K to implement the 10-task example architecture shown in Figure 1. It can be concluded that none of the RTOSes for the 68332 are adequate for the RAM resources available on the 683xx. Therefore, it is necessary to consider simpler RTOSes that use fewer resources at the expense of capability. A couple of RTOSes that do not support blocking appear as potential candidates.

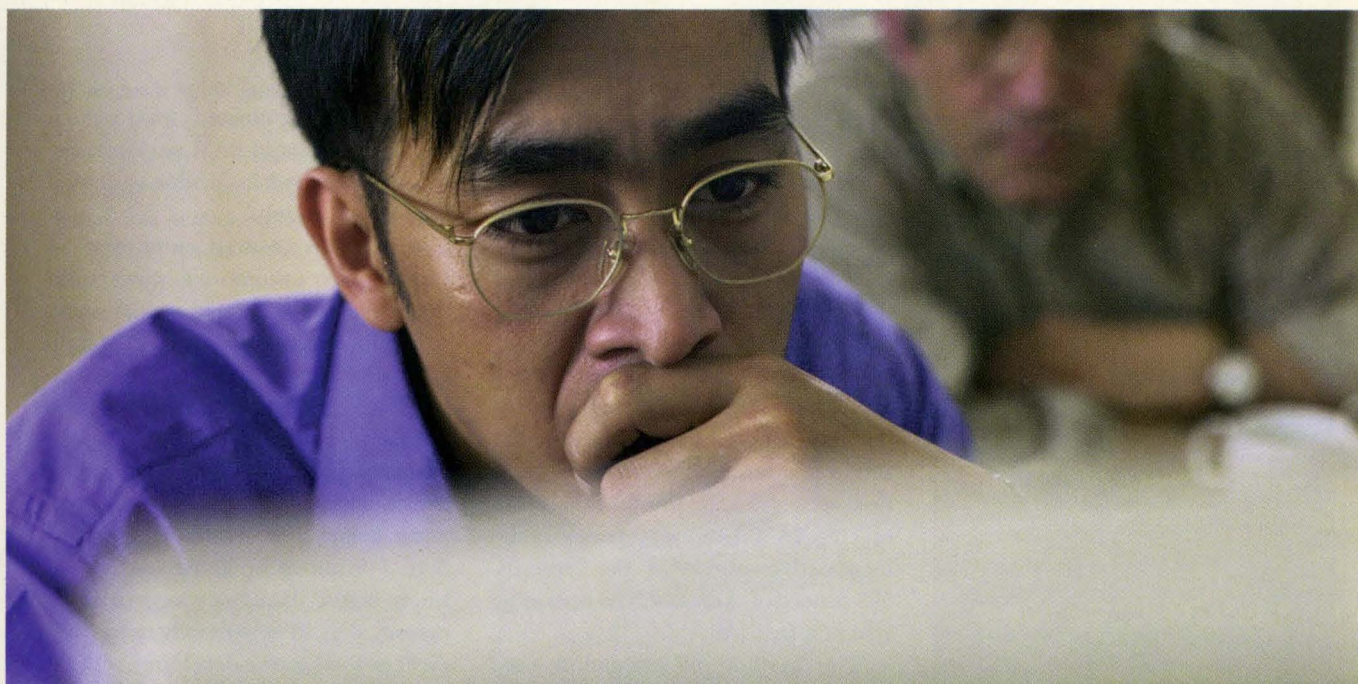
PowerPC 505 processing overhead results. The data in Table 3 present the most efficient cases for periodic and asynchronous task overhead based upon the overhead analysis equations provided in Equations 3 and 6.

As with the 68332, the PowerPC 505 configuration has 250 periodic task switches per second and a 4ms tick interval.

As shown in Figure 7, most of the kernels have very similar CPU performance for this particular application. Ven. 2 has significantly poorer performance while Ven. 7 has slightly better performance. The reason for this closeness is that the PowerPC 505 has a large number of registers. Thus, most of the time is spent saving registers and not much room is left to differentiate each kernel, assuming that each is somewhat efficiently written.

Additional tests for the various cache configurations resulted primarily in a magnitude change of the CPU overhead. As would be expected, the cache-off case uses the most CPU processing, followed by cache-locked. The free-running cache used the least amount of the CPU.

PowerPC 505 RAM results. Figure 8 shows the kernel RAM when varying the number of floating-point periodic tasks. Six other tasks are assumed to be in the system. Three of these tasks are



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!

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



It's all right here.

www.eedesign.com

activated from an ISR. This data assumes zero task variables for the stack and zero ISR variables for the stack. This means that the application RAM usage is zero and the data is then only for the kernel's RAM.

As Figure 8 indicates, there is a large spread on the RAM for each of the kernels. A typical operating scenario with five periodic tasks requires from approximately 6K to 15K of RAM just for the kernel.

Practical solutions

Based on the data in this report, using a standard commercial operating system for the 683xx may not be practical for an automotive powertrain application because of the limited amount of on-chip RAM. However, because of the increased resources of the PowerPC 505 processor, a commercial operating system is feasible.

The method of evaluation used here allowed for some base measure-

ments to be collected from which expected CPU and RAM performance can be predicted for a wide variety of kernel applications.

While not covered extensively in this article, automotive powertrain applications use a very small fraction of the total kernel features. This indicates that a more efficient kernel can be developed for automotive powertrain applications. Some of the kernels evaluated were developed by scaling back a larger, more full-featured operating system and calling it a "micro" kernel. While these kernels were definitely better than their larger predecessors, they were generally not as efficient as the kernels developed with an initial focus on resource constraints.

For the more resource-constrained M6833x, the non-blocking kernel technology appears promising and will receive further scrutiny.

The automotive RTOS standard OSEK does not provide any unique capability to solve the resource constraints issues.^{1,2} However, some promising "low end" implementations of OSEK may address the resource constraint issue in the future.³ Further investigation is underway at Ford Motor Company.

esp

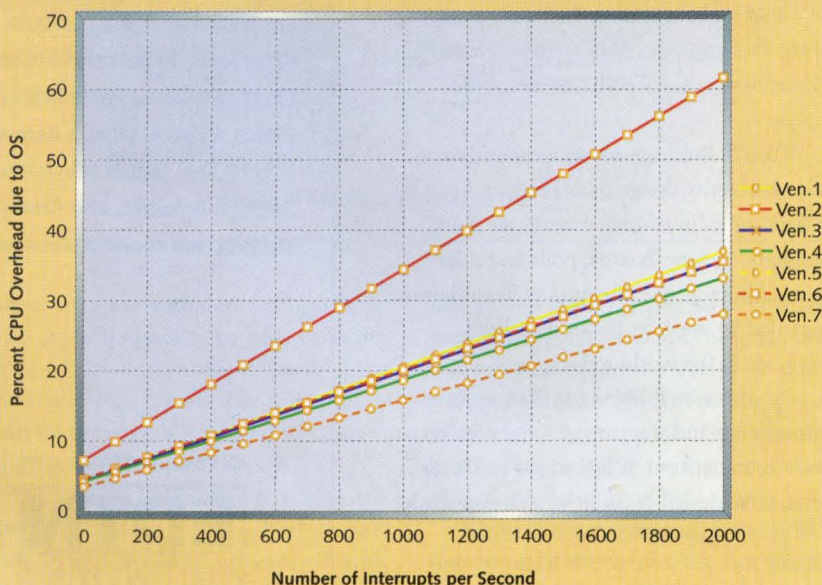
TABLE 2 RAM for 68332 kernels for example architecture

	RAM total
OS1	1473 bytes
OS2	3841 bytes
OS3	2159 bytes
OS4	1938 bytes
OS5	2137 bytes
OS6	1902 bytes
OS7	3385 bytes
OS8	2493 bytes
OS9	1485 bytes
OS10	3764 bytes

TABLE 3 PowerPC best periodic and asynchronous OS overhead timing

	Tick	Best Periodic	Best ISR
Ven. 1	47.01	124.855	163.395
Ven. 2	109.74	177.61	274.585
Ven. 3	31.79	132.16	158.695
Ven. 4	66.505	72.765	122.895
Ven. 5	38.395	115.62	158.955
Ven. 6	38.055	119.135	158.175
Ven. 7	12.38	128.035	123.265

FIGURE 7 Number of interrupts per second



Scott Ranville has a BSEE and an MS in control theory engineering from the University of Michigan. After college, Scott started work at Ford in the electrical fuel and handling division developing in-vehicle battery charging algorithms. In 1995, he transferred to the scientific research laboratory within Ford, where his current research topics include automatic code generation from a model, unit testing, and real-time scheduling. He can be reached by e-mail at sranvill@ford.com.

Steven Toeppe is a product development manager at The MathWorks. He is primarily focused on developing simulation products for automotive and production intent applications. Previously he conducted research at Ford Motor Company Research Laboratories, focusing on software, control system modeling, and simulation technology.

