

Embedded Systems[®]

P R O G R A M M I N G

A Record/Playback Scheme for Software Test



Building a Generic
Protocol Engine

Timing Tricks for **PICs**

A **Test** for Embedded
Programmers



Internet Appliance Design:

A Simpler Approach to Web Servers
More on SNMP

Cut Development Time

Using In-Circuit Emulators and BDM's



NEW
seeHau
User Interface

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[™] 

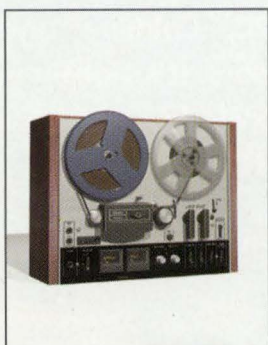
EST

Embedded Support
Tools Corporation

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

contents

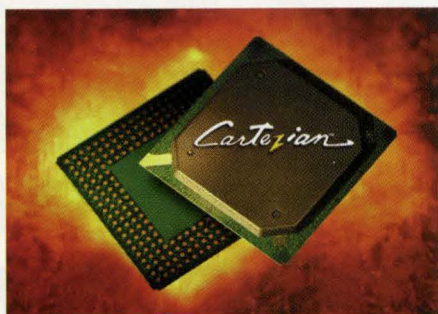
MAY 2000



COVER

A record/playback scheme for testing in complex environments can be "reely" effective.

Cover illustration by Rupert Adley.



135

ZILOG's eZ80 Cartesian Communications Engine features a synthesizable 32-bit RISC/DSP co-processor.

Cover Story

32

Playback: Reality-Based System Testing

A record-and-playback test methodology lets you exercise your system with realistic test data. But making a system playback-ready offers some challenges of its own.

BY WES HOWL

90

A Generic Protocol Engine for Synchronous Protocols

Despite the diversity of communications protocols, there are many similarities among them. You can build one protocol engine to handle the commonality and then adjust it to the needs of each project.

BY ANDREY JIVSOV

108

Timing Tricks for PIC

The upside of using a microcontroller with single-cycle instructions is the ease and precision with which your code can synchronize its timing. This article presents five timing tricks that will help you do just that.

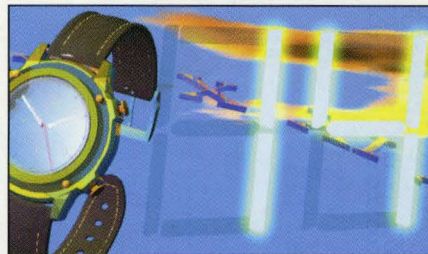
BY ROBERT SCOTT

119

A 'C' Test: The 0x10 Best Questions for Would-be Embedded Programmers

Pencils up, everyone. Here's a test to identify programmers with the potential to enter the arcane world of embedded systems.

BY NIGEL JONES



Featured Section

internet appliance design

53 **CONNECTING...** Hold Everything

Michael Barr puts off his planned series of articles on UDP/IP as he searches for the tools he'll need to accomplish the task. In the meantime, he discusses calibration and tweaking.

BY MICHAEL BARR

61 Web by Proxy

Adding a TCP/IP and web server stack to an embedded system is an expensive proposition. If your product already communicates, it may be better and cheaper to use a proxy.

BY BILL GATLIFF

Part 2 in a Series

69 Exposing MIB Data to a Web-based Interface, Part 2

This month we conclude this discussion of how web-based management can benefit from the right architecture and an SNMP MIB inheritance library.

BY KEDRON WOLCOTT

83 Real-Time Java War Yields Uneasy Truce

Real-time Java is not an oxymoron, according to the two groups presently developing competing sets of specifications.

BY ALEXANDER WOLFE

89 Embedded Internet Tools

New internet appliance design products.

108

Learn some timing tricks for using PICs.

departments

5 **#INCLUDE** Discontinuity Bites Back BY LINDSEY VEREEN

7 **PARITY BIT**

11 **NEWS VECTORS**

135 **NEW PRODUCT GALLERY**

137 **RECRUITMENT**

140 **MARKETPLACE**

144 **ADVERTISER INDEX**

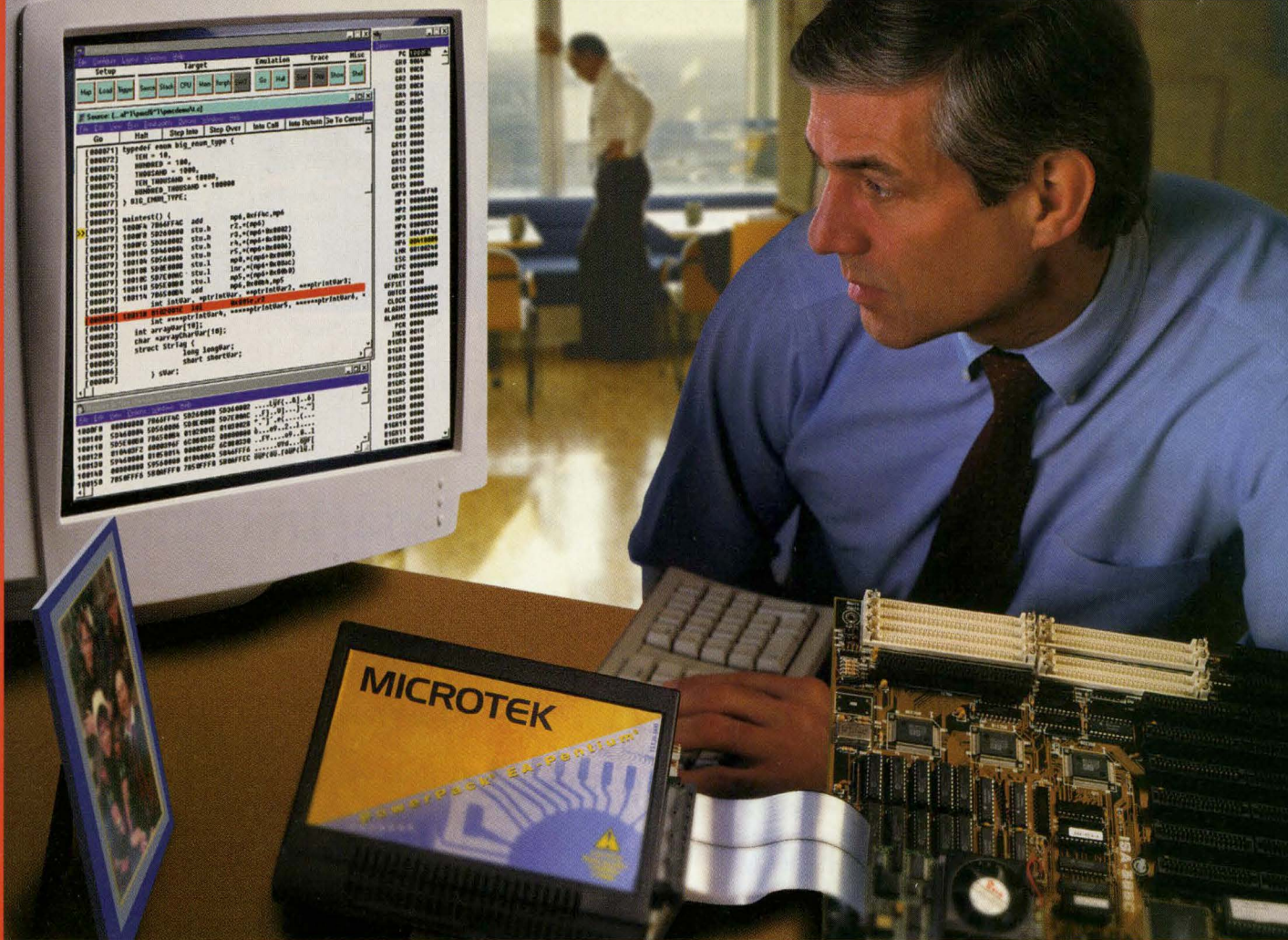
columns

15 **PROGRAMMER'S TOOLBOX** Old Kids on the Block BY JACK W. CRENSHAW

131 **SPECTRA** Music and Noise BY DON MORGAN

145 **BREAK POINTS** ESC Chicago BY JACK G. GANSSLE

151 **STATE OF THE ART** Quitting Time BY P.J. PLAUGER



“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

CONSULTING TECHNICAL EDITORS

Jack G. Ganssle
Jerome L. Krasner, PhD

CONTRIBUTING EDITORS

Jack W. Crenshaw
Larry Mitting
Don Morgan
P.J. Plauger
Dan Saks

VICE PRESIDENT/ELECTRONICS

Donna J. Esposito

EMBEDDED/DSP GROUP DIRECTOR

Mike Flynn, (415) 278-5251

PUBLISHER

Eric Berg, (415) 278-5220

EASTERN REGIONAL SALES MANAGER

Damon Graff, (781) 487-7585

EASTERN SALES REPRESENTATIVE

Jared Grimm, (781) 487-7586

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

esp@omeda.com

Back issues may be purchased on a prepaid basis through: Miller Freeman, 1601 West 23rd St., Suite 200, Lawrence, KS 66046; (800) 444-4881; (785) 841-1631

REPRINTS

Sherry Bloom, (415) 808-3980

ONLINE PRODUCTION COORDINATOR

Billy Biondi, wbiondi@cmp.com

PRESIDENT/CEO, CMP MEDIA INC.

Gary Marshall

EXECUTIVE VICE PRESIDENTS

Regina Starr Ridley John Russell
Steve Weitzner Tony Uphoff

**SENIOR VICE PRESIDENT/
GLOBAL SALES & MARKETING**

Bill Howard

**SENIOR VICE PRESIDENT/
BUSINESS DEVELOPMENT**

Pam Watkins

PRESIDENT/ELECTRONICS

Steve Weitzner

Discontinuity Bites Back

I recently suffered a dental discontinuity: my dentist opted for full-time motherhood. She sold her practice, passed my records on to her successor, and headed out to the suburbs. Although she made my migration path to the new dentist as easy as possible, I nevertheless took the opportunity to do some reconnoitering, dentistry-wise, and ended up with a new, even better practitioner. There may have been nothing wrong with her successor, but at the same time I had no compelling reason to go passively with that particular flow.

In business as well as in one's personal life, we sometimes resist following the strategies that others have laid out for us. As soon as one of our suppliers introduces an element of incompatibility in its product upgrade path that requires any re-engineering, we are tempted to reassess our options.

Users of ISI's pSOS real-time operating system will sooner or later have a choice of their own to make. One of the big stories at the Embedded Systems Conference in Chicago was Wind River's announcement that it was going to 86 pSOS. Everyone had been speculating on what Wind River's plans would be vis-à-vis this second operating system, and several pundits predicted that pSOS would be made redundant. The plan, according to Wind River chairman and co-founder Jerry Fiddler, is to converge VxWorks and pSOS so that pSOS users have a relatively painless upgrade path via a compatible API.

It's not so much that there is anything wrong with the prescribed migration path that Wind River is offering, but a couple of factors mitigate against customers embracing it universally. For one thing, the emotional factor comes into play: some pSOS users may not want to join the Wind River crowd. This is just

human nature and nothing against Wind River. And while it makes no sense to abandon something willy-nilly in which you've invested heavily, if there's any cost at all associated with the migration to the next-generation product, then customers are liable to go through an evaluation process. I suspect that Wind River will have its work cut out for it trying to keep its combined customer base intact as it makes the transition to a converged platform.

Fiddler has already addressed the issue of discontinuity from the perspective of how Wind River will benefit from it. As the post-PC era gains momentum, the traditional leaders such as Microsoft and Intel will lose the advantage of legacy that has kept their momentum going. Other companies will benefit from the shift from desktop systems to Internet-enabled appliances, and Fiddler fully intends that Wind River will be one of them.

With the convergence of pSOS and VxWorks, Wind River seems to have created a discontinuity of its own. Granted it makes no sense to spend precious engineering resources supporting two parallel product paths, but no matter how smoothly you try to make the transition, it's never going to be as easy as falling off the proverbial deceased tree. Other RTOS vendors are busily putting plans into place to capture defectors. Wind River may soon discover that discontinuity cuts both ways.

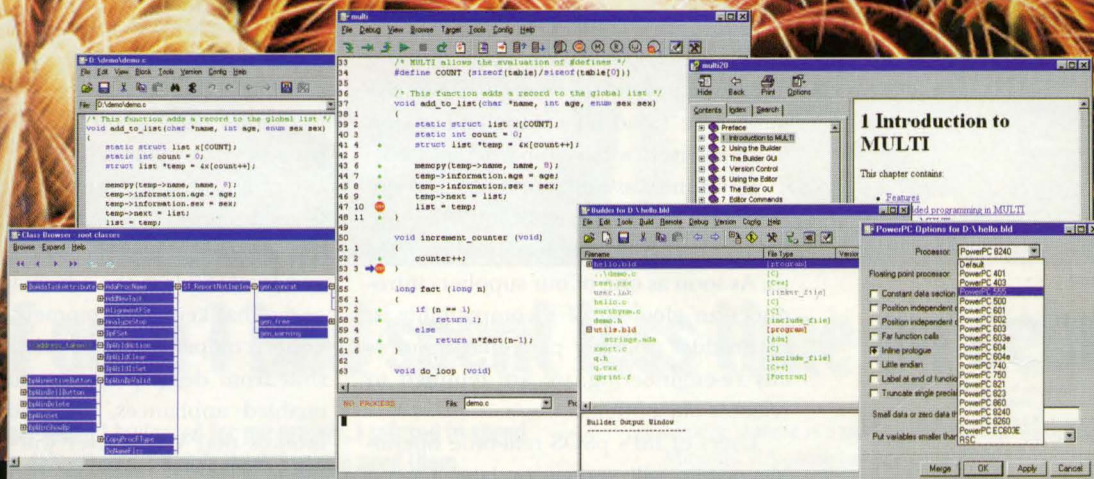
lvereen@cmp.com



Visit our Web site at
www.embedded.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!

Green Hills
• SOFTWARE, INC. •

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.



Two more cents on UML

I read with interest the Unified Modeling Language point/counter-point discussion between Steve Mellor and Bran Selic ("Modeling Complex Behavior Simply" and "How to Simplify Complexity," March 2000, p. 37). As one who is actively involved in the development and direction of the UML, as well as someone who knows both Steve and Bran personally, I'd like to throw my own two cents into the ring in an effort to clarify what I think is the essential difference between these two approaches.

Steve's position, as I understand it, is that objects should be behaviorally as simple as absolutely possible. To this end, Steve believes that mixing entry and exit actions on the same state machine is an inherently terrible idea, let alone nested or (God forbid!) and-states, history pseudostates, forks, and joins.

Bran, on the other hand, is of the opinion that using the various capabilities provided by statecharts allows a more parsimonious and scaleable description of behavior for objects, and that such statecharts are easily understandable and so Steve's concern is overstated.

I think problems to be solved have an apparent complexity that is the sum of the essential complexity and the incidental complexity. The essential complexity is the minimum complexity required to solve the problem regardless of the approach and regardless of where in the model it occurs. Incidental complexity is added due to how the problem was solved. Slavish adherence to a single approach can add significant incidental complexity. For example, Steve's approach adds more objects to the collaboration, each of which has a very simple statemachine. The essential complexity,

though, hasn't been reduced at all—it has just moved from the individual objects to a much more complex collaboration. And in so doing, the incidental complexity may have risen significantly because of the complexity of how the collaboration works.

There are certainly times when simplifying the individual objects by complicating the collaboration minimizes the overall apparent complexity. And there are times when the reverse is true—making the objects slightly more complex greatly simplifies the collaboration. It depends on the nature of the collaboration and the nature of the object behavior. Statecharts have a rich semantics that allow the designer to make their own trade-offs based on their experience and on the nature of the problem they face. And, frankly speaking, I don't think the use of actions on transitions (when actions are executed only on some paths into a state), actions on state entry (when an action must always be executed when a state is assumed), or actions on state exit (when an action must always be executed when a state is left) confuses anyone or overly complicates the state behavior. In fact, the statechart symbology is rich and parsimonious, which allows it to be expressive without adding inordinate complexity or difficulty in understandability.

The big advantage that any graphical language brings to the table is the ability to view a system at several different levels of abstraction. Nested and and-states in statecharts are perfect for viewing the state behavior at different levels of abstraction. In addition, when a transition must take place from many states to a single state (such as often happens in error handling), it is a simple matter to enclose

all the relevant states within a single super state. In a flat statemachine, some transitions must be drawn from each of the relevant substates individually. This adds complexity rather than diminishing it.

As with all design approaches, your mileage may vary. Software developers deal with different problems and different environments and come to problem solving with different experiences and different approaches. And all this diversity is a good thing because it places many more tools at our disposal to solve the wide range of problems that we, collectively, face.

Bruce Powel Douglass, Chief Evangelist
I-LOGIX

OSEK update

In my article, "OSEK/VDX Network Manager and Implementation Language" (April 2000, p. 96), I wrote in the summary that the direct network management method was limited in applicability due to the requirement that only eight messages are defined for network management and the inherent latency due to this limitation. What was unclear to me in the NM specification and has since been clarified is that the network management messages are not the only messages on the bus. Normal communication messages are also sent using the Communication Standard. The NM messages are the lowest priority messages on the bus and therefore do not interfere with normal communication.

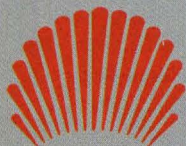
I apologize for any confusion that this has created. In fact, Direct NM is the primary form of NM used in Europe at this time.

Joe Lemieux
joe.lemieux@eds.com




FUJITSU

HITACHI



華為技術
HUAWEI

 **Italtel**

NEC

OKI


3Com

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



DigitalDNA[™]
from Motorola

THE **HEART** OF SMART.[™]

If you get excited just thinking about the possibilities for networking and computing, you're our kind of designer. And we're happy to connect you with all kinds of smart solutions. DigitalDNA technology from Motorola is more than chips. It's software, systems and the ideas of thousands of innovative engineers dedicated to helping create the next generation of routers, switches, servers, modems, desktops and more. You'll find DigitalDNA 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 whenever you need a smart networking solution, you know the name to turn to. www.digitaldna.motorola.com

POWERPC[™] • FSRAM • COMMUNICATIONS PROTOCOL MODULES • DSP • LDMOS/RF • COLDFIRE[®]

M O T O R O L A E M B E D D E D S O L U T I O N S

✓ *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.





WRS to converge VxWorks and pSOS

Following its merger with Integrated Systems Inc. (ISI), Wind River Systems Inc. has announced that it will converge its product line into a single development and operating system. Over the next couple of years, Wind River says it will phase out pSOS, but users of both VxWorks and pSOS will continue to receive additional releases.

Wind River announced that the next version of its Tornado IDE and VxWorks RTOS will be called Cirrus and will support Linux and all other popular host systems. The next release of the pRISM IDE and pSOS RTOS will be called Stratus. Both releases will be available by third-quarter 2000. Wind River will then converge to one RTOS and IDE called Cumulus, available in 2001. Cumulus will include all features of Cirrus and support the pSOS API and other features unique to pSOS. Wind River says it has also created a migration team for customers that need technical help.

"Their strategy of absorbing the best of ISI into Wind River has resulted in their having had to abandon pSOS, thereby creating an opportunity for competing RTOS vendors at the deeply embedded level," said Dr. Jerry Krasner, director and research editor for Electronics Market Forecasters.

ARC acquires Precise and VAutomation

ARC Cores will acquire Precise Software Technologies Inc. and VAutomation Inc. Precise Software is an Ottawa-based embedded Internet protocol and real-time operating systems company and VAutomation makes serial communications and peripheral cores. Terms were not disclosed for either transaction. Operations for both acquired companies are expected to remain at their current sites.

Briefly Noted...

Alcatel and **STMicroelectronics** have entered into an agreement that will allow Alcatel to adopt the STMicroelectronics ST100 DSP core for use in a variety of system-on-chip solutions. * **Enea OSE Systems** has announced that its OSE real-time operating system is available for the MIPS R3000. * **Green Hills Software** has introduced an optimizing C/C++ compiler for the PowerPC to support **Motorola's** AltiVec technology. * **Applied Microsystems Corp.** has joined the PowerPlay Initiative which aims to help create open standards for mul-

ti-player games on the Internet. * **NEC Electronics** has created an Automotive Strategic Business Unit. The unit will focus on the automotive industry and will take advantage of U.S.-based manufacturing, design, marketing, and application expertise to serve the North American automotive market. * **Motorola Computer Group** has launched HA Linux, its first Linux offering for applications that require 99.999% availability (5NINES). * **I-Bus Inc.** has merged with Phoenix Power to create I-Bus/Phoenix Power and Computing Systems. * **Wind River Systems** has agreed to acquire **Embedded Support Tools Corp.** for 6.3 million shares of Wind River stock. * Wind River and **Tensilica** have announced a real-time operating system and integrated development environment for a configurable processor. * **ZILOG** has introduced a licensing program for its eZ80 Internet Engine. * **Xtech Embedded Computers** has changed its name to Teknor Applicom Inc. Xtech recently completed the acquisition of Teknor Industrial Computers Inc. * **Lineo** has begun shipping Lineo Embedix Linux 1.0 for the x86 and PowerPC. * **Mitsubishi Electronics America** has sampled the M37516, its 8-bit microcontroller with 32K of on-chip flash memory and SMBus interface. * The Hard Hat Linux operating system, developed by **MontaVista Software**, will be used in the first stand-alone Internet radio. **Kerbando** developed the radio to access Web-based streaming audio without a PC. * **CodeGen** has made its BIOS firmware product, SmartFirmware, available on all Cogent Computer Systems' development platforms. * **Analog Devices** has been named the fastest growing supplier in the DSP market, by Forward Concepts, a DSP technology market research firm. * **Tektronix Inc.** has added **Dragonfly Software Development LLC** and **SynapticAD** to its Embedded Systems Tools Partners Program. * U.K.-based **Embedded Solutions Ltd.** has opened an office in the Silicon Valley in Campbell, CA. * **ITCN** has been awarded a Small Business Innovative Research (SBIR) Phase II contract from the U.S. Navy.

NAMES IN THE NEWS

JOSEPH BOREL was awarded one of the IEEE's Third Millennium Medals. Borel recently retired as STMicroelectronics' director of design automation and integrated systems for the company's central R&D group. ● STMicroelectronics has appointed **ENRICO VILLA** corporate vice president, director of the European region. ● Ampro Computers has named **PAUL ROSENFELD** vice president of marketing. ● **ANDREI MOLDOVEANU** has been named director of marketing for Nematron Corp. ● Lineo Inc. has named **KIM D. CLARK** vice president of engineering; **ALLAN SMART**, vice president of the Professional Services Division; and **PAUL CAMERON** director of engineering. ● Intrinsix Corp. has named **H. KENT BOWEN** of Harvard University and **WALLACE (WALLY) A. CATALDO**, a financial expert, to its board of directors.

DAvE's Back!

DAvE 2.0 – The Digital Application virtual Engineer from Infineon Technologies.

DAvE Who?

DAvE stands for Digital Application virtual Engineer and is Infineon Technologies' code generator for their range of 8-, 16- and 32-bit Micro-controllers. It provides initialization, configuration and driver code to ease programming for beginners as well as experts. DAvE comes with an easy to use windows user Interface running on Windows 95/98 and Windows NT. DAvE is now available in version 2.0 free of charge from Infineon Technologies.

Working with DAvE means that you can select your controller, click on the peripherals of your controller, select the functionality of your controller and tell DAvE to generate the appropriate code for you. The only work that remains for you to do is to write the application itself. Even then, DAvE will generate detailed documentation for your project.

The tooltip help provides detailed help on every disabled control – even giving reasons for why it's disabled. This will increase the ease of system configuration and allows developers to understand the controller in it's full functionality. Having multiple dialog boxes that can be kept open at the same time provides easy visual display of dependencies between various peripherals of a controller.

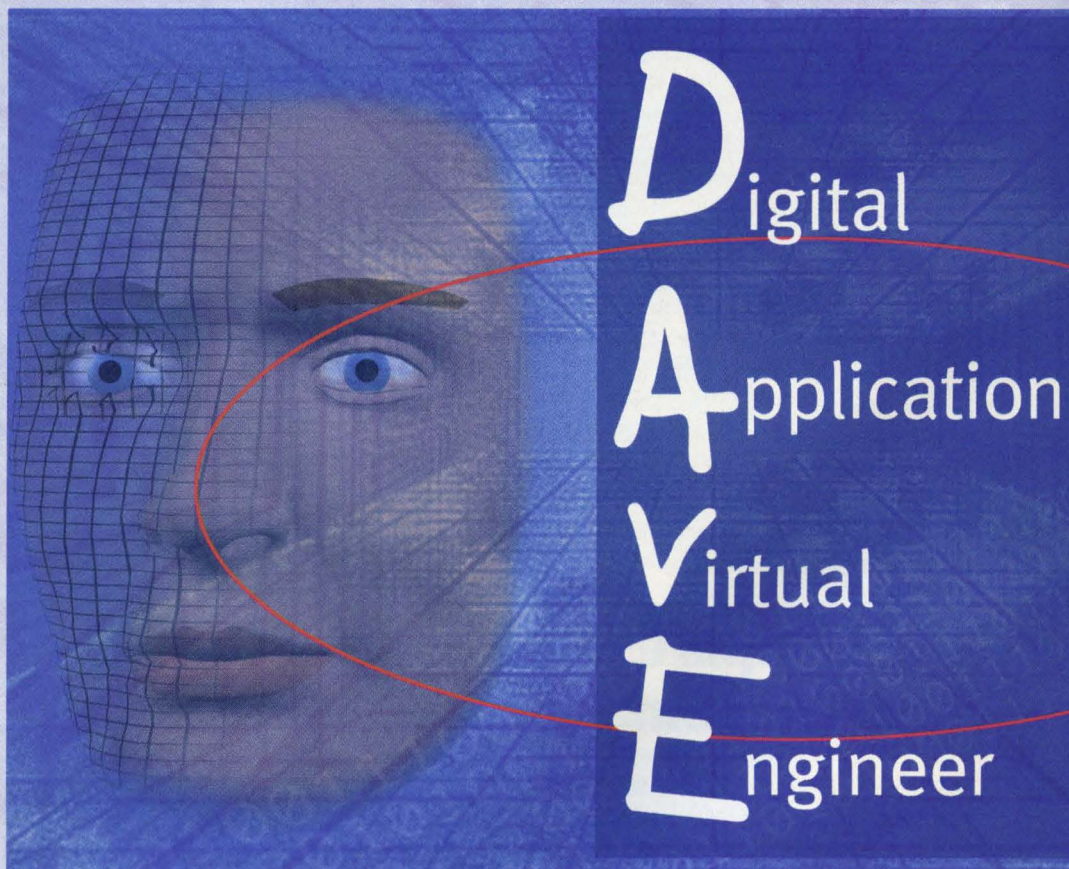
A register explorer also allows you to follow up changes of the configuration

in the settings of registers. The explorer gives one-click access to the complete range of registers within the controller that the engineer is working with. The register-view provides information on the register address, the reset value, as well as the current value based on the configuration.

(Visual Basic for Applications) like Script language

- DAvE comes with SDK (Software Development Kit) toolchain, including Dialog editor, Template generator, Setup wizard, Deployment wizard and complete documentation. The SDK supports the Add-In interface as well as

- New Setup that allows you to install DAvE 2.0 with only the necessary components completely on the hard disk
- Setup includes automatic web update for components
- Add-in interface that allows complete access to the DAvE database from inte-



What's new in DAvE 2.0?

- The entirely new program source code enables DAvE 2.0 to follow new and innovative object-orientated concepts
- The new MDI (Multiple Document Interface) User interface allows all configuration masks to be open at the same time
- It's programmable via VBA

the derivative implementation interface

- Software Interface for 3rd party tool integration
- New register viewer, scalable window showing register address, reset value, current value (due to configuration) as well as highlighted bits affected by the last change in settings
- New File Explorer

grated components and features seamless integration of additional capabilities

- Support of more than 20 Infineon derivatives
- Support of the new 32-bit TriCore architecture
- Improved help system
- Enhanced code generation
- Improved error handling. Possible problems are shown

in the documentation file that is generated by DAVe.

Every controller that is supported by DavE 2.0 comes with the complete original documentation and DAVe 2.0 also offers direct links to related topics in the manual. Inside each configuration mask, any additional entries to the

latest configuration. The software interface is already supported by many Infineon tool partners such as Keil, Tasking, Greenhills, Hitex, pls, nohau, Lauterbach and Appliware.

The DAVe SDK – Develop your own Add-Ins

DAVe 2.0 includes a complete Software Development Kit (SDK). This allows the developer to implement his own Add-Ins, as well as derivatives accessing the complete database of DAVe – includes manuals and SFR definitions.

Implementations are based on a VBA-like scripting language. The SDK comes with a dialog editor for custom DAVe controls as well as a template generator that develops the complete Add-In framework for the developer. A packaging wizard wraps the final product into a ready-to-go setup. The SDK is fully documented and an API reference is also additionally provided.

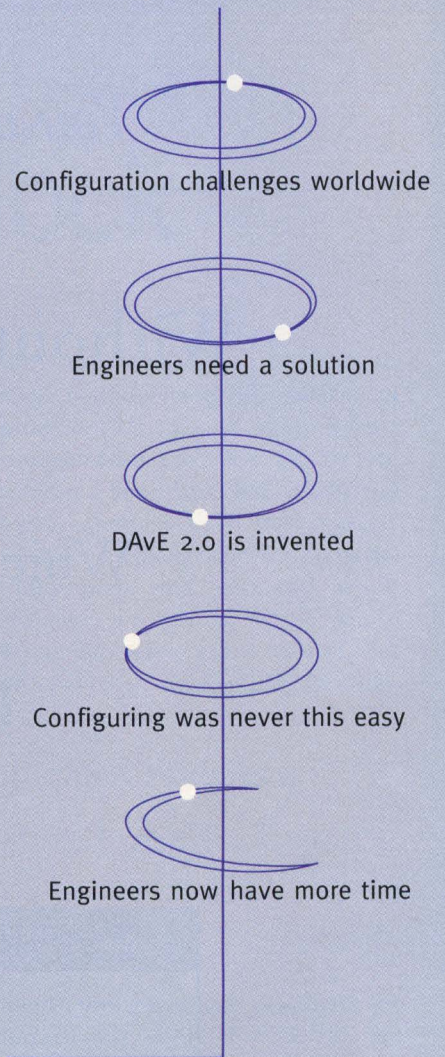
DAVe's On-Line

The amount of available derivatives for DAVe are constantly growing. To keep DAVe up-to-date, Infineon keeps the latest implementations on the DAVe website available for download. Go to www.infineon.com/dave to get the latest product information, a FAQ list, updates, news, new Add-Ins and discussions on DAVe.