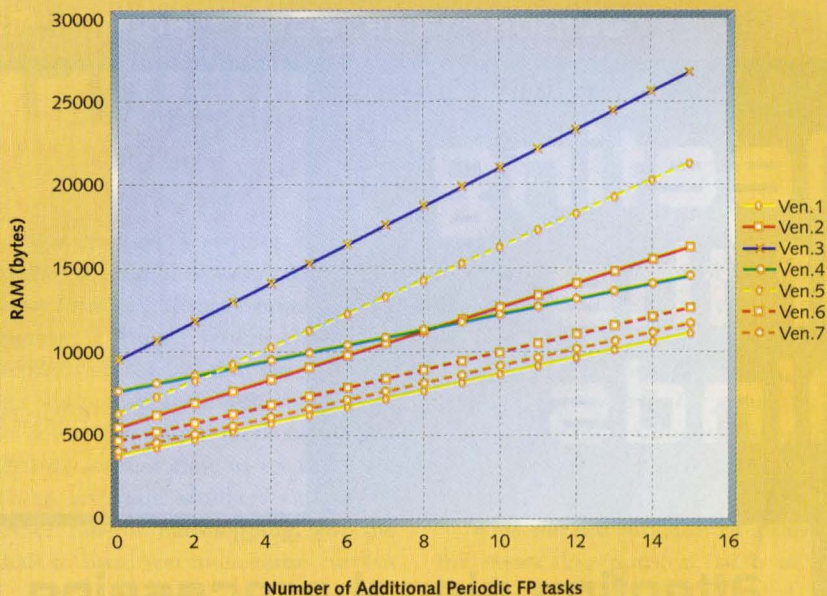
gy. He has a MSEE and BSEE and 20 years of software engineering experience. He has held a variety of positions managing the development of embedded systems software. E-mail him at stoeppe@mathworks.com.

Special thanks to Ed Nelson (Ford Research), Steve Mihalik (Motorola), and numerous RTOS sales people and FAEs.

References

1. OSEK/VDX Operating System Specification 2.0, OSEK Group, June 1997.
2. OSEK/VDX Communication Specification Version 2.1, revision 1, OSEK Group, June 1998.
3. K. M. Zuberi, P. Pillai, K. G. Shin, "EMERALDS-OSEK: A Small Real-Time Operating System for Automotive Control and Monitoring," 1999 SAE Proceedings, February 1999.
4. K. M. Zuberi, K. G. Shin, "EMERALDS: A Microkernel for Embedded Real-Time Systems" in Proc.

FIGURE 8 Number of additional periodic FP tasks



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.

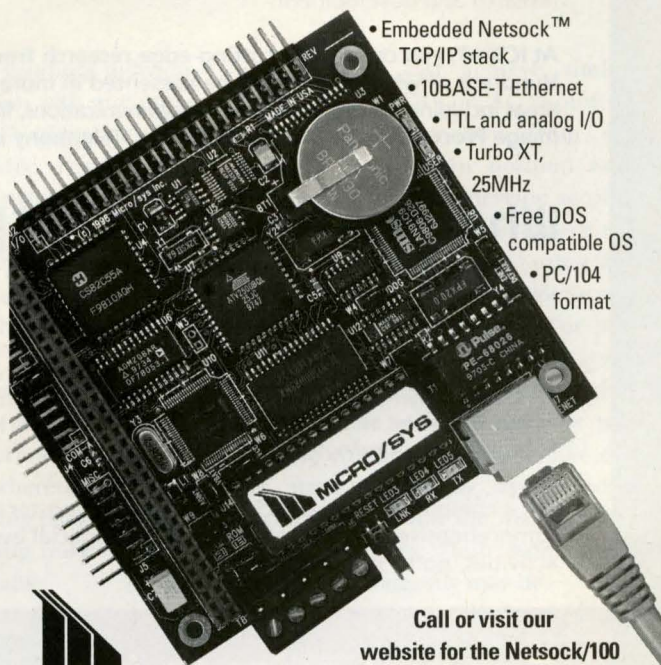
It might just put the sparkle back in your eyes!

CMX RTOSes and TCP/IP Stacks Support Most 8-, 16-, 32-bit Processors and DSP's.



680 Worcester Road
Framingham MA 01702
Ph: (508) 872-7675
Fax: (508) 620-6828
email: cmx@cmx.com
WWW: www.cmx.com

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



MICRO/SYS

Glendale, CA
(818) 244-4600 • Fax: (818) 244-4246
www.embeddedsys.com

Call or visit our website for the Netsock/100 data sheet or a FREE 284-page Handbook

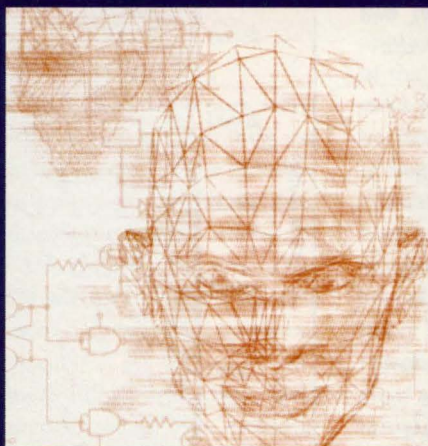


ICSPAT

International Conference on Signal Processing Applications & Technology

ICSPAT October 16-19, 2000 • Adam's Mark Hotel • Dallas, TX, USA

Meeting of the Minds



View the complete
ICSPAT program online
at www.dspworld.com

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

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
- A chance to network with engineers and researchers from around the world
- Industry activities including the Vendor Seminar Series with leading industry representatives
- The official proceedings on CD-ROM

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.

ESP1

Special events include:

- 3 Full-Day Tutorials • FREE Product Training • Plenary Panel Discussion
- Special Guest Lecturers • ICSPAT Awards Ceremony • Keynote Address

Sponsors:

CMP
EETIMES

CMP
Communication
design

CMP
EmbeddedSystems

CMP

www.dspworld.com



Don Morgan

The PID Filter

Motion control has many applications and, as a result, many configurations. It can be something as simple as a switch for fan or pump motors, or highly complex, involving three-space mathematics, electronic gearing, and sophisticated jerk control for robotics and sensitive equipment. In this column, we will address the proportional integral differential (PID) controller mechanism, which is ubiquitous in the motion control industry. Though terminology varies, as does implementation, we will attempt to remain as simple and as broadly applicable as possible.

A simple control system might amount to little more than a motor and two end-of-travel limit switches. With a D.C. motor, one could control the direction of motion with the direction of current flow. The motor would turn in one direction until stopped, either by the user or the end-of-travel limit switches. This is pretty simple and might well be useful as long as performance or efficiency is not an issue. But there are times when one cannot simply switch the current on and off to control the motion. Either the mechanics of the system or some characteristic of the load may require a controlled start or a controlled stop. In this case, a trajectory generator is an appropriate alternative.

A step beyond the simple on/off switch approach is the incremental system. This kind of approach involves no feedback but depends upon proper equipment design to perform the necessary functions, while the control feeds commands to the equipment.

You can use a stepper motor to devise a simple system employing some form of trajectory generation. A motion control system like this depends on the fact that the stepper motor has a multiplicity of poles or *detents* that map directly to the pattern of current flow in its windings. There are four windings capable of either full- or half-stepping. For the shaft to turn, you must cause current

application) by the states of the four metal oxide semiconductor field effect transistors (MOSFET), we can use that information to create predictable accelerations and decelerations. This way, we can control, for example, a table or lazy susan without spilling the contents or load onto the user with each move.

With the introduction of a device for measuring position, such as an

PID may seem straightforward, but it's not. You won't find code that's universal or standardized. The most you can hope for is some guidelines.

to flow through the windings individually or in combination with one other winding in a certain sequence. This sequence is invariable. For the software engineer, this sequence becomes a table. Traveling the table from top to bottom causes the shaft to turn in one direction; traveling the table from bottom to top causes the shaft to turn in the other direction.

A typical stepper motor will have magnetic detents at 1.8 degree intervals. With this information, you can accurately predict where you are in a revolution by counting passes through the table and keeping track of where you are within the table.

Knowing where you are

Since we are always able to locate the shaft (assuming you have a stepper motor with enough torque for the

encoder, we can achieve similar results with a DC or AC motor by using the information from the encoder to ascertain position and, based upon this data, develop accelerations and decelerations. These systems are basically passive, using any information inferred or measured about position to stop, start, or produce an elementary trapezoidal acceleration control. Actually, because they can be made to suit known or predictable conditions, controls like this are used for many applications. Performance is not the main goal in these instances, but safety and mechanical longevity may be.

There are some basic problems with this type of system. First, there is no facility to correct for changes in load or environment. Another problem is that characteristics of the load can prevent the system from produc-

To get the kind of performance that will support these applications, people typically turn to some form of servo control that uses the PID filter.

ing the necessary motion. In addition, there is no optimization of motion for speed or efficiency.

Of course, if changes in the load or system fall outside the constraints of the original design, the motor may simply refuse to turn at all. Or if the load or carriage changes too dramatically, it may hurl whatever it's carrying from its position during travel. These are gross changes and can usually be corrected.

A still more significant disadvantage is that the control is not capable of sensing and correcting for characteristics of the motor or load that allow it to optimize its control or fit it to the particular devices you use.

Take, for instance, a laser light control. So little inertia is involved in a system like this that its simple controls can result in overshoot, which would not be acceptable for an application such as, say, surgery. Or suppose you were making a cutting machine and had to cut a pattern to very tight specifications. Maybe you could fit the specification with a simple control by moving very slowly, but that, since profits lie in speed and accuracy, is not what production wants. And what if you were designing a system to carry the space shuttle from its garage to the launching platform? We can go on and on with this. It is clear that in numerous applications, precision of control and speed are important.

The PID filter

To get the kind of performance that will support these applications, people typically turn to some form of servo control that uses the PID filter. The interesting feature of this type of control is that it depends on error to produce performance. We will start with a simple description of a PID servo system and take on greater com-

plexity as our understanding of the system progresses.

Servo systems use prediction based on absolute time to determine what position, velocity, or torque should be at any particular moment—this is called *commanded* position, velocity or torque. It compares this value with information from the motor, or system, about how it is actually doing—this is called *actual* position, velocity, or torque. Please note that position, velocity, or torque are not the only things that can be controlled here, simply the most common.

Within this system, there is a trajectory generator that predicts where the system should be, how fast it should be moving, or how much force it should be delivering to its load at precise units of time, typically called *servo cycles*. The length of the cycle can be anywhere from tens of microseconds to hundreds of milliseconds.