The website also provides an ordering number for DAVe. Just write an eMail or FAX to obtain DAVe. For comments, suggestions and questions write to: DAVe@infineon.com

{PERPETUAL THINKING PROCESS}

INFINEON CYCLE



DAVe will help you compare and evaluate the different members of Infineon's 8-, 16- and 32-bit families of microcontrollers and help you find the right chip for your embedded control application.

www.infineon.com/dave

version 2

©2000
Infineon
Technologies

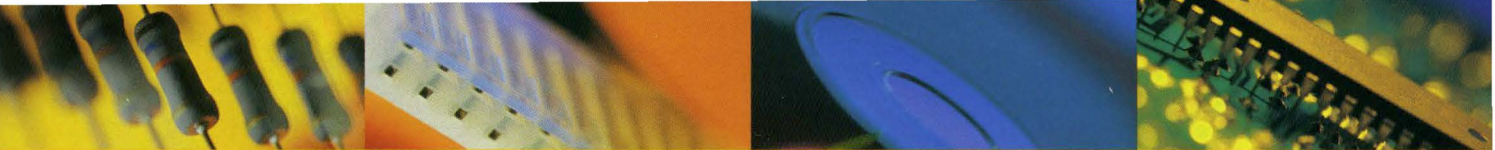
manual can be selected that are directly related to the selected mask settings.

The Tool Connection

DavE 2.0 offers a software interface allowing tool partners to directly communicate with DAVe and its database. Via the interface, 3rd party tools can receive information about the generated code and the



Never stop thinking.



You never
buy
a part
Without doing
the **research.**

Which is why
we call it
info-commerce.

specs

| app notes

| pricing

| availability

Research, Compare, and Buy
electronic components and tools
from over 2600 suppliers.

The #1 way to reduce new product introduction cycles.



QUESTLINK

The #1 info-commerce site for the electronics industry.

www.questlink.com



Jack W. Crenshaw

Old Kids on the Block

Some of you readers have written to express support of my recent drive to identify companies whom you can trust to be pleasant to deal with, and whom you can't. It's not my intention to turn this column, which is supposed to be the column of the "resident math guru," into a review column, but I do have new and, in one case, exciting news.

Regular readers will recall that lately I've been looking at alternative programs for solving math problems. You'll recall my raving over Matlab, from The MathWorks Inc., and its companion program, Simulink. If you do any kind of work at all with dynamic simulations, you owe it to yourself and your company to get Simulink. 'Nuff said.

Not so appealing is the high cost of Matlab and its toolboxes. Taken all together, a corporate customer could easily spend \$5,000 for a single Matlab license. I've mentioned, in passing, that other alternatives are around such as Mideva and Scilab. Both come highly recommended, but I haven't personally looked at them yet.

Last month, however, I was thumbing through a magazine when I came across an ad for Maple 6, from Maplesoft. Now, I thought, more than a little belatedly and slow on the uptake, there's a nice idea. If Maple is the premiere symbolic engine of the world (and it is), and if all the other companies like Mathsoft and The MathWorks use it (and they do), and if a certain company seems to have forgotten how to make their program send the right data to it, or interpret what comes back, hey, why not go to

the horse's mouth, and get Maple itself?

So I did. I got a copy of Maple 6, with all the bells and whistles (they're all built in—no Barbie-doll tease as with some other companies). I'm happy to report that I'm in love all over again. Maple 6 is, in a word, *wonderful!*

Perhaps the reason I was so slow on the uptake is that I've always thought

WYSIWYG interface (which must make Mathcad incredibly difficult to program). Like Matlab, Maple 6 has a command-line interface. (To be fair, you can select a WYSIWYG mode, and build equations from their component pieces, much like the Mathtype utility built into Microsoft Word. However, it's tedious to use, and I gather the Maple gurus tend to stick with the command line.) On the other

This month, our fearless columnist revisits a couple of old friends, with mixed results. Then it's back to minima.

of Maple as a symbolic engine only, sort of like MacSyma. I don't know what I was thinking it had for a user interface. My own experience with Maple has been as it was embedded into Mathcad, so I didn't really think much about its stand-alone user interface at all. I probably never would have even thought of trying it, except that a few nice readers suggested I give it a try.

The thing that makes Maple 6 stand out is that Maplesoft has signed an agreement with NAG, the premiere supplier of high-performance, high-accuracy numerical solution modules. The NAG library is included in Maple 6, which means that not only can you get symbolic solutions galore, you can get fast and accurate numeric solutions, as well.

I won't kid you: Maple 6 is not Mathcad. It doesn't give you that nice,

hand, the generated equations *are* formed just as you'd expect to see them printed in a book, and they look great. What's more, you can insert text into the worksheets, using a wide variety of fonts, to build a nice, editable Rich Text Format (RTF) file.

If you've ever used Mathematica, you'll be comfortable with Maple 6. The user interface is much the same, although Maple has far fewer strangenesses like square instead of round parentheses. Maple 6 also has loads of graphics capabilities including the easiest way to plot yet invented by man: right-click on an equation, select "plot," and watch the picture pop up. What's that you say? You want to plot more than one function on the same graph? No problem, just drag the second equation onto the graph. Bingo!

To put icing on the cake, Maple 6 can call Matlab functions, not to men-

Break through:
the first true
x86 System-on-a-Chip

Stretch Your Imagination

Put the "FailSafe MachZ™" in your next generation embedded System.

You can design the MachZ system into virtually anywhere your imagination can take you. It's very small and 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** $\leq 500\text{mW}@100\text{MHZ}$

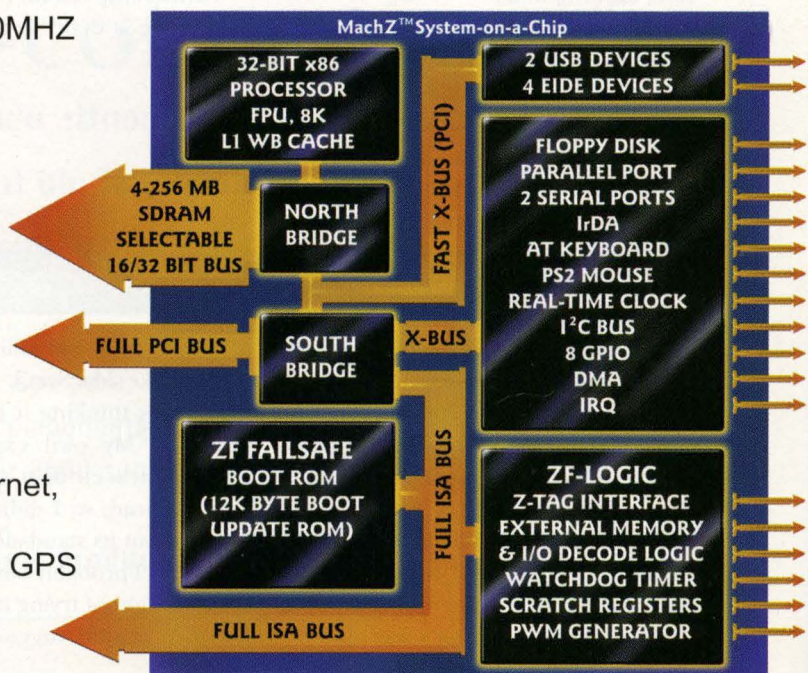
- **Auto-Boot FailSafe System**

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



Let your mind begin building the future.

www.zfmicro.com

ZF Embedded A division of ZF Linux Devices

Palo Alto, CA 94303 USA

Tel: 800.683.5943 Info: ZFmicro.com



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

ZF EMBEDDED
 FailSafe™



I was so excited when I read those words that I called up and ordered a new Dell the very same day. There's just one small problem: whatever deal Dell and Red Hat set up, Dell forgot to tell their personnel.

tion user-created C or Fortran functions. If you have Matlab on your computer, just tell Maple 6:

```
with(Matlab)
```

and you're in business. Matlab can also access Maple functions, so it's one big happy family. Wahoo! My cup runneth over. I'm beginning to think my getting upset with Mathcad was the best thing that's happened to me lately.

DELL fumbles...but recovers for loss

This next story starts out with more than a little associated sadness. Over the years, I tend to get attached to the companies I think of as "good guys." I guess Heathkit was the first such company I got attached to, and it was a loss to the nation when they went belly up. Another such company was definitely Borland, and discovering that Borland C++ v. 5.0 had bugs was like discovering that there was no Santa Claus. It felt almost like losing a loved one.

For years, I've had that special feeling about Dell Computers. My second PC clone, after a homebrew job, was a Dell (33MHz 486), which was nicely made, dead reliable, and ran like a train. I was so impressed that I bought three or four more; one for my wife, two for my son, and a couple more for me. I never had a moment's problem with any of them, the price was right, and, most important to me, any questions I might have were quickly and effectively answered by the Dell tech support people. To me, Dell was the Borland of hardware.

Okay, so I strayed from the fold for a time. I'm currently using another no-name clone that I bought five years

ago, and have been periodically updating. The company I worked for had a deal with a vendor that made the price very attractive. After the last upgrade, however, I realized that the price wasn't all that attractive after all. As in the ad for V-8 juice, I realized, "Wow, I could've had a Dell."

You regular readers know that lately I've been trying to wean myself from the Evil Empire and get into Linux. You can imagine my excitement, then, when I read in Robert Young and Wendy Rohm's *Under the Radar* (Scottsdale, AZ: The Coriolis Group, 1999) that Dell was one of the many large computer corporations who backed the IPO of Red Hat, Inc. More importantly, Dell and Red Hat worked up a partnership agreement. Young and Rohm write, "We decided Red Hat Linux would be a participant in the Dell Plus program, where a customer could call and order a configuration with Red Hat installed." And further, "... Red Hat participates in the Dell Plus program. This program allows customers of Dell Computers to call and order a new computer preinstalled and configured with Red Hat Linux. Dell has also devoted a special Web page to Linux machines and Linux software, with a link to Dell's Gigabuy, an on-line store for software and computer peripherals."

Such a deal—my favorite computer company, hooked up with my hope-to-be-favorite operating system vendors. Who could ask for more? I was so excited when I read those words that I called up and ordered a new Dell the very same day.

There's just one small problem: whatever deal Dell and Red Hat set up, Dell forgot to tell their personnel.

Though I am very much hopeful to get to the place where I am using

Linux exclusively, in today's reality I am irrevocably hooked into Microsoft Windows—Word is the editor I use to write this column, and it's what my editors expect to get. What I really needed was a dual-boot system, with both Linux and Windows installed. Microway, the vendors of my DEC Alpha system, had no problem with that, but Dell did. When I ordered the computer, the salesman was somewhat taken aback when I told him I wanted Linux preinstalled. He was taken even more aback when I told him I wanted a dual-boot system.

After going off to check with his boss, the salesman came back on line to tell me that yes, I could indeed have Linux, but only Linux. No dual boot, no Windows. Worse yet, I had to give up significant performance and peripherals. At the time, Dell's top-of-the-line systems were 750MHz systems, but I could only get Linux with 600MHz or less (Why? What does Linux care how fast the clock is running? Something about the ATA66 local bus driver). I also couldn't get certain other features usually available, like a built-in Zip drive or a USB port.

However, the salesman assured me, all was not lost. He said that once I got the computer, I could call Dell's tech support and they would be happy to walk me through the process of installing Windows.

This, I had to think about. Should I order the system with only Linux, and install Windows myself? Or should I get the plain vanilla (but 150MHz faster) system with Windows and the other hardware, and install Linux myself? I could certainly do either; I have Partition Magic and System Commander and have installed dual-boot systems before. I reasoned, however, that Linux is the more difficult system to install (true). Also, there seemed to be little point in insisting on the other hardware if Linux couldn't support it. I had visions of spending the next six months seeking out Linux device drivers. So I opted to go

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®

Next day, I called Dell's customer service department. Compared to this call, the conversation with the tech support guy was the balcony scene from "Romeo and Juliet."

with Linux preinstalled instead of Windows (good thing, too, or else I'd probably be deep in the study of bus drivers, and I don't mean Greyhound). I also reasoned, again rightly, that if I let Dell install Linux, I'd get certain Dell extras that I otherwise wouldn't get.

Well, I got the computer just before Christmas. What with the holidays and the cares of this world, like travelling on business and getting this column out, I only recently got around to placing that call. Imagine my surprise when Dell's tech support guy told me curtly, **"Dell does not support Linux."**

"Huh?" I asked. "But Robert Young said...." The techie only repeated the dread phrase, "Dell does not support Linux."

"But your salesman said...."

"Dell does not support Linux."

At this point, I was beginning to get miffed. I had bought a computer under the express condition that I'd get help installing Windows, and I wasn't getting it. I had taken a 150MHz performance hit, and I was getting nothing in exchange. The salesman had flat-out lied to me. I decided to return the computer, as a matter of principle.

Next day, I called Dell's customer service department. Compared to this call, the conversation with the tech support guy was the balcony scene from "Romeo and Juliet." The customer service rep told me, rather nastily, "This computer is non-returnable." She explained that I had kept it too long before trying to return it. I shouldn't have celebrated Christmas so long.

"But," again I whined, "The salesman told me...."

"Policy is policy," she said.

"Policy is what you make it. Mail

fraud is forever," I threatened.

"My decision is final," she countered.

"Let me speak to your superior," I growled menacingly.

"I have no superior; I'm it," she parried.

"Let me speak to Michael Dell."

"This conversation is over. [plonk]." Check and mate.

Grrrr.

After that, I sat down and wrote what must surely qualify as Dell's irate customer letter of the millennium. I sent it to Dell's customer service department via e-mail.

Two days later, I got a nice call from a manager at Dell. He apologized for the way my case had been handled. He assured me that Dell was still the nice, helpful company it had always been. He said that yes, indeed, Dell and Red Hat *do* have a partnership, but Dell has been a little slow in relaying that message to their personnel and in training them to deal with Linux. He assured me that the tech support and customer service people had been "talked to."

If there's a moral to this story, it's "Don't believe everything you read." Management agreements are one thing; reality is quite another.

So, do I get to return the computer? Nope. Do I get the missing 150MHz back? Nope. Is Dell going to send me another system with both Linux and Windows preinstalled? Nope. But at least I got someone at Dell to apologize. And, the manager assured me, if I needed help installing Windows, I could call Microsoft, and he'd pay for the service call. Such a deal.

Meanwhile, I called Red Hat's support line, which was as helpful as Dell's was useless. Though I really didn't get

the right department, the nice person there was extremely knowledgeable and walked me through the entire process of setting the computer up for dual boot. The first step? Wipe the hard drive.

Where we're going

If you're coming in new to this column, you'd be justified in thinking that it's a review column for computer hardware and software. It's not, really. Currently, we're in the midst of a multi-issue study of methods for finding the minimum of a function $f(x)$. We're nearing the end of that portion related to finding the minimum for the case where x is a single scalar variable, and we have no available derivative data. The main lesson we've learned is that we must, at all costs, never allow the minimum to slip through our fingers. In other words, we want to be sure the data points we retain always bracket the minimum.

The term "bracket" is easy enough to define when you're looking for the root of a function as opposed to a minimum. If the function is well-behaved (meaning, no fair using a square wave), and we have one point where $f(x)$ is positive and another where it's negative, you can be content that, as sure as God made little green apples, the root:

$$f(x) = 0,$$

is somewhere between them. The bracketing concept is a little more difficult to control when we're talking about minimizing a function, but it's still easy enough. We require three points, such that:

$$f(x_0) > f(x_1) < f(x_2) \quad (1)$$

In simple terms, the middle point, P_1 , must be strictly lower than the two end points. As long as this condition is maintained, we (theoretically) cannot fail to find the minimum. In practice,

We Put Our Heads Together



And Came Up With This Great ARM Solution.

ThreadX

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

MULTI 2000

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

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

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

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

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

www.ghs.com/armsolutions

Best support for the ARM Processor

ARM6
ARM7
ARM7M
ARM7TDMI
ARM7500fe
ARM8
ARM9
ARM9E
StrongARM

Other Processors Supported:

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

From the brains of

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

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

The solution, I'm convinced, is to intermix steps of Golden Section search and parabolic fit, according to some criterion.

however, things can get a lot trickier. The culprit is computer error, caused by the approximate nature of floating-point arithmetic. In mathematics, we can imagine that every continuous function can be computed to as many digits as we could possibly need. In a real computer using floating-point arithmetic this isn't true. Even with double or long double precision, the data word has a finite number of bits. If we get too greedy and try to narrow the range down too far, we can end up either with all three points having equal y -values or, worse yet, the last few bits changing randomly as we move slightly in x . Either way, the minimum gets away just as we were approaching (or, more accurately, already past) success.

Here's the situation in a nutshell: imagine that you have a magic microscope that can allow you to plot the function to any desired degree of magnification. (Most of you, in fact, do have such a magic microscope, in the form of a curve-plotting package of some sort.) If we zoom in the magnification enough, every function with a *smooth* minimum (as opposed to some kind of V-shaped trough)—no matter how asymmetric in the large view—begins to look more and more like a parabola. That's because, as we demonstrated last month, the Taylor series expansion reduces to only the term in the second derivative.

Zoom in the magnification some more and the parabola gets ever flatter. As long we don't get too greedy so that the parabola remains well-defined, we can clearly see where the minimum is. We can pick any three well-defined points on that parabola, fit a quadratic through them, and absolutely nail the minimum. We demonstrated that last month.

On the other hand, if we zoom fur-

ther yet, the parabola begins to look flatter and flatter, and the minimum becomes less and less identifiable. Eventually, one of two things will happen: either you'll get a curve so flat it looks like a straight, horizontal line, or you'll begin to see random noise caused by computer roundoff error. Either situation is fatal to our quest.

Confession time

Two months ago, I presented a nice, robust "divide-and-conquer" method based on the Golden Section ("And Another Thing..." March 2000, p. 15). Last month, we began to study second-order methods based on fitting a parabola to the available data. We found that iteration using the second-order method converged much faster than methods based on bisection of intervals. However, some things about the parabolic fit method still bothered me. In particular, I noticed that the method tended to hang onto points that were well away from the true minimum, leaving us with an interval (defined by the three points retained from step to step) that isn't shrinking much. That bothers me. In bisection methods, we use the size of the interval, not the estimated minimum, as the criterion to decide that iteration is complete. If the interval doesn't shrink, how do we know when to quit? It's easy to say that we simply compare successive estimates of the minimum and stop when the difference between them is small. But we also saw, only last month, that two successive trials can give exactly the same estimate, even though it's far away from the true minimum. So we have a problem.

The solution, I'm convinced, is to intermix steps of Golden Section search and parabolic fit, according to

some criterion. I fully intended to present that solution to you, in completed form and ready to use, this month.

Now I have a confession to make: the reason I was confident I could do this is because I had, or thought I had, a solution in my back pocket. It's Brent's method, as described in William H. Press's *Numerical Recipes in C* (New York: Cambridge Press, 1988). Though I haven't used it, at first glance it seemed to be exactly what I wanted: A combination of Golden Section and parabolic fit methods.

Well, it still is that, but the more I looked at Brent's method, the less I liked it. Or, more precisely, the less convinced I became that I *do* like it. At second glance, I can see that the method can and will use either the Golden Section or parabolic methods exclusively, depending on what it interprets to be the nature of the function. I am not satisfied that this is a Good Idea. As noted before, I think occasional section steps are necessary to be sure that the bounds of the region continue to shrink. Brent's method may in fact do this, but I can't be sure.

I was not encouraged by the fact that *Recipes* doesn't explain the method very well. Concerning a key criterion in deciding when the iteration is complete, they say only, "the reason for comparing [the result of the current parabolic step] to the step *before* last [emphasis in original] seems essentially heuristic."

My discouragement became complete when I sought further explanation for the method. When I first began this series, I must have bought fifteen books on the topics of either numerical methods, minimization, or optimization. (*Amazon.com* and I have a partnership agreement of our own: every month, they send me books and I send them money.) I searched through those books for other references to Brent's method. I didn't find one. Not one. Even *Recipes* gives only



A Golden Opportunity

Get The Winning Development Tools

Best Support for ST6 μ Cs

New golden tool chain is available from Ceibo and Raisonance for Gold ST62 μ Cs.

The development tools support most common STMicroelectronics ST62 μ Cs, all in a box.

The Ceibo EB-ST62 Emulator includes Raisonance Windows IDE and software, which can be easily upgraded to a highly optimizing C-Compiler, the first ever for ST6 family. This is the golden opportunity for design wins and quick time to market with a complete range of 8-bit devices.

For more information, or to place an order, call Ceibo at 1-800-833 4084 or contact us via the Internet at eb-st62@ceibo.com.

Online orders also accepted at www.ceibo.com.



Development Tools of Choice
www.ceibo.com
info@ceibo.com
 1-800-833 4084

RAISONANCE
www.raisonance.com

ST
mcu.st.com

Product and company names are trademarks of their respective organizations

It's a treasure trove of methods, loaded with literally hundreds of canned subroutines and often witty and certainly wise sayings. Yet, I find that I almost never use one of the canned subroutines verbatim.

Brent as a reference, and Brent's book is out of print.

Sometimes I get pretty frustrated by *Recipes*. A lot of people give it as a reference, and when I'm talking about a new subject, I often get e-mail recommending an algorithm from it. Clearly, the book is the most widely used numerical reference in computing. It's a treasure trove of methods, loaded with literally hundreds of canned subroutines and often witty and certainly wise sayings. Yet, I find that I almost never use one of the canned subroutines verbatim. Mainly because I don't like the coding style, which is clearly refried Fortran. Often, however, it's because the authors present complete routines with almost no explanation of why they're written as they are.

Face it: we have different approaches. *Numerical Recipes* covers a huge volume of techniques, for all phases of computer-based mathematics. I couldn't cover it all in this column if I lived to be as old as Methuselah. But, before I can present a method in this column, I have to do two things: I have to understand it myself, and I have to understand it well enough to explain it to you. I hate to use canned subroutines without knowing how they work, and I won't ask you to accept them from me, either.

The obvious problem with a book like *Recipes*, which presents solutions to so many problems, is that it can't possibly explain each solution to the same level of detail that I do here. Look at how much effort I've expended just on minimization alone. We're in our eighth month of discussion on the subject and still have a long way to go. *Recipes* devotes all of two pages to Brent's method.

So I find that, while I use *Recipes* a

lot, mining it for nuggets and ideas, I usually don't use its methods without tinkering around with them on my own. I also find that in the explanation of the methods there always seems to be that one tantalizing bit of information missing that would have helped me understand the method. It's frustrating.

Understand, I am not saying that Brent's method is *not* a good general-purpose solution. It may even be the very best of methods. I tend to trust the judgement of the *Recipes* authors on that score. I simply can't be sure. I really need to understand the subject more thoroughly before I can pass that understanding on to you, and it looks as though I'm going to be doing it on my own.

To paraphrase the bridge-builder in the movie *The Ghost and the Darkness*, "This may take longer than I thought."

What can we know?

Let's take this thing one step at a time (literally). Suppose we are just setting out to perform a step of Golden Section bisection. We begin with the starting situation we require, which is three points satisfying the condition of Equation 1, so we know we have the solution bracketed. We'll now compute one new value x and probe the function at that point. That new value x will be computed using the Golden Section proportion.

I'd like to be able to assure you that the three points we start with are already proportioned according to the Golden Section, but I can't. We may, for example, be just starting out, in which case the two intervals are almost certainly of equal size, that is, true bisections. Or, we may have just executed a step of parabolic curve fit, in

which case the relation between the two intervals may be anything at all. All I can promise is that the *next* point x will be chosen according to the Golden Section proportion. Let's assume that it goes to the right of x_1 , so that:

$$x = x_1 + (2 - \phi)(x_2 - x_1) \quad (2)$$

where:

$$\phi = \frac{\sqrt{5} + 1}{2} \quad (3)$$

is the golden ratio. If the original middle point was in fact proportioned according to the Golden Section, we'd now have four points with the relative values:

$$0, 2 - \phi, \phi - 1, 1$$

or, numerically,

$$0, 0.382, 0.618, 1$$

To avoid confusion in what follows, let's now relabel these points as P_0 , P_1 , P_2 , and P_3 , so the total interval spans $x_0 \dots x_3$, and the two interior points are at values x_1 and x_2 .

Given any three points, we can fit a parabola through them, as we did last month. But we now have four points. What do we do with this embarrassment of riches? Answer: use two sets of three. Let's first use P_0 , P_1 , and P_3 . The derivation of the curve fit is simple enough so that I can repeat it here, with slight variations from last month.

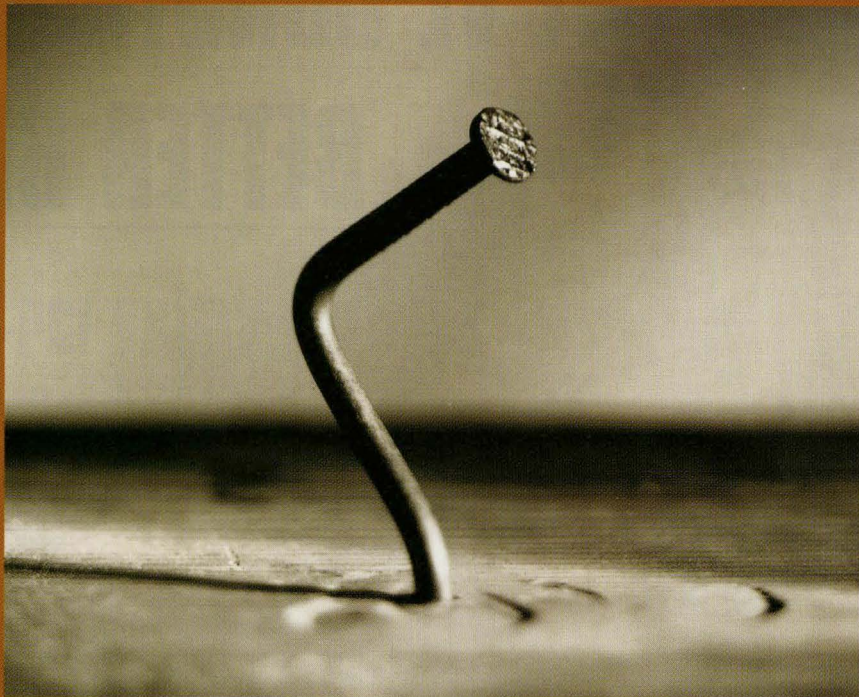
We assume a functional form:

$$f(x) = y_0 + (x - x_0)[a + b(x - x_3)] \quad (4)$$

Fitting this function at the three points gives the conditions:

$$y_0 = y_0 \quad (\text{no help})$$

$$y_1 = y_0 + (x_1 - x_0)[a + b(x_1 - x_3)] \quad (5)$$



To make no mistake.

That is what you seek.

To be always right. Ever accurate. To not look like a fool.

But that's not always easy. Not when you're exploring new ground.

When you will only find out what you don't know the hard way.

By trial and error.

We can help.

With test equipment that is accurate, easy and intuitive to use.

That can not only speed through a test, but through those wonderful folks in management.

And with the help and advice of our engineers who, like you, want to do what's right.

Make no mistake.

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



Agilent Technologies

Innovating the HP Way

A creative visual metaphor for a bridge advertisement. A suspension bridge is shown against a purple gradient background. Instead of traditional stone or steel pylons, the bridge is supported by four black rectangular bridge deck components. These components are labeled from left to right: "Bridge", "PCI-to-PCI Bridge" (with the Texas Instruments 'ti' logo), "PCI-to-PCI Bridge" (with the Texas Instruments 'ti' logo), and "PCI-to-PCI Bridge". The bridge cables are thin gold lines that connect the top of these components to a central point above the bridge deck. The bridge deck itself is a long, thin black strip that runs across the bottom of the image.

WE'VE CONSTRUCTED A BETTER BRIDGE.

- Pin-compatible* with Intel® PCI-to-PCI bridges, providing a simple drop-in replacement
- CompactPCI Hot-Swap functionality allows PCI bridge to be used in a variety of embedded systems
- MicroStar BGA™ package option allows designers to reduce board size without increasing the number of layers
- The PCI2250 and the PCI2050 provide two-tier internal arbitration for up to four and nine secondary masters, respectively

T H E W O R L D L E A D E R I N



TI'S PCI-TO-PCI BRIDGE DELIVERS PERFORMANCE AND VALUE.

A new line of PCI-to-PCI bridge products is being introduced by Texas Instruments, offering the best new price/performance value mark in the industry. With over 30 years of data transmission experience, TI is the industry leader in PCI technology. TI provides world-class manufacturing capability, leadership, compatibility testing, and packaging technology to deliver high-quality, yet affordable PCI-to-PCI bridges.

For more information about TI's PCI-to-PCI bridges, visit us at www.ti.com/sc/pci. If you'd like to take the best route, TI's PCI-to-PCI bridge is the way to go.

www.ti.com/sc/pci

* CompactPCI Hot-Swap functionality has been added.
MicroStar BGA is a trademark of Texas Instruments. Intel is a registered trademark of Intel Corporation. 3531-05 © 2000 TI

D S P A N D A N A L O G

 **TEXAS
INSTRUMENTS**

$$y_3 = y_0 + (x_3 - x_0)a \quad (6)$$

Equation 6 gives us a value for a :

$$a = \frac{y_3 - y_0}{x_3 - x_0} \quad (7)$$

Note that a has the form of a linear slope, between the points P_0 and P_3 . Accordingly, let's relabel this as m_3 .

Equation 5 now becomes:

$$y_1 = y_0 + (x_1 - x_0)[m_3 + b(x_1 - x_3)]$$

Solving for b gives:

$$y_1 - y_0 = (x_1 - x_0)[m_3 + b(x_1 - x_3)]$$

$$\frac{y_1 - y_0}{x_1 - x_0} = m_3 + b(x_1 - x_3)$$

Again, the left-hand ratio has the form of a linear slope, this time between P_0 and P_1 . We'll call this slope m_1 . We now have:

$$m_1 = m_3 + b(x_1 - x_3)$$

$$b = \frac{m_1 - m_3}{x_1 - x_3}$$

or

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

Yet again, we have a form similar to a slope, only this time it's a "slope of slopes," which is, of course, a measure of the acceleration, or curvature, between points P_1 and P_3 . Let's call this c_1 , for curvature.

We now have both parameters solved, and we have a new form for the fitted parabola:

$$f(x) = y_0 + (x - x_0)[m_3 + c_1(x - x_3)] \quad (9)$$

We next seek the value of x for which this function is minimized. To get that, we differentiate Equation 9 to get:

$$f'(x) = [m_3 + c_1(x - x_3)] + c_1(x - x_0) \\ = m_3 - c_1(x_0 - 2x + x_3)$$

Setting this to zero gives:

$$m_3 = c_1(x_0 - 2x + x_3)$$

$$\frac{m_3}{c_1} = x_0 - 2x + x_3$$

$$2x = x_0 + x_3 - \frac{m_3}{c_1}$$

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

When we cast the equation in this form, we can see immediately that the value of the minimum consists of two components. The first is simply the average value, which is the same as the bisection, of the two end points x_0 and x_3 . The second is a term dependent on the linear slope m_3 , and the curvature c_1 . Note in passing that we have no worries about a divide-by-zero error related to c_1 , since it cannot be zero if the condition of Equation 1 (adjusted for the relabelling) is met.