The controls for a servo system are commonly (but not exclusively) called *gains*. These are the parameters used to scale the different errors (difference between commanded and actual position, velocity or torque) for the controller so that it can correct appropriately.

The primary force in a servo system is the *proportional gain*, which is a gain applied to the error in the system. The error in the system is the difference between where the shaft of the motor or the load should be and where it actually is. One could make a controller based on this alone, but it would not be optimal because it could take a very long time to arrive at its destination, as you can imagine. As it approaches the target, the error falls off and as a result, so does the force used to drive it there. It can easily happen that there is not enough force to push it the final distance and the controller falls short of its destination.

Can't you just turn up the proportional? Yes, but this can result in missing the final position and oscillation.

To overcome this problem, we make use another error, one which is based on the difference between actual position and commanded position and referred to as *following error*. This is the *integral gain*. We sum or integrate the following error over time and add this to the proportional gain to produce enough force to make it to the target position, velocity, or torque. This means that if the proportional gain is low for any reason, the integral of the error can add to that force and help push us home. Remember, the integral gain is the amount by which we multiply the integral of the error; it is *not* the integral. Too much integral gain can result in oscillation.

Of course, as you approach the target, this error falls away until, when the target is actually reached, it becomes zero.

The goal of using these errors to drive the controller is to tune a system so that it goes to velocity or position in the shortest time possible, with no overshoot or undershoot. If you were to plot the step response of such a system—whether velocity-, position-, or torque-controlled—it would result in a perfectly square step. That is the goal, though it is not always easy to attain.

Another error that can be added to improve performance is the derivative gain. This is a gain applied to the derivative of the error, which is to say, the rate of change in the error. This is helpful for correcting problems in step response, in particular those errors that occur during acceleration or deceleration to predicted values. This is the spring action in the differential equation associated with the filter. It is predictive in that it alerts the filter that the rate of change is either increasing or decreasing, depending upon whether you are accelerating or decelerating to the target. If you are getting close to the target but your rate of change

is very high, this term can help slow the system and eliminate overshoot. Again, if you are accelerating and the rate of change is too slow, this gain can increase it to eliminate under-shoot.

The math actually makes some other aspects of the system very clear.

PID: the math

A simple formulation for the PID is:

$$U[t] = Kd \frac{de}{dt} + Kpe[t] + Kie[t]dt$$

$U[t]$ is the output, Kd is the derivative gain, Kp is the proportional gain, and Ki is the integral gain. It is simple indeed.

An interesting aspect of this sort of system is immediately evident in this relationship: when the error goes to zero, the output is zero. In systems with low gains, you will experience what is called a "loose" response. Even though the system may not be dead on the commanded value, there is little effort (because the gains are low) to get it there. This is not optimal for most situations.

For this reason, most designers try to produce as high a gain as possible, which means they need low noise, high-resolution input. Since the PID loop depends on error, it must be off the commanded target to produce a correction value. On a low-resolution, high gain system, this can result in oscillations. If the resolution of the sensors is very high, however, any deviation will produce error values that can be multiplied by even moderate gains to produce a "stiffer" response.

The code

Unfortunately for a concept that seems so straightforward, the code for PID is not universal or standardized. Given a specific application, you could set up two controllers for different companies with exactly the same gains

(assuming a consistent terminology) and get different results.

It is a simple concept with many different implementations. Here I will stick to the basics, which I suspect you have figured out already from the preceding text.

In the pseudo-code that follows, `cmd` is commanded output, `actl` is actual output, `sum` will be the integral or sum of the error, and `error` will play himself. The gains will be Kp for proportional, Ki for integral, and Kd for derivative. U is the output and dt is the servo update rate. In some cases, `old` is appended to a variable to indicate that it is the last value:

```
error=cmd-actl;
sum=oldsum+error;
U=Kp*error+Ki*sum*dt+Kd*(error-
olderror)/dt;
Olderror=error;
```

Next month

We will look at the Park and Clarke algorithms commonly used in DSP-based control for AC machines. This algorithm allows phase variable information to be transformed to a static frame for computation and then re-transformed to the dynamic frame for driving the system. **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.

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 BIOSStart™ 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



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

© 2000 General Software, Inc. All rights reserved. General Software, the GS Logo, and Embedded BIOS are trademarks of General Software Inc.



Tools for Embedded Developers

Software

RTOS for StarCore SC100

The RTXCDSP real-time operating system targets StarCore SC100-based applications for Motorola's semiconductor products sector and Lucent Technologies' microelectronics group. The RTXCDSP kernel features a modular architecture which allows users to match scheduling architecture with application requirements. Implementations include single-stack, which provides multi-threading and level-specific thread priorities, and multiple stack, which enables management of a set of kernel objects that the user can define statically (through a host-based configuration module) or dynamically, by program code at run time. Dual-mode implementation is also an option. It's available now.

Embedded Power Corp.

Austin, TX
(512) 338-9211
www.embeddedpower.com

Analyzer/exerciser

SBAE-10 is a self-contained tool that enables USB designers to perform testing, debugging, and verification for USB protocols and DC parameters. Its features include host or device mode, protocol analysis, inrush current analysis, performance analysis, real-time and off-line statistical analysis, and 720MHz timing analysis. It is also USB 2.0 upgradeable. Timing characterization of USB signals for skew measurement and integrity verification is possible down to 1.3ns resolution on USB 1.x and will be available down to 700ps resolution on USB 2.0. The starting price

for the SBAE-10 analyzer is \$7,900. It's available now.

Catalyst Enterprises, Inc

San Jose, CA
(408) 268-4145
www.catalyst-ent.com

Design tool

System View is a system-level design tool for DSP and communications applications. Version 4.5 links System View with TI Code Composer Studio for C5x/C6x software development and test. It includes new libraries with new models for communications applications, enabling system designers to simulate a range of communications systems without having to develop custom functions. RSP and RF designers can immediately simulate and test their subsystems. System View v. 4.5 features Real Time DSP Architect, which supports the TI TMS320C54x family of DSPs. This includes the TI TMS320C549 and 5410 EVM boards. The software is targeted to designers developing products for 2G, 3G, and Bluetooth communications standards. System View v. 4.5 is available now for Windows 95, Windows 98, and Windows NT hosts. Configurations start at \$3,995.

Elanix Inc.

Westlake Village, CA
(818) 597-1414
www.elanix.com

Hardware

Development kit

The Rabbit 2000 TCP/IP Development Kit provides a hardware platform with an 8-bit microprocessor

and Dynamic C development software for developing 10baseT Ethernet applications. The development board included in the kit allows executable code to be downloaded into flash memory or optional battery-backed RAM. Two communication ports are available: an RS-232 port and a factory configurable port for either RS-485 or RS 232. Other features of the development board include four high-current outputs, four digital inputs, seven timers, a real-time battery-backable clock, and 10baseT Ethernet interface. Source code is provided in addition to the Dynamic C software on CD-ROM, as well as ICMP, HTTP, SMTP, FTP, and TFTP capabilities. Users can write directly to TCP or UDP sockets to develop custom applications. The kit costs \$199 and is available now.

RABBIT Semiconductors

Davis, CA
(530) 757-8400
www.rabbitsemiconductor.com

In-circuit emulator

The DProbeHS is an in-circuit emulator that supports the 8051 family of TEMIC semiconductors. It supports selected 8051 derivatives up to 70MHz operating frequency. The emulator also features between 2.7V and 5.5V operating voltage, up to 1MB dual-ported emulation memory, up to 1MB hardware breakpoints for code, 64K hardware breakpoints for data read and data write, and optional 32K frames plug-on trace DTrace16. The DProbeHS is available now.

Hitex Development Tools

Sunnyvale, CA
(800) 454-4839
www.hitex.com

Chips

DSP chip family

The StarPro family of DSP chips is based on a modular DSP platform architecture and system-on-chip methodology, allowing customization for a number of infrastructure application environments. StarPro 2000 is the first chip in the family. It integrates three StarCore SC140, 768K of on-chip shared memory, and application-specific peripherals with direct memory access, all interconnected by the Daytona bus. In addition, this chip operates at up to 3,600 MMACs (million multiply and accumulates per second) and has an operating frequency of 300MHz, while consuming 1.5W at full-speed. It will be available in sample quantities in April 2001, and in production quantities later that summer. It is \$100 each in quantities of 100,000.

Lucent Technologies

Allentown, PA
(800) 372-2447
www.lucent.com

Applied computing processors

Higher performance Pentium III and Celeron processors are now available. The Pentium III comes in three versions. One features 733MHz speed, a 256K L2 cache on die, and a 133MHz processor side bus, in FC-PGA package, and another features 700MHz speed, 256K L2 cache on die, and a 100MHz processor side bus, in FC-PGA package. There is also a low-power version which operates at 500MHz in BGA package. The 700MHz Pentium III is supported by both the 440BX AGP chipset and the 810 chipset. The 440BX AGP chipset supports the low-power Pentium III. There are two versions for the Celeron proces-

sor. One operates at 566MHz, in FC-PGA package, and features 128K L2 cache on die. The other processor is low power, and operates at 400MHz, in BGA package. The low power Celeron is supported by the 440BX AGP chipset.

Intel

Santa Clara, CA
(408) 765-8080
www.intel.com

OEMS

System board

The CPCI-947 is a 64-bit CompactPCI system board for building embedded and real-time systems. Configured as a PCI system host processor board, the CPCI-947 supports an i960 data movement processor that opens up I/O bottlenecks and facilitates data throughput for embedded applications. Dual 10/100BaseT Ethernet controllers allow the board to be used as a processing engine for either intelligent LAN communications or as redundant links for fault-tolerant applications. The CPCI-947 hosts one PMC module for custom I/O expansion. It also features two 64-bit PCI interfaces connected to a PCI-to-PCI bridge and 64MB SDRAM (expandable to 128MB). The SDRAM controller is capable of data transfers twice as fast as either PCI bus (528MBps peak transfer rate). A PMC Module interface resides on the secondary PCI bus and can be accessed by the board's i960RN processor or other devices on the CompactPCI bus. The is available now, starting at \$1,627.

Cyclone Microsystems

New Haven, CT
(203) 786-5536
www.cyclone.com



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, 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.

 **TASKING**

**The Embedded
Communications
Company**

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.

3 0 3 - 6 2 8 - 5 5 6 0

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:



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?

See our Online Salary Calculator!

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
- Motorola 68HC11, Motorola 68332 Microprocessor experience
- Algorithm development, device drivers, SLC, 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.

Engineering Associates

- 4-year degree in Electrical Engineering or Computer Engineering required

All positions require relocation to Southeast Michigan. Please mail, fax or email your resume, indicating position of interest, to: EDS Recruiting, Dept. 72-9578, Attn: Deborah Polvi, 700 Tower Drive, 5th Floor, Troy, MI 48098; Fax: (520) 833-9309; Email: deborah.polvi@eds.com.



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.

scientific.com

Software and Hardware
Professionals

Nationwide opportunities available for:
Engineers/Designers

Quality Assurance/Testers/Verification

- | | |
|---------------|------------|
| • Embedded SW | • C/C++ |
| • RTOS | • Firmware |
| • ASIC/FPGA | • VLSI |
| • IC Design | • Java |

Scientific Placement, Inc.

800-231-5920 800-757-9003 (Fax)

crs@scientific.com

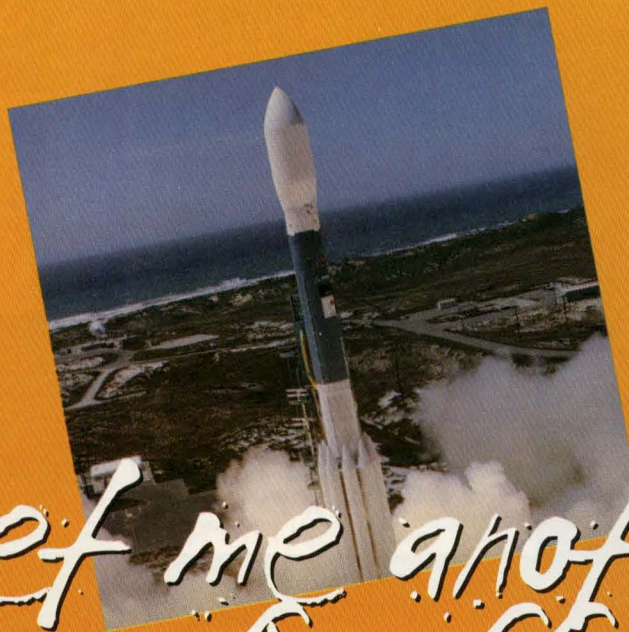
Free Resume Assistance
All fees are employer paid



**Having
problems
landing
qualified
embedded
designers?**

**Advertise in
Embedded
Systems
Programming's
career section!**

Contact Sam
Louis
415/278.5223



"Get me another
cup of coffee"
is cool
if preceding,
"then we'll launch
the satellite."

www.boeing.com/employment

 **BOEING**
Forever New Frontiers

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



FREE
2 WEEK
TRIAL

www.signum.com

SIGNUM
SYSTEMS

800-838-8012
805-523-9774
Fax: 805-523-9776
11992 Challenger Ct. • Moorpark, CA 93021

Windows[®] CE Embedded Computer

Visual Basic or Visual C/C++

Small Powerful & Easy To Program.
Plug the CE-Minus-SC400 (486 SBC) into your next custom design or into our CE-Plus and LCD-Plus I/O expansion options. The CE-Minus is easy to program using our custom ported Windows-CE & Tools.

www.RLC.com
Enterprises, Inc.
Toll Free 1-888-RLC-TECH

CE-Minus
Single Board Computer

\$199 Qty 100



2.5" x 4.5"

1/4 VGA LCD
320 x 240

Monochrome
Display



Touch Screen
Standard

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 2.39"

- 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



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

WE SUPPORT YOU FROM DEVELOPMENT...

LV48 Series

- Supports over 5500 devices
- Low voltage to 1.8V
- Windows 95/98/NT software
- EPMaster S495
- Micromaster S995



...TO PRODUCTION



Matrix

- Reliable, high speed production programmers
- 28F800 9 sec program
- Link up to 48 sites with no loss of speed

ICE Technology Inc.
430 Peninsula Ave, Suite 4
San Mateo, CA 94401
iceusa@icetech.com
www.icetech.com

1 888 ICE 2305
1 888 423 2305
Fax: 1 650 375 8666
outside US/Canada call:
+1 650 375 0427
credit cards accepted



15 year
Embedded software
market leader in USA
for sale.

Direct inquiries
to:
agilson@manatt.com

Fast. Reliable. Affordable.



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

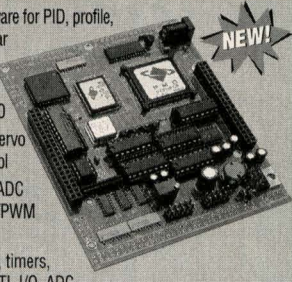
(916) 924-8037

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/>



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!



1724 Picasso Ave., Suite A
Davis, CA 95616 USA
Tel: 530-758-0180 • Fax: 530-758-0181
<http://www.tern.com>
tern@netcom.com



www.nci-usa.com

The fastest, easiest, most affordable way to test digital designs.

- 72 Channels @ 250MHz
- 36 Channels @ 500MHz
- Up to 2 Meg/Channel
- Easy-to-use Windows Interface



NEW!
Portable
Logic
Analyzer



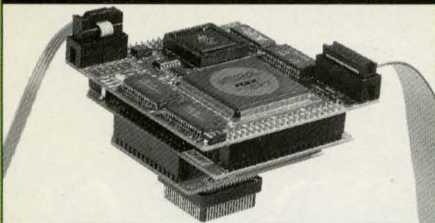
GoLogic™

72 Channels \$3,995
36 Channels \$2,995

NCI Logic Analyzers

6438 University Drive • Huntsville, AL 35806
phone: 256.837.6667 • fax: 256.837.5221

8051, 80C196 PIC®, AVR®

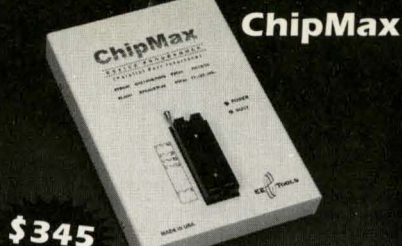


Integrated Development Systems: Compilers, Simulators, Programmers, In-Circuit Emulators

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**



\$345

- Programs over 1200 devices (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



Sunnyvale, California, USA
Tel: 408-734-8184
Fax: 408-734-8185
www.eetools.com

**"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



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.intec.com
email: sales@intec.com
U.S. and Canada: 800.327.7171
Other: 414.273.6100
FAX: 414.273.6106



SBC2000-332

Vesta BASIC
or C Software

32 bit 68332

Low Power

2 x RS-232

Watchdog

Real Time Clock

EPROM

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



Headquarters, Arizona, USA
Tel: 480.926.0797
Fax: 480.926.1198
sales@metaice.com
www.metaice.com

MetaLink - Europe GmbH
Tel: 49 (8091) 56960
Fax: 49 (8091) 2386
islinger@metalink.de
www.metalink.de

Debugger+Programmer

Complete System **\$375**

FREE

Demo/Prototype board

All tools **qualified** by
Scenix Semiconductor



SX-ISO

- 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

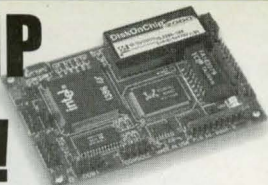
Probes for **16F87x** and **16C77x** by Jan 2000
With data break support

Advanced TransDATA
CORPORATION

Dallas, Texas
Tel 972.980.2960

www.adv-transdata.com

JUMP on the NET!



The μ FlashTCP gives you 10BASE-T Ethernet connectivity, a full-function TCP/IP stack and 2 serial ports in a package 30% smaller than PC/104 solutions.

Field-proven TCP/IP stack, DOS and PC-compatible BIOS make development quick and easy.

Prices start at \$169 qty 100.
Development kits are available.

Call 530-297-6073 Fax 530-297-6074
Check our web site for the latest updates
www.jkmicro.com/uflash

JK microsystems

From the Author
of WATTCP

eRTOS

DOS
Realtime
Kernel with
TCP/IP Support

- Preemptive & Cooperative threads
- Critical Section Protection
- Interthread Messaging
- Complete Re-entrant TCP/IP
- Web, CGI, FTP, Email, Telnet
- Web Graphics
- Interrupt-driven Serial Support

www.ertos.com

Call 530-297-6073 Fax 530-297-6074

JK microsystems

NOW!