Return with me now to the yesterday of last month, where I tried a couple of middle values over the interval on our test function of 0..1. That function, you'll recall, is

$$f(x) = \cos(2\pi x^3) \quad (11)$$

For $x_0 = 0$ and $x_3 = 1$, the value of the function itself is one. This means that the slope m_3 is zero. Now we can see that the algorithm has no choice but to "recommend" an expected minimum of 0.5 for *any* choice of x_1 , even though that value is very far indeed from the true minimum.

If there's any lesson to be learned from this result, it's that comparing successive values of x_{\min} is *not* a good indicator of success.

But, you say, let's not forget that we now have a fourth point, x_2 . This time, we'll fit a parabola through P_0 , P_2 , and P_3 . Following exactly the same derivation, we end up with a fitted function:

$$g(x) = y_0 + (x - x_0)[m_3 + c_2(x - x_3)] \quad (12)$$

$$\text{where } c_2 = \frac{m_3 - m_2}{x_3 - x_2} \quad (13)$$

The estimated minimum is, for this function, at:

$$x_{\min 2} = \frac{x_0 + x_3}{2} - \frac{m_3}{2c_2} \quad (14)$$

We see that the forms for the minimum are identical, except for the use of c_2 instead of c_1 . Again, if $y_0 = y_3$, then $m_3 = 0$, and the differing values of the c 's won't alter the estimated location of the minimum.

So we've now learned yet another important result: comparing the x -values for the minima using x_1 and x_2 isn't helpful, either. In this case, both sets of points yield exactly the same, wildly incorrect estimate.

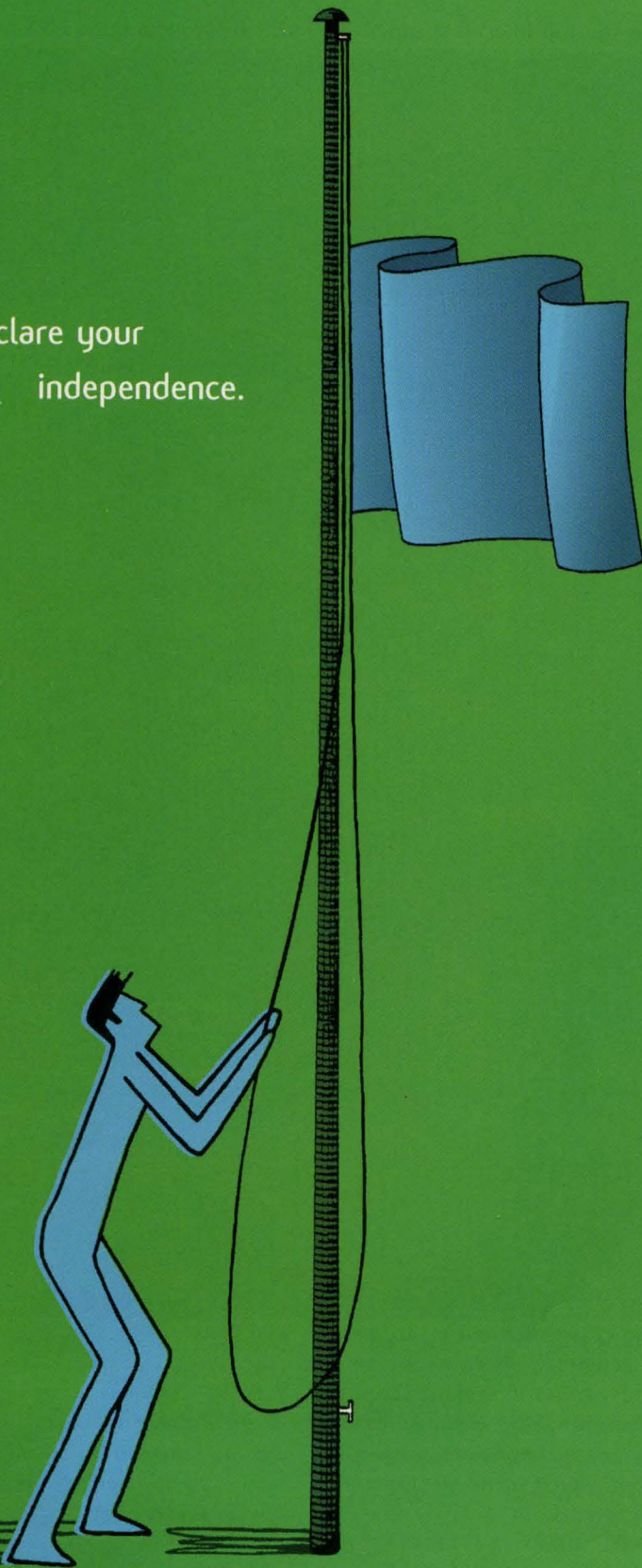
Bending the bow

Last month, I used the mental image of a flexible rod or bow, balanced on the points P_0 and P_3 . If we pull down on this bow at any point, it always flexes in the form of a parabola. Altering the choice of the middle of the three fitted points changes the *curvature* of the parabola, but it won't and can't change the parabolic shape. If the two values y_0 and y_3 happen to be equal, as was the case for our test function, the lowest point of the bow will always be halfway between them. Therefore, the value of x_{\min} will always be exactly the same, 0.5, regardless of our choice of x_1 .

Clearly the values of x_{\min} do not give us enough information to tell if we've got a good fit. The question is, are there other values that do? Is there something about the parabolic "bows" that are different as we change x_1 ? The answer is obvious when you think of things in terms of the bent bow: the difference is the curvature itself, which is reflected, not in the location of x_{\min} , but in the value of $f(x)$ it predicts.

Let's again write the functions for the two cases. This time, I'll use a subscript on the $f(x)$ to discriminate the central point used to determine the fit:

Declare your
independence.



Who controls your business? Or to put it another way: who controls your ASIC cores?

Processor cores are the very heart of your product. The vital component without which you don't even have a product or a business. Yet who designs them? Nobody in your company.

Now think again: who controls your business?

If you don't like the answer, here's a different question. Can you imagine your engineers being able to design their own processors? On their desk? In half a day? Imagine the control that would give you. The differentiation from your competitors. Instead of being handcuffed to your ASIC vendor, you have your own chip and you're free to find a manufacturer.

All bidders please form an orderly line,
lowest at the front.

Don't pinch yourself. This isn't a dream. With ARC Cores' processor creator (the ARChitect), you can take complete control. You can make substantial savings in silicon costs (some of our customers are already doing so). And that's before you even take into account the advantages of creating the processor in hours, days or weeks instead of months.

You can change the way your engineers work,
and change the way you do business.

Download a free demo of our ARChitect processor creator at www.arccores.com to find out how.
Then declare your independence.

microprocessors by

ARC
Start thinking.

$$\begin{aligned} f_1(x) &= y_0 + (x - x_0)[m_3 + c_1(x - x_3)] \\ f_2(x) &= y_0 + (x - x_0)[m_3 + c_2(x - x_3)] \end{aligned} \quad (15)$$

For the predicted minima:

$$\begin{aligned} y_{\min 1} &= y_0 + (x_{\min 1} - x_0)[m_3 + c_1(x_{\min 1} - x_3)] \\ y_{\min 2} &= y_0 + (x_{\min 2} - x_0)[m_3 + c_2(x_{\min 2} - x_3)] \end{aligned} \quad (16)$$

the difference between the two values is:

$$y_{\min 2} - y_{\min 1} = (x_{\min 2} - x_0)[m_3 + c_2(x_{\min 2} - x_3)] - (x_{\min 1} - x_0)[m_3 + c_1(x_{\min 1} - x_3)]$$

Expanding and "simplifying" yields:

$$\Delta y_{\min} = (x_{\min 2} - x_{\min 1})m_3 + c_2(x_{\min 2} - x_3)(x_{\min 2} - x_0) - c_1(x_{\min 1} - x_0)(x_{\min 1} - x_3) \quad (17)$$

To further simplify, we must consider the values of c_1 , c_2 , $x_{\min 1}$, and $x_{\min 2}$. From Equations 10 and 11, we have:

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

and

$$x_{\min 2} = \frac{x_0 + x_3}{2} - \frac{m_3}{2c_2}$$

Thus:

$$\begin{aligned} x_{\min 2} - x_{\min 1} &= \frac{m_3}{2c_1} - \frac{m_3}{2c_2} \\ &= \frac{m_3}{2} \left(\frac{1}{c_1} - \frac{1}{c_2} \right) \end{aligned} \quad (18)$$

Likewise:

$$\begin{aligned} x_{\min 1} - x_3 &= \frac{x_0 - x_3}{2} - \frac{m_3}{2c_1} \\ x_{\min 1} - x_0 &= -\frac{x_0 - x_3}{2} - \frac{m_3}{2c_1} \\ x_{\min 2} - x_3 &= \frac{x_0 - x_3}{2} - \frac{m_3}{2c_2} \\ x_{\min 2} - x_0 &= -\frac{x_0 - x_3}{2} - \frac{m_3}{2c_2} \end{aligned} \quad (19)$$

Define:

$$h = x_3 - x_0 \quad (20)$$

which is the full interval spanned by the points. Then we can write:

$$\begin{aligned} x_{\min 1} - x_3 &= \frac{1}{2} \left(h - \frac{m_3}{c_1} \right) \\ x_{\min 1} - x_0 &= \frac{1}{2} \left(-h - \frac{m_3}{c_1} \right) \\ x_{\min 2} - x_3 &= \frac{1}{2} \left(h - \frac{m_3}{c_2} \right) \\ x_{\min 2} - x_0 &= \frac{1}{2} \left(-h - \frac{m_3}{c_2} \right) \end{aligned} \quad (21)$$

Then, finally:

$$\begin{aligned} (x_{\min 2} - x_3)(x_{\min 2} - x_0) &= -\frac{1}{4} \left(h - \frac{m_3}{c_2} \right) \left(h + \frac{m_3}{c_2} \right) \\ &= -\frac{1}{4} \left[h^2 - \left(\frac{m_3}{c_2} \right)^2 \right] \end{aligned} \quad (22)$$

Likewise:

$$(x_{\min 1} - x_3)(x_{\min 1} - x_0) = -\frac{1}{4} \left[h^2 - \left(\frac{m_3}{c_1} \right)^2 \right] \quad (23)$$

Equation 17 now becomes:

$$\begin{aligned} \Delta y_{\min} &= \frac{m_3^2}{2} \left(\frac{1}{c_2} - \frac{1}{c_1} \right) - \frac{c_2}{4} \left[h^2 - \left(\frac{m_3}{c_2} \right)^2 \right] + \frac{c_1}{4} \left[h^2 - \left(\frac{m_3}{c_1} \right)^2 \right] \\ \Delta y_{\min} &= \frac{m_3^2}{2} \left(\frac{1}{c_2} - \frac{1}{c_1} \right) - \frac{h^2}{4} (c_2 - c_1) + \frac{m_3^2}{4} \left(\frac{1}{c_2} - \frac{1}{c_1} \right) \\ \Delta y_{\min} &= \frac{3m_3^2}{4} \left(\frac{1}{c_2} - \frac{1}{c_1} \right) - \frac{h^2}{4} (c_2 - c_1) \end{aligned} \quad (24)$$

Now we have a relationship that depends only upon the values of the c 's. The values of h and m_3 are fixed by the geometry. There is little point in going further in seeking simplicity; substituting the expressions for the c 's just makes things worse. We also don't need to do so. Equation 24 represents, in a fairly concise form, the difference between the values of $f(x)$ estimated for the two parabolas. Now our criterion for success can be made very simple:

this difference must be small, compared to some range yet to be defined. If it isn't small, we have to conclude that one of our parabolas, and probably both, are not yet close enough fits to the actual function to be useful.

Obviously, we will need to perform some arithmetic to calculate the values of the m 's and c 's, and, finally, Δy_{\min} . Such calculations take time, but they are not very complex, after all. The m 's and c 's are simple divided differences, and Equation 24 is not exactly backbreaking work for a modern CPU. Nevertheless, we must acknowledge that extra CPU time is required, so we'll want to minimize it.

At this point, I'm thinking that we can be fairly sure that, when our total interval $x_0 \dots x_3$ is large, there isn't much point in trying to fit parabolas. We're better off just doing a few simple steps of Golden Section estimation. Then we perform the double parabolic fit, using the two inner points from the last section step. The value of Δy_{\min} , as given by Equation 24, then tells us if we're getting near the solution.

If we are, then we might take several parabolic steps before we do anything else. From our experience last month, these steps may well take us sufficiently close to a solution, and we're done. If not, then we'll take more Golden Section steps to get closer and try again.

I realize that all of this is not an algorithm yet. There are many more criteria to consider, which is where we'll take things up next month. See you then.

esp

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

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®



Playback: Reality-Based System Testing

A record-and-playback test methodology lets you exercise your system with realistic test data. But making a system playback-ready offers some challenges of its own.

A critical challenge in the development of any system that processes data digitized directly from that unpredictable analog space we call the physical world is presenting realistic test data to the system. Environmental simulators are a good starting point, but they are not the final answer.

Not too long ago, I was a member of a team tasked with creating an airborne system that could operate in mountainous terrain and detect the presence and compute the location of active radar emitters. These emitters would be operating out of our line-of-sight on the opposite side of a mountain ridge from a C-130 flying below the ridgeline. To succeed, our system

would have to operate in a signal environment with a characteristic known as "multi-path." In a multi-path environment, the pulses from a single radar emitter have countless signal propagation paths to a receiver because of environmental reflections. Unfortunately, we could not accurately model the effects of multi-path and so could not thoroughly test our system prior to flight test. But we flew and succeeded anyway—here's how.

We have taken a step beyond using simulated environmental data and routinely test our electronic warfare (EW) systems with actual target environment data recorded during field trials. We refer to the technique as "playback." Playback allows us to dramatically improve product quality and significantly reduce the system verification

effort. Our group has built playback technology into nearly every major EW system created in the last eight years. Without it, system performance would be virtually impossible to completely verify and testing unbearably tedious.

This article will discuss playback technology in general, with references to EW systems as real-world examples. The following points will be covered:

- Definition of playback
- Benefits of playback technology
- Evaluation criteria for determining your playback needs
- Playback data requirements
- Playback design topics

Playback defined

Playback is the recording of both target environment input and system out-

We discovered that, for us, playback provided benefits useful at nearly every stage of system operation and had benefits to our company as a whole.

put data and the subsequent use of that data as input to embedded software tests and to various off-line analyses. Playback has two modes that can be used separately or in combination: *replay* and *record*. In record mode, the system outputs a playback file while obtaining data from either normal sensor inputs or a playback file. In replay mode, the system obtains its primary inputs from a playback file, created during record mode, instead of from sensor devices.

The purpose of playback is to permit repeated testing of the embedded system software, in a lab environment, against data of interest recorded in the potentially remote, hostile, or expensive-to-access target environment. Input data from the target environment is always a very valuable commodity because it represents the problem domain as it is, not as simulated or imagined. Target environment data has two other very important aspects: it can be very expensive to acquire and it has credibility with the customer.