Optimised
ANSI C
on the
Microchip
PIC.

the
world's
first

pic

ANSI C
COMPILER

CALL NOW
1-800-735-5715

Fax 1-407-722-2902
www.htsoft.com/pic sales@htsoft.com

HI-TECH
SOFTWARE

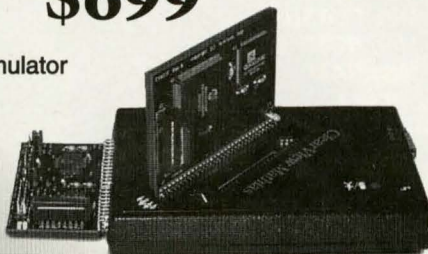
Embedded Systems Development Tools



Complete PICmicro[®] Development System

Get the TOTAL Package for only
\$699

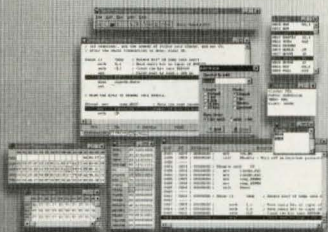
PICmicro[®] In-Circuit Emulator
PICmicro[®] Assembler
PICmicro[®] Debugger
Windows[®] IDE



ClearView[™] Mathias is a full-featured In-Circuit Emulator with a highly productive Development and Debugging environment for the PICmicro.

Fully emulates the selected PICmicro, including program memory, register memory, EEPROM, I/O activity, SLEEP mode and all peripherals.

Intuitive, Easy to Learn full-featured environment with integrated ClearView Debugger.



TDE provides integrated source-level debugging for ALL popular PIC Compilers and Assemblers.

See why Engineers choose TechTools

Visit our website and request your FREE CDROM!

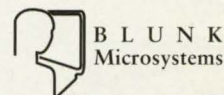
www.tech-tools.com

CONTACTS: (972) 272-9392 Fax: (972) 494-5814 Email: sales@tech-tools.com

Copyright © 1999, TechTools, P.O. Box 462101, Garland, Texas 75046-2101 • ClearView, CVASM16, PICwriter, PICstation, the "Wizard" symbol and TechTools are trademarks of TechTools, P.O. Box 462101, Garland, Texas 75046. • PICmicro is a registered trademark of Microchip Technology, Inc. • All other trademarks are trademarks or registered trademarks of their respective company.

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



(408) 323-1758

www.blunkmicro.com

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

8051 EMULATORS

New!



SOC51Rx2

Ceibo USA: 1-800-833-4084

Tel.: 314-830-4084

Fax.: 314-830-4083

E-MAIL: info@ceibo.com



www.ceibo.com

- * Online Ordering
- * Electronic Catalog

- * Software Updates
- * Price List



Real-Time Debugging

CPU32 ColdFire 68HC12
68302 68020 68306

call toll free

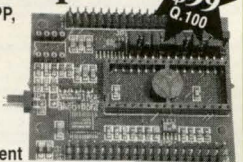
info@noral.com

888-88-DEBUG

www.noral.com

EMBEDDED INTERNET MADE EASY! with the DOS Stamp™

- Free TCP/IP, PPP, Mini-server
- Easy to Use
- Low Cost
- Low Power
- Tiny Size
- High Speed
- DOS Environment



From \$99 Q.100

Easy Software Development: Use your C/C++ or Basic compiler to produce DOS EXE. Download EXE to flash disk via serial port. EXE runs on power up.

Standard Features: BIOS & DOS-ROM, 128K flash disk, 512K SRAM, 40 MHz AM188ES CPU, 16 digital I/O (opto rack interface), 2 RS-232, 2 timer/counters, simple bus interface, real-time clock with timed power-up

Options: 8-ch 12-bit ADC, flash disks up to 288 MB, 1 RS-485 Tiny Size, Low Power: 2"x2.6", 5V @ 200 mA at full speed.

Visit <http://www.bagotronix.com> for info, prices, and FAQs

BAGOTRONIX
2900-1 Crescent Drive
Tallahassee, FL 32308
850-942-7905 phone & fax

GALEP-III Pocket Multiprogrammer

This size fits all!



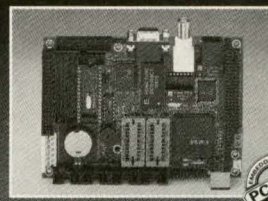
- ◆ Programs 8-bit and 16-bit EPROMs, EEPROMs, Zero Power RAMs, Flash, serial EEPROMs ◆ GAL, PALCE, ATF ◆ 87xxx, 89xxx, PIC12/16/17Cxx ◆ All DIL devices **without adaptor** ◆ **Lightning fast** parallel data transfer (e.g. 27C512 read/compare 2 sec) ◆ Power supply independent due to **rechargeable battery** ◆ 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 / GAL... each \$149.00

1951 4th Ave, Ste 301 • San Diego, CA 92101
Tel: (619) 702-4420 • <http://www.conitec.com>

386SX40 PC104 Embedded Controller



Ali M6117D Intel compatible 386SX40MHz

- 4MB System DRAM on board
- HMC HM8508 VGA Chip w/1MB DRAM
- Support DiskOnChip up to 144MB
- REALTEK 8019AS Ethernet on board
- 4S/PIDE/KB/FDD/DIO on board
- 7 Level Watch Dog Timer
- Support DOS, Window 3.X, Linux, XvWork, and QNX
- 145mm x 102mm (5 3/4"x4"), 2 side PCB

Other PC/104 relative products available:
486SX, AC/DC, Digital I/O, Ethernet Link, 4 Port RS232

NUCLEUS Electronic Corp. 800-683-7335

Tel: (909) 468-5700

Fax: (909) 468-5704

<http://www.nucleus1.com>

Email: info@nucleus1.com

QuickTool Finder

www.avocetsystems.com

Every Chip
Every Tool
One Source
One Click

C Compilers

Assemblers

Simulators

RTOS

In-Circuit Emulators

BDMs

AVOCET SYSTEMS, INC.
(800) 448-8500

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
8x751/2
C500
DS5000
8080/85
8051XA
80166
80196xx
8086/88
x86 Intel/AMD
80386EX
80186/188xx
Conexant/WDC
6502/6516
C18/19/29
Z180/64180/Z80
Z8/Z8000/Zx80
Microchip PIC
V20/30/50
20C20
TLCS-90

UniSTAC for Strong ARM designs



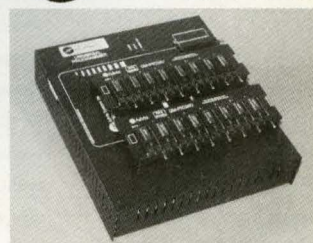
UniSTAC is a full-featured, high-end development system for Strong ARM® SA-1110 designs. Powerful In-circuit Emulator featuring: 8 Mbytes emulation memory, non-intrusive Real-time trace capture and display, support for hardware breakpoints.

Also Supported:
Power PC, ARM7/9,186/386/486, Pentium®, 680x0, SuperH, V800/850E, R3000/4000, M16C, M32R

- **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®

Sophia systems™
408-467-9911

www.sophia.com



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-ADVING 408-243-7000

Advertiser	URL	Page	Advertiser	URL	Page
Accelerated Technology	www.atinucleus.com	111	Introl Corp.	www.introl.com	133
Advanced Transdata	www.adv-transdata.com	133	iSYSTEM USA	www.isystem.com	80
Advantech	www.advantec.com/epc	26	JK Microsystems	www.jkmicro.com/uflash	134
Advin Systems	www.advin.com	135	JK Microsystems	www.jkmicro.com	134
Agilent Technologies	www.agilent.com	13	KADAK Products Ltd.	www.kadak.com	107
Agilent Technologies	www.agilent.com	79	Keil Software	www.keil.com	114
Allant Software Corp.	www.allant.com	33	Lantronix	www.lantronix.com	14,15
American Arium	www.arium.com	C3	Lauterbach	www.lauterbach.com	87
Archimedes Software	www.archimedessoftware.com	132	Lineo	www.lineo.com	45
ARTISAN Software Tools	www.artisansw.com	39	Linuxworks	www.linuxworks.com	42
Avocet Systems Inc.	www.avocetsystems.com	C4	Mentor Graphics	www.mentor.com	94
Avocet Systems Inc.	www.avocetsystems.com	135	Metalink Corp.	www.metaice.com	133
Axiom Technology	www.axiomtek.com	99	MetaWare Inc.	www.metaware.com	73
Bagotronics	www.bagotronics.com	135	Micro Digital	www.smxinfo.com	89
Blackhawk	www.blackhawk-dsp.com	106	Micro/sys	www.embeddedsys.com	123
Blunk Microsystems	www.blunkmicro.com	134	Microchip	www.microchip.com	31
Boeing	www.boeing.com/employment	131	Microsoft Inc.	www.microsoft.com/embedded/	23
BSQUARE Consulting	www.bsquare.com	71	Microtek International	www.microtekintl.com	4
Ceibo	www.ceibo.com	135	Microware Systems Corp.	www.microware.com	115
CMX Systems Inc.	www.cmx.com	123	Motorola	www.digitaldna.motorola.com	11
Cogent Computer Systems	www.cogcomp.com	142	National Semiconductor	www.national.com	67
CONITEC Datasystems Inc.	www.conitec.com	135	National Engineer Search	www.nesnet.com	130
COSMIC Software	www.cosmic-software.com	116	NCI	www.nci-usa.com	132
Dinkumware Ltd.	www.dinkumware.com	141	Needhams Electronics	www.needhams.com	132
Domain Technologies	www.domaintec.com	134	Nohau	www.nohau.com	C2
EBS	www.ertfs.com	141	Noral Micrologics Inc.	www.noral.com	135
EBSnet Inc.	www.ebsnetinc.com	63	Nucleus Electronics	www.nucleus1.com	135
Electronic Data Systems	www.eds.com/careers	130	Pacific Softworks	www.pacsoft.com	51
Electronic Engineering Tools	www.eetools.com	133	Paradigm	www.devtools.com	86
EMAC, Inc.	www.emacinc.com	72	ParaSoft Corp.	www.parasoft.com	35
Embedded Power Co.	www.embeddedpower.com	27	Phar Lap Software Inc.	www.pharlap.com	32
Embedded Systems Academy	www.americanraisonance.com	102	Phyton Inc.	www.phyton.com	133
Embeddedtechnology.com	www.embeddedtechnology.com	105	PointBase	www.pointbase.com	41
emWare	www.emware.com	59	Premia Corp.	www.premia.com	37
Enea OSE Systems	www.enea.com	56	Rabbit Semiconductor	www.rabbitsemiconductor.com	103
esmertec	www.esmertec.com	61	Radisys Corp.	www.radisys.com/SS7	77
Espial Group	www.espial.com	65	RLC Enterprises	www.rlc.com	132
EST Corp.	www.estc.com	1	Scientific Placement	www.scientific.com	130
Express Logic	www.expresslogic.com	113	Signum Systems	www.signum.com	132
General Software	www.gensw.com	127	Sophia Systems	www.sophia.com	135
Grammar Engine Inc.	www.gei.com	142	TASKING	www.tasking.com	129
Green Hills Software	www.ghs.com	6	Tech Tools	www.tech-tools.com	134
Hitachi Semiconductor Corp.	www.superh.com	25	Tektronix	www.tektronix.com/TLA600	83
HI-TECH Software	www.htsoft.com/pic	134	Tern Inc.	www.tern.com	132
HI-TECH Software	www.htsoft.com	96	The Math Works	www.mathworks.com/escp	69
Hitex Development Tools	www.hitex.com	85	TimeSys Corp.	www.timesys.com	47
Hitex Development Tools	www.hitex.com	133	Treck Inc.	www.treck.com	55
Hiware	www.hiware.com	119	U.S. Software	www.ussw.com	53
IAR Systems	www.iar.com	91	Vesta Technology	www.sbc2000.com	133
ICE Technology Inc.	www.icetech.com	132	Wind River Systems	www.wrs.com	8,9
I-Logix	www.ilogix.com	81	Working Engines Inc.	www.workingEngines.com	130
I-Logix	www.ilogix.com	19	Waferscale	www.waferscale.com	38
Infineon Technologies	www.infineon.com	74	Xilinx	www.xilinx.com	81
Infineon Technologies	www.infineon.com	21	ZF Linux Devices	www.zflinux.com	17
InterNiche Technologies Inc.	www.iniche.com	49	Z World	www.zworld.com	132

FREE SUBSCRIPTION REQUEST FORM

Embedded Systems PROGRAMMING

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.

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 μ C/ μ P
- 03 ☐ 16-bit μ C/ μ P
- 04 ☐ 32-bit μ C/ μ P
- 05 ☐ 64-bit μ C/ μ P
- 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 →

www.espmag.com

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 <input type="checkbox"/> 1 | 05 <input type="checkbox"/> 5 |
| 02 <input type="checkbox"/> 2 | 06 <input type="checkbox"/> 6 |
| 03 <input type="checkbox"/> 3 | 07 <input type="checkbox"/> 7 |
| 04 <input type="checkbox"/> 4 | 08 <input type="checkbox"/> 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



NORTHBROOK IL 60065-9468

PO BOX 3404

Embedded Systems



POSTAGE WILL BE PAID BY ADDRESSEE

FIRST-CLASS MAIL PERMIT NO. 2253 NORTHBROOK IL

BUSINESS REPLY MAIL



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



PLEASE TAPE ALONG THIS EDGE. NO STAPLES.

CMP MEDIA PRODUCT CATALOGS

Complete Embedded Product Information

Available Now!

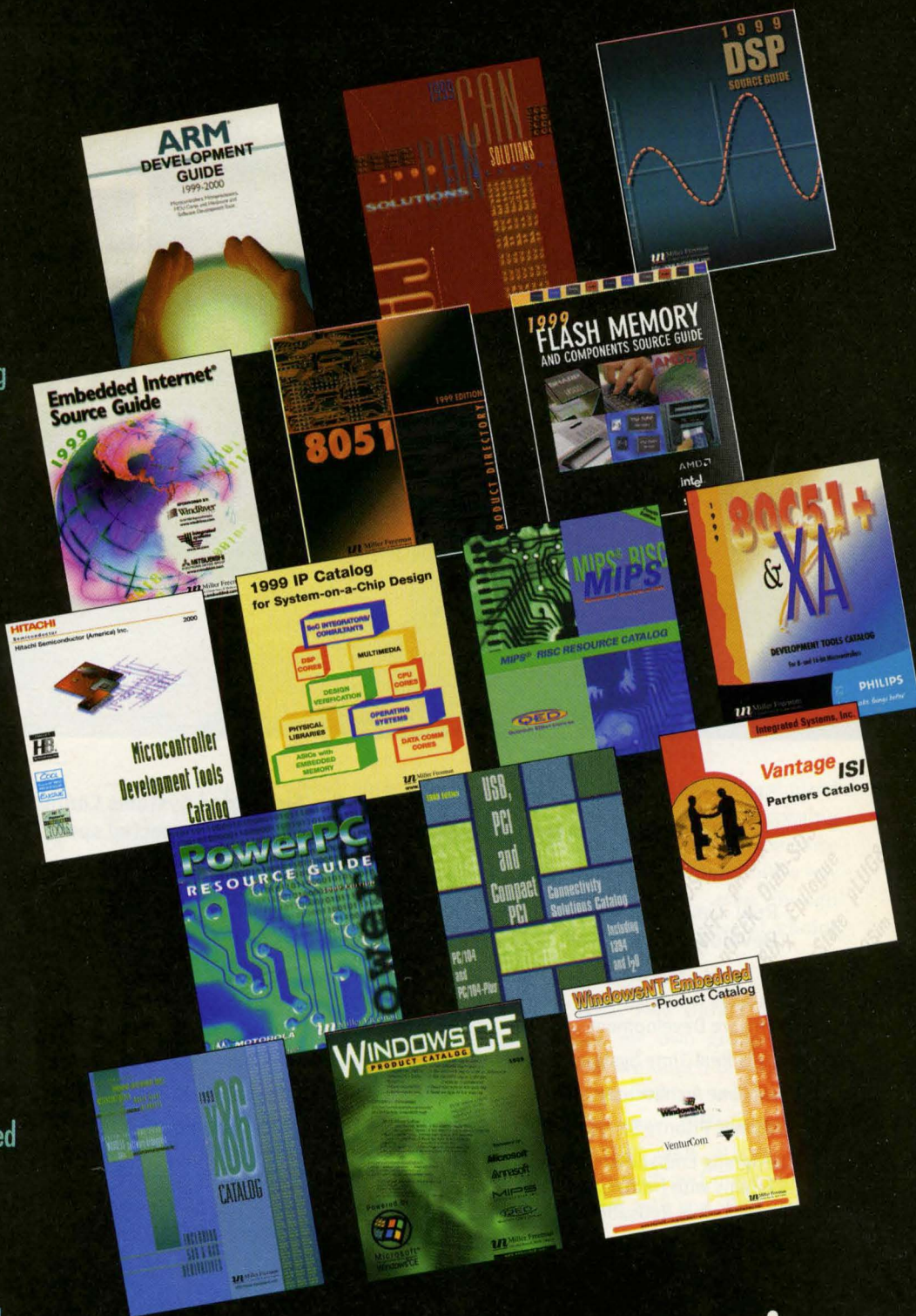
ARM Development Guide
AMD Fusion E86 Catalog
CAN Solutions Directory
DSP Source Guide
Embedded Internet Source Guide
8051 Product Directory
Flash Memory and Components Source Guide
Hitachi Development Tools Catalog
IP Catalog for System-on-a-Chip Design
MIPS RISC Resource Catalog
Philips 80C51 + XA Development Tools
PowerPC Resource Guide
Siemens Microcontroller and DSP Development Tools Guide
USB, PCI and CompactPCI Connectivity Solutions and PC/104 — PC/104-Plus Catalog
Vantage/ISI Partners Catalog
X86 Catalog: including 586 & 686 Derivatives
Windows® CE Product Catalog
Windows® Embedded Catalog

3 easy ways to get your FREE product catalogs:

1. pick them up the Embedded Systems Conferences
2. order online through www.embedded.com
3. call 1-800-500-6815 in the U.S., 1-785-841-1631 outside the U.S.

Check out all the product catalogs at www.embedded.com

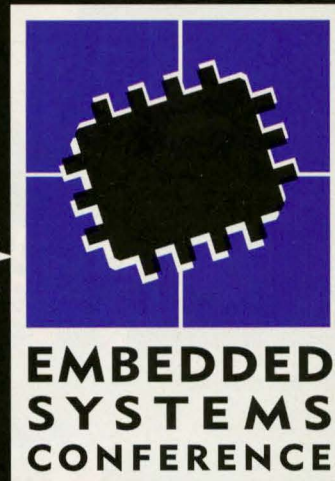
Additional charge applied to multiple orders



CMP
OEM Group



Connect to tomorrow's complete embedded solutions



September 24 – 28, 2000 • San Jose Convention Center • San Jose, CA

Don't miss the most innovative Embedded Systems Conference to date.
More than 390 leading vendors, the industry's most respected speakers, and 69 new classes.