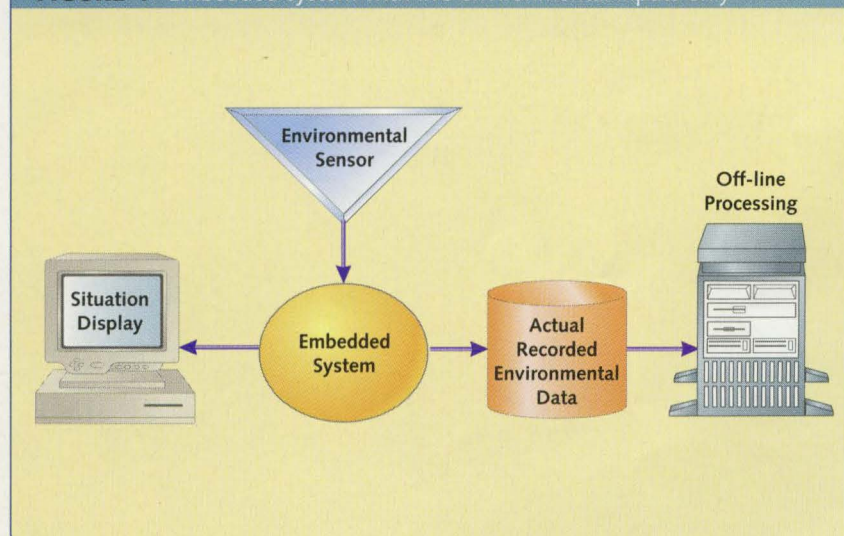
I've provided three figures that should make it clear just how a playback-equipped system differs from those without. Figure 1 depicts an embedded system with only environmental sensor inputs and a log file output. Obviously the embedded system software has no access to the log file. While it may be possible to evaluate the performance of the system off-line, only another field trial will show if the software changes were effective. A system like this can only be thoroughly tested in the presence of simulated analog data presented to the environmental sensor(s); usually a very expensive proposition for all but the simplest of target environments.

Figure 2 shows an embedded system with both sensor and simulated environmental inputs. This is a dra-

matic improvement because now the developers can repeatedly test the system, in the lab, against data that approximates live sensor inputs. This was the typical configuration of our EW systems prior to the introduction of playback technology. We could either run our system against simulat-

running against the recorded sensor data. The re-execution of a field trial via replay of the playback file is fundamental to playback's ability to dramatically reduce system verification effort. In the case of our flight test, we simply would not have succeeded without it.

FIGURE 1 Embedded system with live environmental inputs only



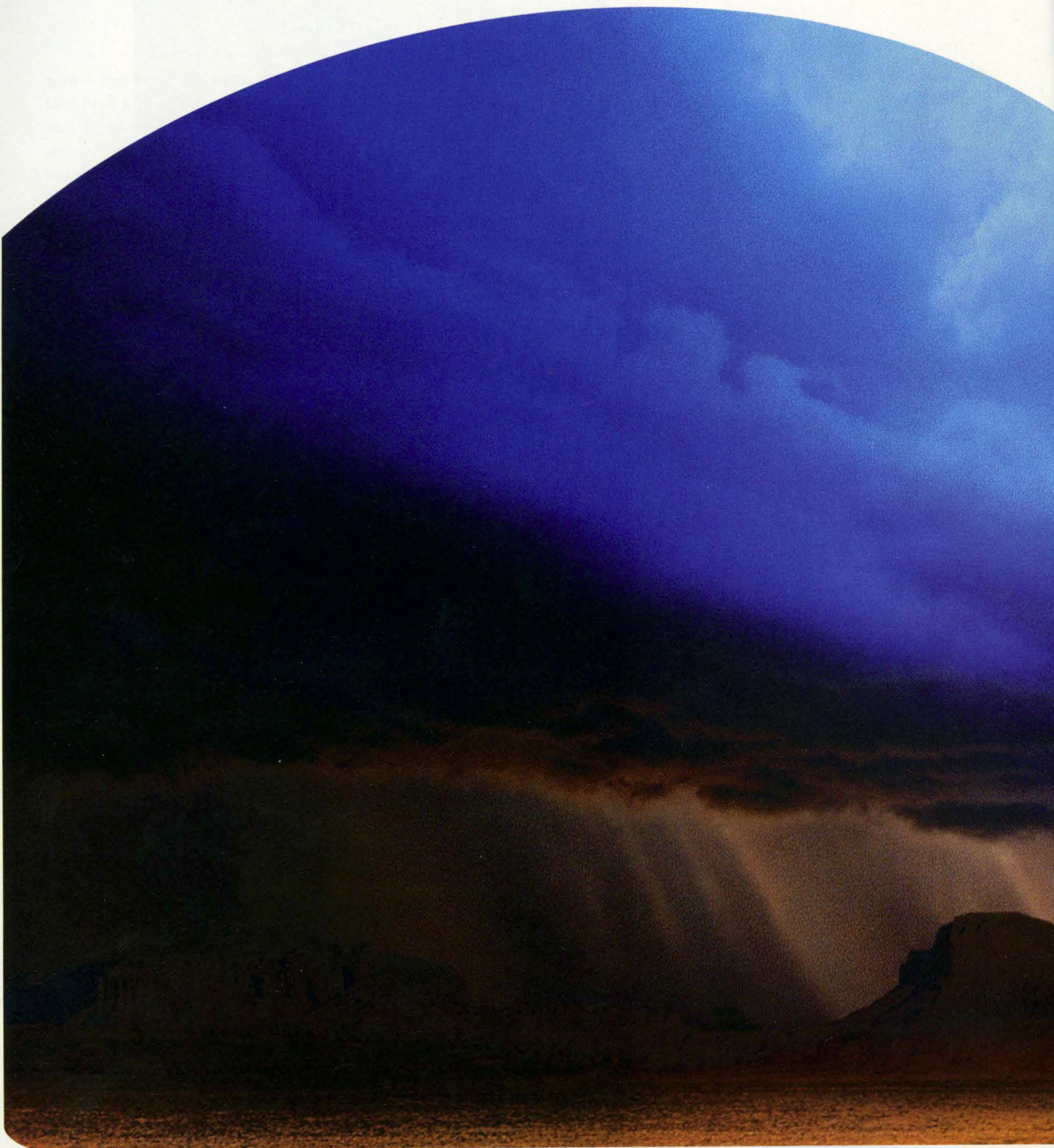
ed radar pulse and navigation data or fly against the test range. But once we were back on the ground, our embedded software had no access to the recorded data from the target environment. The log file could be exhaustively processed off-line and many errors were discovered in this manner, but the central algorithms of our system remained isolated from the data they needed to complete their testing.

Figure 3 shows an embedded system with all three possible inputs: live, simulated, and actual recorded environmental inputs. Such a system can perform field trials, simulations, and effectively re-execute a field trial by

Uses of playback during flight test

Once we implemented playback, we wondered how we ever got along without it. We discovered that, for us, playback provided benefits useful at nearly every stage of system operation and had benefits to our company as a whole. Let me illustrate the benefits that accrued to us during our field trial starting with the 7 a.m. pre-flight check and ending with post-flight analysis in the late p.m. Table 1 summarizes the benefits.

Stimulating other systems with a playback-equipped system. All on-board mission-critical subsystems had to pass a cursory pre-flight test to demon-



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.

The recording of the sensor data did not depend on our application software generating correct answers.

actual radar pulses diffracting over mountain ridges and reflecting from mountain slopes and spinning propeller blades.

The entire flight test was made up of one or more "runs," each of which consisted of a series of points, called "way-points." A test involved flying, at a specified speed and altitude, from one way-point to the next while our system looked for the test emitters that were being switched on and off, according to a test procedure, by an independent test team on the ground. Record mode was enabled only during a run since that was the only time we had a known standard (that is, the test procedure) against which to compare our performance and because conserving disk space was a high priority.

The recording of the sensor data did not depend on our application software generating correct answers. Even if we did less well than we hoped during the test, we would still bring back the recorded pulse and navigation data and that would allow us to succeed on the next flight.

Performance verification. Test range playback files served a dual purpose with respect to customer verification. First, we could literally re-fly the flight test in the lab and show, via a target display, that the system was performing correctly. Second, the customer would input our playback files to their own analysis software for further verification as to the accuracy of our results.

Replay in the lab. After the flight, the target system was physically removed from the aircraft and set up to run in a lab environment. At this point, we had access to the test procedure and so knew precisely when the emitters were turned on and off, their geographic location, and the frequency and modes at which they were operating. The target system would then be run in replay mode against the very same emitter data we had flown against hours before. This gave us the opportunity to debug our system until

FIGURE 2 Embedded system with live and simulated environmental inputs

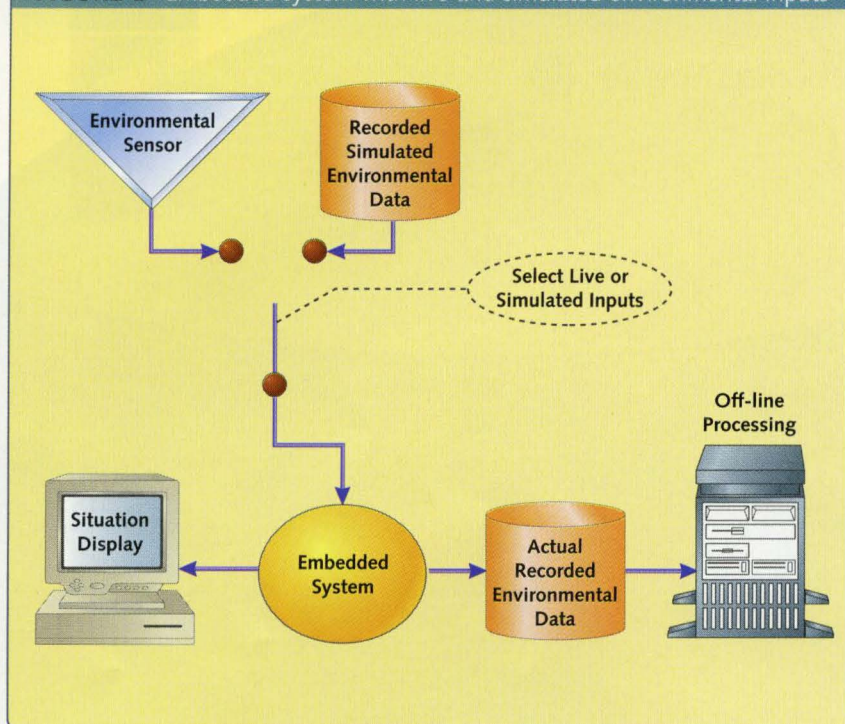


TABLE 1 Uses of playback during flight test

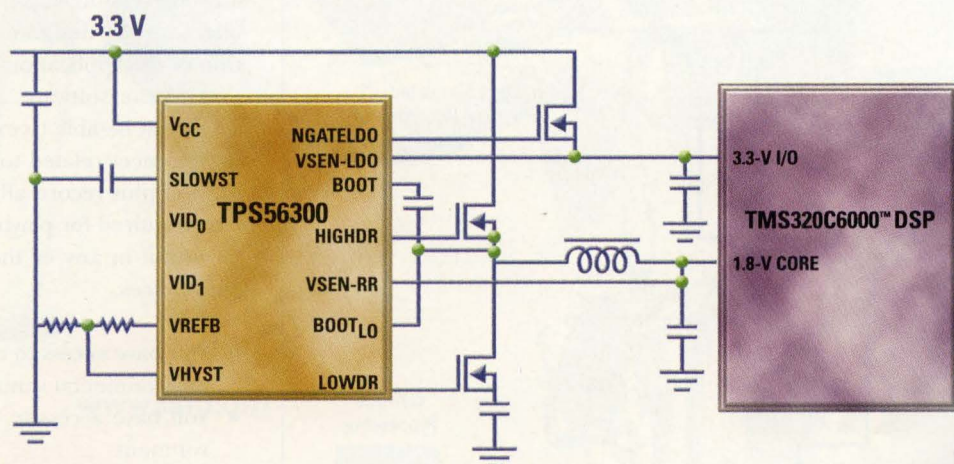
Time	Activity
7:00 – 7:30 a.m.	Pre-flight check. Used playback files to simulate flight test to determine if all systems are communicating
9:00 – 10:00 a.m.	En route to test range. Operated system to detect FAA radar activity and recorded data.
10:00 a.m. – noon	Record test range radar data while performing our test.
Noon – 1:00 p.m.	Record targets of opportunity while other on-board systems tested
3:00 p.m. – 5:00 p.m.	Mission re-fly in lab for customer verification of performance
7:00 p.m. – whenever	Replay range test files in lab to fix software bugs

strate flight-test readiness. Failure to do so could cause the flight to be scrubbed or delayed. Although scanning for nearby FAA radar emitters was an easy way to show that our system was working, use of playback files from previous flight tests provided more realistic outputs to other on-board systems. In this mode, we were able to see that other systems were processing our data correctly. The test officer knew that we were using playback files to do this and approved

because he knew that it was "real" data from a recent flight.

Range test. The range test was the moment of truth, not only for our application software but also for our playback technology. Obviously, it was a critical moment for our application software to show its stuff. But, less obvious, it was a key moment for our data recording software. On the range, we could collect the data that we could neither simulate nor succeed without:

DUAL CONTROLLER FOR LOW-VOLTAGE SYSTEMS MANAGES POWER-UP/POWER-DOWN SEQUENCING.