**Real-Time
Development Classes:**

- The Twenty-Five Most Common Mistakes with Real-Time Software Development
- Really Real-Time Systems
- Real-Time Architectures
- Dynamic Priority Scheduling
- Designing Embedded Real-Time Systems with Hardware and Micro-Kernel Reusability
- Performance vs. Portability: POSIX Programming Techniques

**Object-Oriented
Design Classes:**

- System Design: Architectures and Archetypes
- Distributed Software Design Challenges & Solutions
- Modeling Complex Behavior Simply
- Rapid Iterative Development for Embedded Systems
- How to Write Object-Oriented Filters and Encoders
- Key Principles in Refactoring Embedded Kernels
- Architecture Driven Software Design for Embedded Systems

**Internet-Appliance
Design Classes:**

- Creating a Miniature Web Server from Scratch
- Designing an Embedded USB Device Driver
- Communication Protocols for e-Appliances
- Inside Real-Time Kernels
- Utilizing the World's Least Expensive Network Topology
- Internet-Appliance Control Software

Co-Sponsored by:

EmbeddedSystems **EE**TIMES

www.embedded.com/esc • 800-789-2223 • esc@cmp.com

CMP



Jack G. Ganssle

The Story of Language

The night, illuminated only by a universe of stars, swallows the sounds of our movements as we ease through the brush. Coming over a rocky rise we spy a flickering light, a beckoning distant campfire whose promise of friendly warmth draws us closer. Though it's been a long day and a tiring trek wariness still prevails; we approach these unknown humans downwind and quietly lest the need to protect their turf turns into a violent confrontation.

Closer now, we listen to a language that's been unheard for millennium, but through the miracle of fiction we somehow comprehend. An old man of 30, gray-bearded and sickly, relates the history of their people in a mixture of song and epic poetry. He heard it from his father, who passed it from other, now dead, generations. The younger folk listen, enraptured by the story. In the manner of youngsters everywhere they little realize that this is more than an interesting tale. Years will pass before they, too, pass this oral history on to their genetic legacies.

It's 4000 B.C. The printing press lies five thousand years in the future. Writing won't appear for another two millennia. Yet a million years have passed since man became more or less self-aware, capable of learning, a tool-maker who, perhaps in small ways, strives to improve his lot. Though the great explosion of new ideas and revolutionary ways of living lies far off in the future, much has been learned over the generations. Until new technologies come along, humans pass this information along just by talk.

Somehow humankind, though, has

learned the most effective way to package information passed this way: the story. We remember little of a dry recitation of facts, but are enraptured by stories, as their elements weave permanent patterns in our minds.

We time travelers press the fast forward button on the Way Back machine and zoom ahead. Writing appears; some of the earliest written words are the stories formerly passed around the

piece with the expectation of a dry but precise engineering approach to language selection. Instead, he told a story. One that showed both triumph and failure, a high tech version of heroic struggles between the forces of good and evil (if you define "good" as getting a bug-free product out on time!). Beowulf and Barr.

As a vehicle for learning, especially when there probably is no scientific

The key to selecting a language is that it's standardized—or so the story goes.

campfire. Jumping ahead once again we hit the information age. Lo and behold, even now the most reproduced book in the world starts with a recitation of many of those campfire stories. Chapter 1: Genesis. We listen to little children reciting tales of a plague that tortured the world 500 years before—"Ring Around the Rosy."

It appears, despite astonishing new communications methods and channels, new "learning strategies," and developmental fads, that humans learn best from something inalienably simple: stories and storytelling. It's a fact we neglect at our peril.

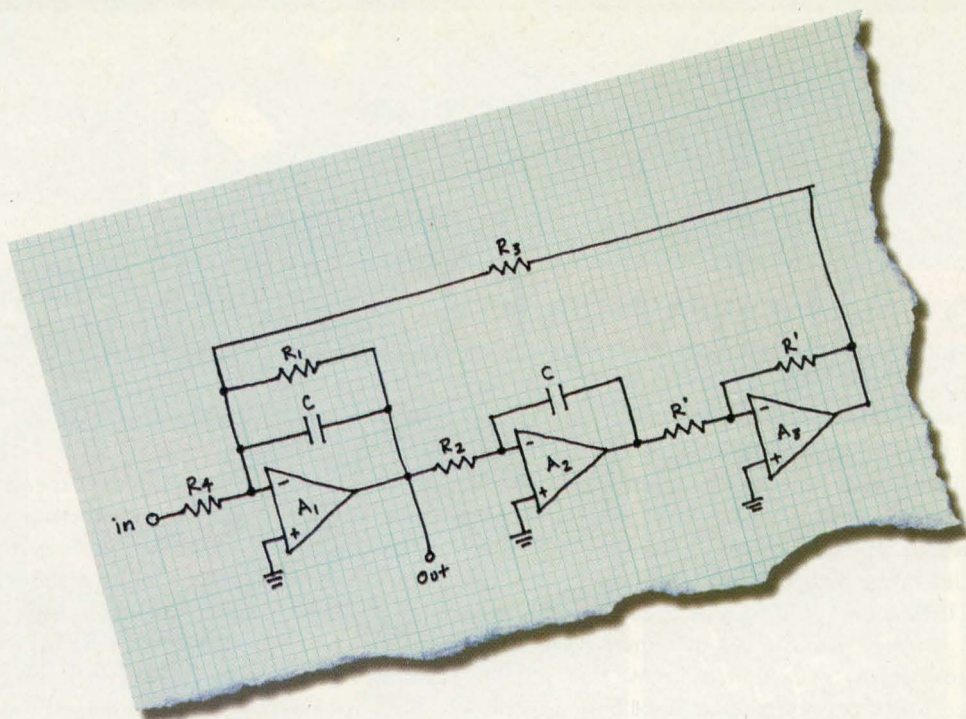
This longwinded introduction is my tale of how I was struck by Michael Barr's article in the March 2000 issue of this magazine about selecting appropriate languages for embedded systems development ("Language Lessons," p. 59). I started reading the

formula to get the most optimized result, the story is king. It's interesting that after 50 years of computer science it seems we still have no definitive way to select a language, even though that's the most fundamental choice we'll make for a project.

Is Ada or Forth the right choice? C or C++? What role should assembly play? I have my own ground rules, though these are honed only through the vicious lessons of bitter experience. Yours may differ. Perhaps, though, I can take a cue from Michael's approach and share some of my own stories with you. Epic they're not. To me, though, they've proven instructional.

A C++ crash

A couple of years ago an instrumentation company asked me to visit their



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 tech-


nology. 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.

site and probe into a project. On the phone the manager sounded in control and confident in the team. The company had a broad range of successful products and, secure in their technology, were working on a new generation device planned to be the basis of future years of products.

Somewhat reluctantly I agreed to the visit, hoping to learn something from how this manager was able to coordinate a decent sized team working on quite complex products. The successes claimed on the phone sounded so good compared to the normal chaos of development! But why did they want to see me?

At the site I found an army of developers working in spacious quarters. Unusually, private offices were the norm, capital equipment budgets were fat, and the manager was a truly enlightened individual dedicated to bringing the best software processes

into the firm. It seemed there was little I could contribute other than admiring comments.

After a couple of hours listening and looking, bits of pain became apparent. Though past generations of products were indeed successful, something seemed awry with the current project. Forty firmware developers were clearly working hard on this new gadget, yet time leaked steadily from the schedule. A year into the project they were already six months late, and getting later at an ever-increasing rate.

Worse, the system had grown beyond anyone's expectation in size despite no scope changes. Bugs crawled from every function. An entire group was dedicated to bug fixes, but, like fighting a hydra, fixing one created three more.

As the day wore on and the manager slowly revealed ever more prob-

lems, his façade of success finally broke and almost tearfully, he asked me what I thought had gone wrong.

The answer, if not clear to him, was obvious to me. Not at all because of superior intellect or experience; rather, I drew on the consultant's secret weapon: theft.

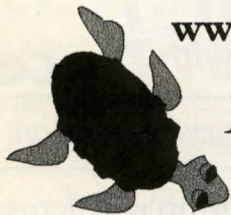
Throughout the day I had had the chance to talk one-on-one with many of the team members. They knew what the problem was. Though some had trouble separating details from meaning, the engineers knew all too well what had happened. I merely stole their thoughts and reformatted them into one idea the manager could understand. Communications problems are at the root of so many troubles, from home life to the office to global political issues.

"The code is junk because the developers are just hacking away," I told the manager. "No one really

Dinkumware, Ltd.

Genuine Software

www.dinkumware.com



Now for Linux

C++ & C Libraries

Dinkum C++ for GCC/Linux
includes Dinkum C Library
Single User License starts at \$90

Dinkum Abridged for WIN/CE
includes Dinkum C Library
Single User License for \$400

C Library

Dinkum C Library for VC++
Single User License for \$100

Single User License
Immediate Delivery From Web Site



*Dinkumware & Dinkum are Registered Trademarks of Dinkumware, Ltd.
Windows is a Registered Trademark of Microsoft Corporation.

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

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



understands the language, and so they build things almost randomly in the hope that things will work."

I went to the doctor once after reading a newspaper article about a disease that seemed to match my minor symptoms. After telling her about the article the doc rolled her eyes and admonished me against playing amateur medical person. In fact, nothing was wrong; my utter lack of knowledge in this field meant I was ripe to believe the overly simplified description of complex issues jammed into three paragraphs of *USA Today*.

Similarly, the president of this company had read a *Wall Street Journal* article describing the benefits of object oriented programming. The promise seemed too good to be true: OOP meant, or so this techno-novice read, that all code would become reusable. I credit the president with identifying

that reusable code could solve many problems. Unhappily, just as with my experience at the doctor's office, the *Journal's* piece couldn't do justice to the complexities of software development. This layperson learned just enough to be dangerous.

He issued an edict: from now on, we're doing all projects in C++! He blithely went on about his normal management, sales, and accounting functions, little realizing the terrible implications of such a far-reaching rule.

Is this a rant against C++? Not at all. A year and 100,000 lines of junk code into the project it became apparent that the rule was bad. The edict meant 40 developers were suddenly plunged into an environment about which they knew nothing.

For it turns out, when this project started, only one developer, the only one fresh out of college, had even the

slightest exposure to C++ and OOP. Thirty-nine others were learning on the job, cranking code, and making mistakes on a mission-critical project, as they tried to grapple with the very different philosophies of OOP. Working with objects means thinking out the design of a product in a very different way than most C designs. C++ is an approach, as well as a language. Trying to have the developers build an OOPy design as they learned the concepts was a bit like designing the space shuttle's orbital dynamics as the engineers learned calculus.

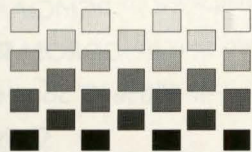
The result was a mad mishmash of conventional procedural code mixed with shoehorned objects, a design that exploited the worst of both kinds of programming. And so I recommended they trash the code and start over. In C. All 40 developers knew C, all 40 were experts at building systems using a C framework.

32/64-Bit CPU Development Boards

- ARM
- MIPS
- Power PC
- M-Core
- PCI

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.

CMA COGENT MODULAR ARCHITECTURE



COGENT

Cogent Computer Systems, Inc.
Tel: 508-278-9400 • www.cogcomp.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

The problem wasn't the language but an arbitrary edict that neglected a critical part of the success of any embedded systems: having expert people. Though the president's wish for reusability was right on, his rule was as disastrous as telling a newspaper reporter to write in Latin. Without people skilled in the development environment you're doomed.

And never expect skills to appear magically overnight.

So to this day I hate having told them to toss out the code, though it was the right choice. I made the same recommendation as did Michael in his story, which is unhappily the universal refrain of consultants around the world. "Toss out the code—it's crap. We'll start over." Firmware is so terribly expensive! But it's impossible to retrofit quality into lousy code. Maybe, with enough effort, we can get rid of most of the bugs in a poorly written system, but maintenance becomes impossible. And crummy code always has nasty, lurking bugs that just seem unconquerable.

So the moral of this unhappy tale is that people's skills are far more important to the success of the project than any technology. No tool, no technology, no "new new thing" will save us from the realities of development woes. Skills first, technology second.

They never die

When Intel invented the microprocessor in 1971, I was working as an electronics technician at a small instrumentation company here in Maryland, putting in too many hours, to support myself while in college. Within a year of releasing their 4004, the wizards at Intel came out with the 8008, a truly useful microprocessor that, in my opinion, started the true microprocessor age.

The engineering manager at my company recognized the value of this part to our products and started designing a new product around the part. The problem: the engineers

Rent the Hottest List around!

Embedded Systems PROGRAMMING

Over 55,000 Qualified Subscribers



Call Rubin Response 847-619-9800



Rubin Response Management Services, Inc.

1111 Plaza Drive, Suite 800, Schaumburg, IL 60173

(847) 619-9800, Fax (847) 619-0149, kristen@rubinresponse.com

request a quote on-line at www.rubinresponse.com

there didn't know how to program computers. Why should they? In the early '70s computers were still mostly inaccessible beasts used more by the IT folks than poor EEs. Somehow they learned that I knew assembly language—for a Univac mainframe—and promoted me to the awesome status of engineer. My job was to write the 4K firmware package for the product. Never has so little been known by so many, never has so much on-the-job training happened in such a short period of time.

We eventually did manage to ship a number of these units, which were successful enough that customers demanded more features, more than could fit in the 8008's 16K address space.

In 1975, the 8080 has hit the market. Through some quirk, I'm running the greatly expanded micro group, and we redesign the instrument around this new processor. It, too, ships.

We designed the product using another Intel invention, the EPROM. They guarantee data retention of 10 years, which of course seemed near to infinity to our very young development team.

A decade passes and, as is always the case, individuals disperse even though the company continues. The product is obsolete, but customers still have them working, measuring oil content on polyester fibers in a factory environment. Ten years to the month after shipping the first of these units, they start dropping bits in the field.

The repair crew at the company discover that all of the original object files exist on the only media of the mid '70s: paper tape. No one has a clue how even to boot up the Intellec 8 "development system," let alone load the tapes and reprogram the

EPROMs. Self-employed now, I'm surprised by a panicked phone call from them. Plumbing the dim recesses of memory I'm somehow able to remember how to use the tools, reload the now-brittle paper tapes, and return the EPROMs.

By the mid '90s I expected another call as another decade sailed by, figuring the memories would once again need refreshing. None came; presumably the instrument had finally been retired. But I learned a critical lesson from this experience: embedded systems never seem to die. They run quietly in the background for years. If we don't plan ahead for maintenance that may literally span generations we're fools. And surely the Y2K fiasco should have taught us the same lesson about software in general.

Selection criteria

I won't even try to tell you how to select your next embedded lingo. However, from the lessons above, it's clear to me that we simply must select one that has been standardized. Prior to the C's ANSI standard, dozens of variants competed for attention. Portability was impossible. In my opinion, C++ has only recently become a viable embedded choice, since its November 1998 ANSI standard. We know that both choices are safe, that a C or C++ programmer 10 years from now will be able to work on our ANSI-compliant code.

At the risk of offending lots of people, I don't see Java as a serious embedded contender just yet, unless your system will go away in a year or two. Java is a language in flux, torn asunder by marketing forces of giants. It's coupled, for better or worse, to the two biggest forces of change in the universe: the PC and the Internet. I suspect that eventually

it will indeed be standardized, and in fact may even supplant C++ due to its simpler, less overwhelming, syntax. But for now, be assured that the Java we write today will look very different in a few years.

The exception to standards (because there's always an exception in embedded work) is assembly language. Fact is, this is different for every processor, is inherently non-portable, and will always be a tough way to write code. Though I do love it so! Assembly makes sense only for time or space-constrained apps, or for small parts of a bigger project where speed is truly an issue.

Moral of the story of language

When my son was very young I'd end each evening with a made-up story. Most were silly things meant to elicit a laugh. When traveling I'd often write a short bit of nonsense that his mom would read in my absence. Now, just barely a teenager, he seems to have forgotten all of the rules I've so carefully tried to instill—but does remember these stories even long after my failing middle-aged brain hasn't the slimmest recollection.

Stories seem to fit the nature of the way the human brain works, much more efficiently than dreary facts. Urban legends spread like wildfire as arithmetic skills collapse. And so, another moral for us all is to learn to frame important ideas in the context of stories. For only those will be remembered.

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.

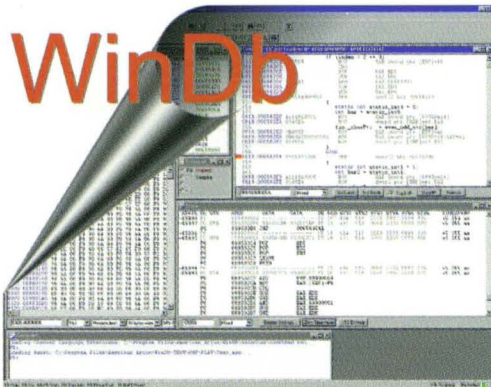
EMBEDDED SYSTEMS PROGRAMMING (ISSN 1040-3272) is published monthly, with an additional issue published in September, by CMP Media, 525 Market St., Ste. 500, San Francisco, CA 94105, (415) 905-2200. Please direct advertising and editorial inquiries to this address. SUBSCRIPTION RATE for the United States 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 1278, Skokie, IL 60076-8278. For customer service, telephone toll-free (888) 847-6177. 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, CMP Media. 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.



It's hard to compete without the right tools.

As with most things, there's a hard way and an easy way to develop any new Intel processor based system. And just for the record, we're the guys who offer the easy way.

We've been working closely with Intel for 9 years to make sure that powerful new American Arium development tools are ready and waiting each time early silicon is released. So whether you're developing a Pentium® processor based embedded system or a server designed around the new Itanium™ processor,



we've got the affordable and readily upgradeable in-circuit emulator you need. Features include:

- New! GNU support
- Unmatched breakpoints
- Industry leading debugger
- Fast setup and use
- 1 year of software upgrades and tech support

Why not call us today and put your system on the fast track to market.



14811 Myford Rd., Tustin, CA 92780
Ph: 714-731-1661 Fax: 714-731-6344
E-mail: info@arium.com

For more information on any of our in-circuit emulators, visit...

www.arium.com

Hardware
and
Software
Tools
for:
8051
68HC11
Z80
Z180
68HC12
Mcore
683xx
PowerPC
68K
x86
68HC05
ColdFire
And
Many
More

Power Tools for Embedded Systems Development



HMI-200 In-Circuit Emulator

The HMI-200 Pro Tool is one of the most popular in-circuit emulator lines ever made: dual trace buffers, overlay memory, complex break and trigger point capability, and built-in hardware-based software performance analysis are just a few of its powerful features. We'll build an HMI-200 for you with the options you need at a price you can afford. Support for 8051, 68HC16, 68K, Z80, 68302, CPU32, Z180/64180, 68306, CPU32+, 8085, 68307, 68020, 68HC11, 8096, 68030, 6809, 68040

Use Avocet tools and relax. With over 20 years of development expertise behind our hardware and software we are the wise choice for your embedded projects. One phone call to Avocet and you'll find the customer service you want, the tools you need, and the technical support you expect.



Background Mode Debugger

Harness the incredible speed and versatility of the USB interface with our USB Background Mode Debugger. Our BDM solution is one of the industry's most popular development tools, and with good reason: easily single-step code at the source or assembly level, manipulate CPU and peripheral registers, and program target Flash devices using our full-featured SourceGate source-level debugger. If you're looking for a Background Mode Debugger our USB-BMD is the obvious choice.

C Compilers
Assemblers
Simulators

In-Circuit Emulators

BDMs

RTOS

AVOCET
SYSTEMS, INC

(800) 448-8500

www.avocetsystems.com
sales@avocetsystems.com