SPLIT-RAIL APPLICATION USING TI'S TPS56300

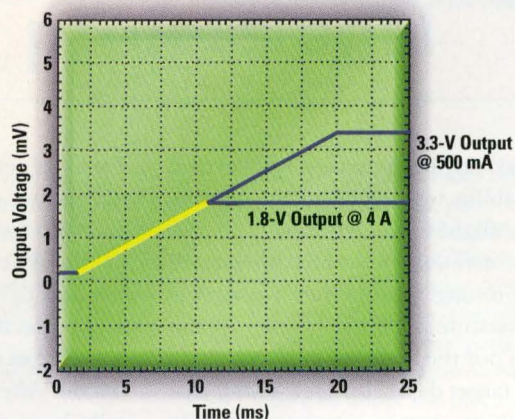


 28-pin TSSOP
PowerPAD™ package

Call now to purchase a **TPS56300EVM-139**
evaluation module at a special discount price of **\$30.00***

- ▶ Sequencing ensures simultaneous power-up of both outputs for better system reliability
- ▶ 2.8-V to 5.5-V input range
- ▶ VID pins offer nine preset output voltages to simplify interface with multiple core and I/O voltages
- ▶ TI's TPS56300 is ideal for split-rail DSPs, FPGAs and microprocessors
- ▶ System-protection features include overvoltage, undervoltage and adjustable overcurrent protection
- ▶ Switching regulator powers the core, LDO powers the I/O
- ▶ TPS56300 from \$4.06 per device in quantities of 1,000

SEQUENCING POWER-UP



To take advantage of our EVM discount or for TI's latest Power Supply Control Products catalog, call

1-800-477-8924, ext. aap3588u

For data sheets, application notes and samples, visit us at **www.ti.com/sc/tps56300**

* Offer good for a limited time only. Limit one discounted EVM per customer.
TMS320C6000 and PowerPAD are trademarks of Texas Instruments. 3524-35

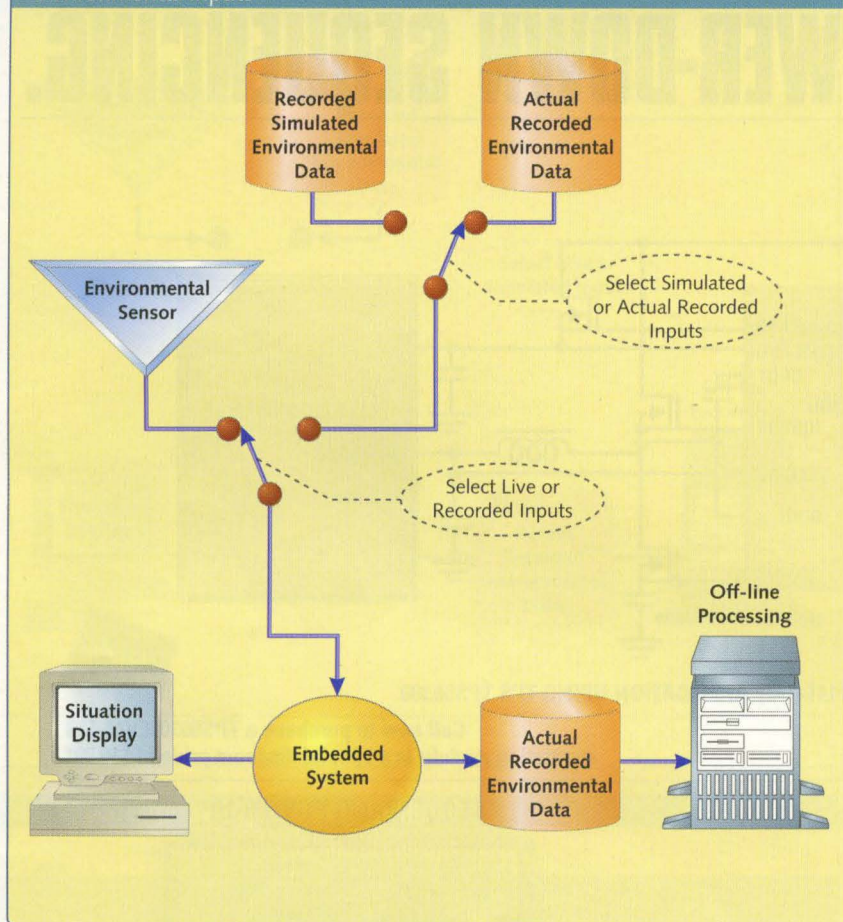
© 2000 TI

THE WORLD LEADER IN DSP AND ANALOG

 **TEXAS
INSTRUMENTS**

A probe version of an embedded system is simply a version of the system capable of recording the environmental data required for playback but with little or no application processing functionality.

FIGURE 3 Embedded system with live, simulated, and actual recorded environmental inputs



test schedule, the customer used our EW system as an environmental probe. This was done in order to improve our understanding of how the multi-path signal environment was perceived by our system.

A probe version of an embedded system is simply a version of the system capable of recording the environmental data required for playback but with little or no application processing functionality. Consider implementing a probe version of your system in parallel with the design and implementation of the application-processing portion of the software. The probe version must be able to exercise all hardware devices related to environmental sensing plus record all environmental data required for playback. This could be useful in any of the following circumstances:

- You have access to a realistic target environmental simulator
- You have access to the target environment
- Completion of application processing software is behind schedule

If your target environment is outer space then, for the purposes of playback, an environmental simulator is as close as you will ever get to the target environment. Access to such simulators can be very expensive, inconvenient, and heavily scheduled. Running a probe version of your system in a simulated environment would provide the processing software development team with valuable inputs without the trouble of actually being connected to the environmental simulator.

If you have access to the target environment then, as with the environmental simulator, running a probe version of your system in that environment will yield useful real-world data to your development team. If the target platform for your system is an aircraft or naval vessel or other expensive platform, you'll find that they are scheduled very tightly to maximize utilization. However, a little negotia-

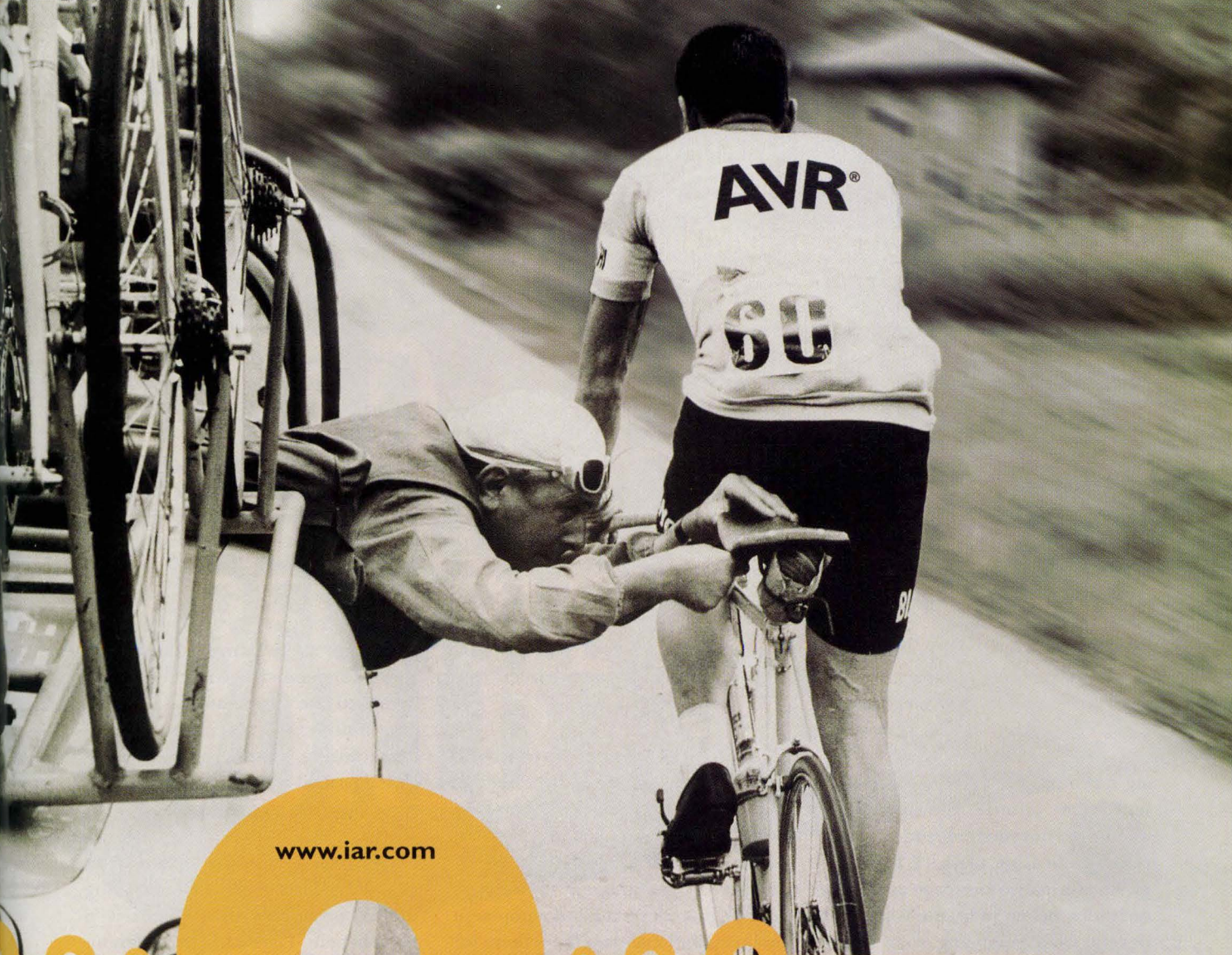
its playback performance was correct. This ability was key to our succeeding in our flight tests.

As mentioned earlier, we started flight testing with neither a complete understanding of the physics of the range nor the ability to generate simulated target data. However, saying that something can't be simulated is not to say that it can't be anticipated. Our analysts had designed algorithms to deal with multi-path as they understood it. The algorithms were heavily parameterized and these parameters were adjusted until the correct values were determined. Also, new algorithms were developed as playback ses-

sions deepened our understanding of the emitter environment.

Flight tests, like most field trials, are heavily scheduled and the culture is "fix and fly" until the money runs out or the aircraft or ship or other test platform leaves for its next scheduled duty station. After a field trial you will typically have only a few days to resolve problems and be expected to go again. It is imperative that you make the most of the time between trials. Playback technology greatly improves the efficiency with which you use this time.

Our system as an environmental probe. Toward the end of our flight-



www.iar.com

IAR and Atmel – fast moving microcontrollers with professional tools support

Atmel's AVR 8-bit RISC microcontrollers are on the move. Fast and competitively priced, they are rapidly winning ground with embedded developers for small applications as well as those demanding high performance tasks. IAR Systems played a key roll in this success. The AVR is well suited for high level language programming and the IAR Embedded Workbench now offers the choice of C/EC++ to most AVR applications.

Today, IAR continues to cooperate closely with Atmel, and provides a complete development solution:

- **IAR Embedded Workbench™ for AVR**, a state-of-the-art integrated development environment with a highly optimized C/EC++ compiler
- **IAR MakeApp™ for AVR**, a device driver wizard generating all initialization code
- **IAR visualSTATE® for AVR**, a graphical development tool with a patented technology to generate production ready code

Clearly, a unique solution that is hard to beat.

As IAR supports more than 30 different architectures, you are probably already familiar with IAR's development environment. This familiarity will help reduce start-up time for your next AVR project. Therefore if you want to reach your goals quickly and without costly interruptions, make sure to adopt the IAR solutions for your embedded systems development. When you're racing against the clock, there's no better partnership to have at your side.



IAR and Atmel – embedded development on the move

For more information, contact IAR at 1-800-427-8868

OR email: info@iar.com



DIFFERENT ARCHITECTURES.
ONE SOLUTION.

USA & Canada 415 765 5500, info@iar.com **Germany** +49 89 90 06 90 80, info@iar.de
UK +44 171 924 3334, info@iarsys.co.uk **Sweden** +46 18 16 78 00, info@iar.se **Denmark** +45 8625 1111, info@iar.dk
ATMEL AND AVR ARE REGISTERED TRADEMARKS OF ATMEL CORP. EMBEDDED WORKBENCH, MAKEAPP, VISUALSTATE ARE TRADEMARKS OF IAR SYSTEMS.

Having used playback on three separate and very different EW systems, I can't see why the builders of virtually any embedded system wouldn't want to use it.

tion with the test director may allow you to operate your probe system in a passive mode to collect valuable test range data. This data can be taken back to the lab and given to the developers to test the application processing software.

Evaluating your system's need for playback

Having used playback on three separate and very different EW systems, I can't see why the builders of virtually any embedded system wouldn't want to use it. However, making a system playback-ready does not come without some effort. You should perform a cost/benefit analysis before continuing down this path. This section presents the criteria for evaluating whether or not your embedded system could profit from playback technology. An affirmative answer to any of the questions below indicates you could benefit from equipping your system with playback.

Are you unable to produce or obtain simulated data that very accurately represents the entire scope of environmental characteristics of interest to your system? This is the fundamental question you need to answer. If the answer is "no" or "we're not sure" then you will benefit from using playback. A negative answer implies that you must submit your system to a field trial to completely verify its performance. Playback is an excellent way to verify system performance prior to going into an expensive field trial.

We knew that we could not model all the propagation paths between a radar emitter and a C-130 flying below the ridgeline on the opposite side of a mountain. It simply wasn't possible, given our schedule and budget. Our analysts said it wasn't possible at any cost because we simply didn't know enough about how our receiver truly

perceived the radar signals in that environment.

Is the target environment in a remote location? Perhaps your system is going to operate on the North Pole. How often do you want your system or yourself to go there? With playback, you go can go there once, record the necessary environmental data and return to the warm lab to test until things are working. Our flight test was over a military test range in California. The ground and air space there is controlled by the military and access to it is by invitation only.

Is the target environment expensive to access? A location needn't be remote to be expensive to access. A naval sea trial may take place right off the coast of your city, but it will still cost plenty and be very inconvenient to transport your engineering staff and equipment on-board and maintain them there for the duration. And if you forget something or something breaks after you put to sea the Navy will have to fly whatever or whomever to and from an aircraft carrier deck. For our flight test, we had to put our system and ourselves aboard the aircraft. An empty C-130 has an operating cost of about \$2,000 per hour. Add in the salaries of the engineers, and the sumptuous in-flight meal, and you're talking serious money. The test range itself costs about \$1,500 per hour and we usually spent from two to three hours on range. So an average flight costs about \$15,000. Playback can save you thousands of dollars in just this area alone.

Is your time in the target environment extremely limited? If the target environment of your system is outer space or an ocean trench, it's likely that acceptance testing on your system will be done in a completely simulated environment. The mechanism by

which this simulation will be accomplished will be fantastically expensive and heavily scheduled. The more expensive the test platform and the more systems that must be integrated together, the less time any one company will have on that platform. You must maximize every minute you have in the test environment; playback is the best way of doing that.

Is the target environment a hostile place for humans and/or equipment?

The interior of a C-130, flying low and slow over desert terrain, is no place for good engineering work to be done. The temperature of the cargo/engineering area was certainly in the 90s and the decibel level equaled or exceeded the temperature. The heat and noise, G-forces, and constant maneuvering, climbing, and diving gave nearly all but the flight crew moderate to severe motion sickness.

Playback data requirements

In any system design effort, knowing the requirements beforehand is a great benefit. Playback is no different. The following sections address the source of most playback requirements.

Identify what portion of the software you intend to test with playback. Most embedded systems with multiple sensor inputs have at least the following three processing layers or phases: acquisition of raw sensor data from hardware interfaces; pre-processing of sensor data; and core processing of pre-processed sensor data. (Figure 4 depicts these layers, using our EW system as an example.)

Phase one is usually comprised of software that interfaces directly to the hardware. This software configures the sensor devices to acquire data, allocates buffers for the received data, copies the data into the buffers, and queues the raw data buffers onto the pre-processing programs.

Pre-processing transforms raw data into a form where the core algorithms of the system can process it. Examples

Change the way you think.

Think what you could change.

BUILD A MORE RELIABLE WORLD™

START WITH A MORE RELIABLE OS



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

An OS with an architecture so advanced it lets you pinpoint memory faults in minutes.

Prototype without system rebuilds. Add features without retesting your entire software suite.

Upgrade systems without rebooting. And respond faster to what customers want.

Now imagine. What if this same OS was backed by two decades' experience in the embedded market. An outstanding reputation for technical support.

And an OS company as reliable as its technology. Think how that could change your world, and your customers' world.

For the better.

www.qnx.com

QNX®
RTOS



Call about our eval system:

800 676-0566 ext. 2324



Build
a more
reliable
world™

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

You need a design that allows the embedded system to record and replay the data and the off-line software to analyze it.

of pre-processing are as follows:

- Application of calibration correction values
- Discarding of invalid data
- Associating data from one sensor with data from another
- Separating raw data into sets meaningful to the core algorithms
- Searching for data requested or required by core algorithms

The core processing software performs those functions or computations on the pre-processed data that meet your system's primary functional requirements. The specific software layer or layers to be tested with playback will determine from which data extraction (DX) point to obtain the sensor data for playback recording (see Figure 4).

Playback is not well suited for testing the outer layers of software that handle hardware devices. The nature of the processing doesn't meet any of the evaluation criteria for an embedded system's need for playback. The recorded data itself will add nothing to using simulated data. The issues at this layer have to do with the software/hardware interface, such as understanding how the device actually operates, meeting response time limits in reacting to hardware interrupts or other notifications, conditioning sensors to acquire the required data, and hardware error detection.

Realistically, there will probably be only one or two target candidates for playback testing and they may be in the pre-processing or, almost certainly, in the core processing layer. The candidates for playback testing are, by definition, those that would most profit from the use of real-world inputs to be fully tested and produce the most important outputs. These functions are at the heart of your system and represent the central reason that the

system was created. You must record the data these functions require.

There was only one portion of our EW system software that was a candidate for playback: the emitter signal processing software, the core processing layer. Our software design had all the data required by the signal processing software placed in a single data structure we called a "dwell buffer" (a dwell is defined as a particular configuration of the radar receiver hardware). Obviously, recording the dwell buffers was mandatory. The decision to target the core processing layer meant that we could use DX point 6 (see Figure 4) where the dwell buffer exited the core processing layer, and log every dwell buffer that passed that DX point. This simplified the job of recording since all the data of interest was in the dwell buffer.

Regardless from which DX point you decide to obtain the target environmental data, you should still log events of interest that occur in any layer. Just because you don't intend to present playback data to a particular software layer in no way rules out the idea of logging events of interest that occur anywhere in the system. These events can still be very useful to off-line analysis of system behavior.

Identify the off-line analyses. What off-line analyses will be performed on the recorded data files? Suggested off-line analyses topics are as follows:

- System input validity
- Input data rates
- Error rates
- System output validity

However, you can analyze the data for any characteristics that are of interest to you. You may want to scan the recorded data files for errors in data supplied by another system. This could be especially useful during system integration.

An off-line program to evaluate the timing of significant events sometimes produces surprising results and may pinpoint areas where your system is not meeting its hard real-time requirements. Also, examining the type of errors that are logged may point to a faulty hardware device. Other curious anomalies such as excessive time to complete certain operations may be revealed with this technique. The things you can search for off-line are limited only by your budget and the data present in the playback files. But, if you didn't record it, you can't analyze it.

Some errors are not visible during normal operations. For example, our system was required to locate an emitter to within a specified percent error. Except for large miscalculations, an observer of our emitter location display couldn't say with confidence that an emitter was within the specified limit or not. Answers such as these were available only by running an off-line analysis program against the recorded data.

Playback design topics

You need a design that allows the embedded system to record and replay the data and the off-line software to analyze it. There are three primary aspects to designing playback into an embedded system:

- Playback file structure and record formats
- Playback file recording
- Presentation of playback data to embedded software


Whether you're designing from scratch or retrofitting an existing system, the issues are the same. The following sections offer some guidance in these design areas.

Playback data design

A database design is needed that captures every aspect of system operation required by both the embedded system and the off-line software.



If you think
a bad date is painful,
try picking up the wrong
DSP partner.

 **TASKING**
**The Embedded
Communications
Company**

Or play it safe and choose TASKING as your partner for leading-edge DSP software. You'll work with some of the world's foremost DSP experts, reduce your DSP development time and increase your DSP application development performance.

TASKING attracts you with an integrated development environment for editing, compiling, building and debugging. TASKING charms you with C/C++ Compiler support for special addressing modes and C/C++ Compiler optimization. TASKING wows you with multi-core debugging, predefined and programmable graphical signal data analysis. And TASKING wins you over with program performance analysis functionality and support for leading DSP real-time operating systems.

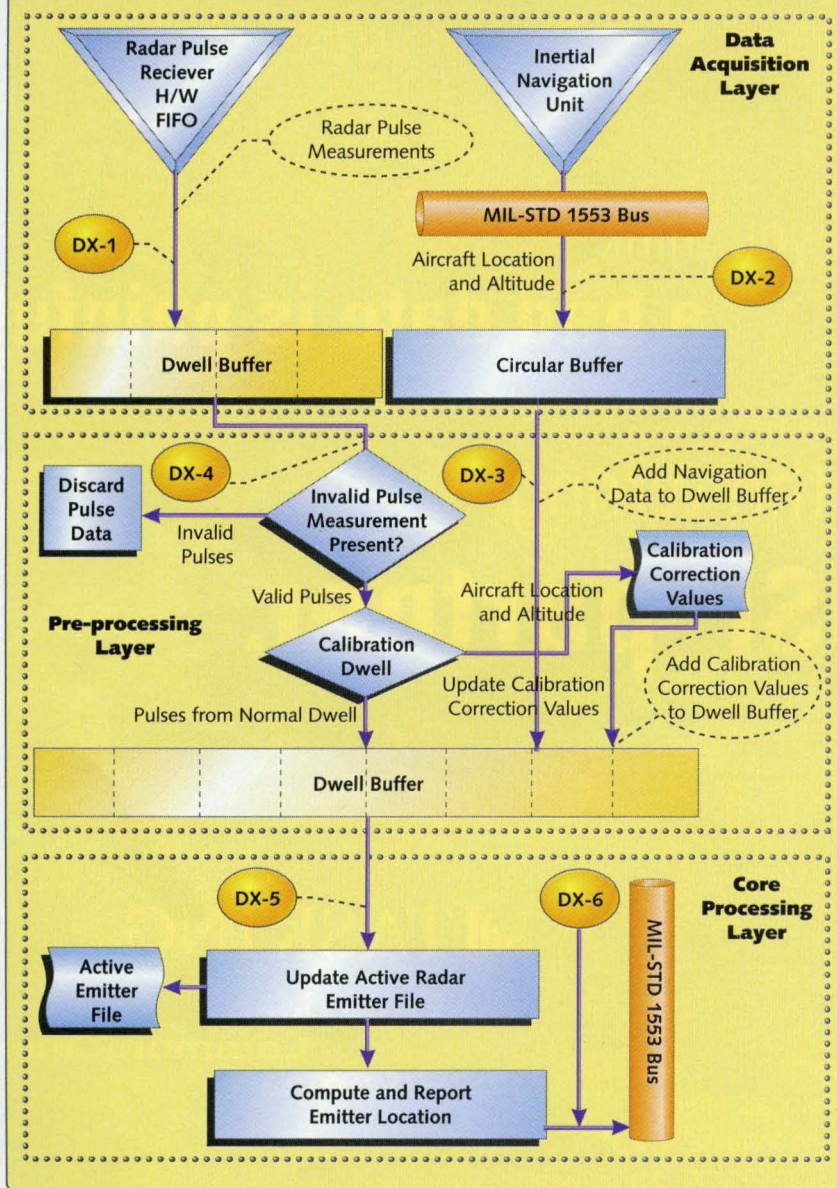
Isn't it about time you and TASKING had a real-time relationship? For more information, visit www.embeddedcommunications.com or call 1-800-458-8276 today.

www.tasking.com

A playback file for a system of even moderate complexity will have tens or possibly hundreds of record types stored within it.

playback files being misinterpreted, perhaps in terribly subtle ways, and lots of wasted engineering time and data.

FIGURE 4 Processing layers and data extraction points in an embedded system



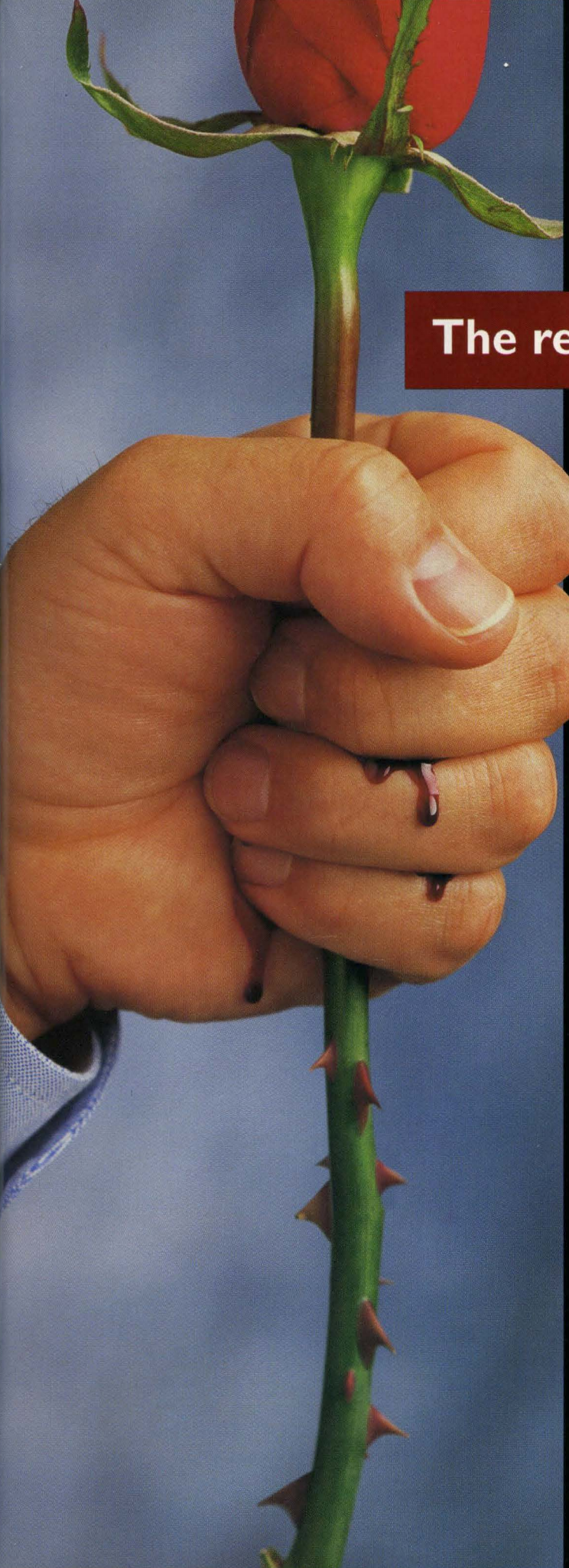
Capture the hidden context. Translate contextual information into explicit message contents. For example, the fact that a particular message is sent over a certain inter-process queue implies the identities of the two processes on either end of the queue. If the identities of the sending and receiving processes are important in interpreting the logged message then you must design in the ability to record a process' identity. The method by which you encode contextual data must allow meaningful interpretation at a time or by persons far removed from the development phase of the project. In other words, recording a random number the RTOS assigned as the process ID is not sufficient. You must log data, the use of which permits translation of that process ID into something meaningful, like a process name.

Playback file space budget. Estimate the disk space requirements of the playback files based on the size of individual data records, frequency of recording various record types, and duration of record mode. Don't skip this step. Playback file sizes can easily grow to hundreds of megabytes and beyond. You must have a fairly accurate estimate of the upper bound of these file sizes early on in order to provide sufficient recording capacity. Failure to do so may result in a system that fails to record sufficient data and can invalidate all your playback efforts.

Data record identification and versioning. A playback file for a system of even moderate complexity will have tens or possibly hundreds of record types stored within it. Obviously, each record class will be assigned a unique type code. Something not quite so obvious and much easier to ignore is

the fact that, over time, the format of some of these records may require change. Therefore, each record must also have a version number. This allows software reading the record to adapt to multiple versions of the same class of record. Failure to provide this mechanism will result in records in

Time synchronization data. In many cases, data collected from multiple sensors can only be processed correctly if the exact time of each sensor collection is known. For example, in our EW system we obtained aircraft navigation data and radar pulse data from two different sensors. The pulse data



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.

Identify hardware calibration values that are input to the software and make sure that these values are output to the playback file.

such a big deal. Each playback file should contain information such as the following:

- Test session identification (for example, flight 10 test sequence 3)
- Date and time of test start and end
- Location of test
- Purpose of test
- Outcome of test
- Miscellaneous remarks

This sort of information may have to be added after the field trial but some mechanism for recording this data should be designed in to your file structure.

Record calibration data. Identify hardware calibration values that are input to the software and make sure that these values are output to the playback file. Failure to do so will virtually guarantee that you will never achieve the same results from a replay session as compared to normal operation.

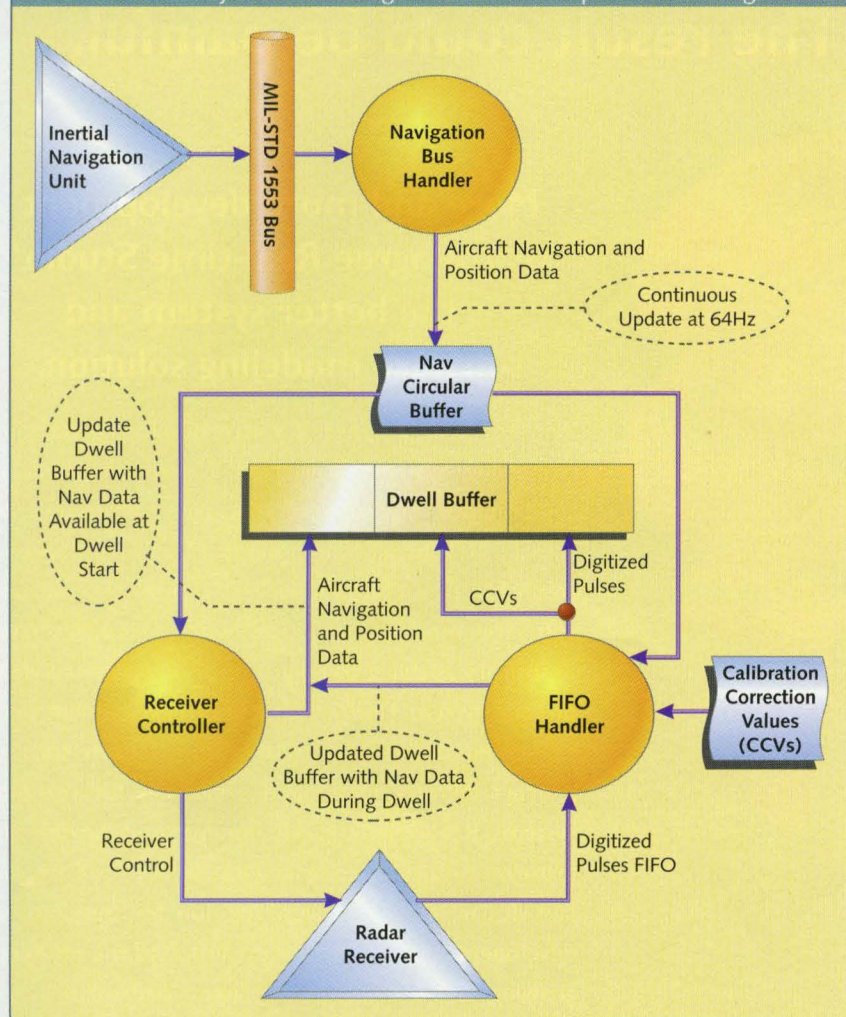
Recording playback files

The design for recording playback files must address the following issues:

- Identify DX points within the embedded software
- Specify a detailed message format for each message type
- Design a mechanism for transferring messages from the DX points to the recording software
- Design a way to enable and disable the recording of all or selected message types
- Design a mechanism for recording the logged messages on permanent storage (disk, CD)

The set of DX points are determined by the requirements developed in the preceding "Playback Data Requirements" section. As mentioned before, the DX points determine the nature of the data to be recorded and are a function of what sections of your embedded software are to be tested and what off-line analysis will be done.

FIGURE 5 EW system block diagram in a normal operational configuration



couldn't be processed accurately unless the precise times of both the pulse and navigation measurement were known. This requirement would be present with or without playback. However, if playback is to succeed on your system then you must make sure that the data that is the basis for time synchronization is recorded.

The typical solution to this problem is to adopt a universal measure of time (like Coordinated Universal Time) as the "system time" and time stamp every message with the system time at the instant the event occurred.

Note, that the time stamp indicates when the event occurred, not when the message was sent, received, or queued.

Playback file header data. A playback file may contain thousands of recorded events but if you can't remember the larger context of the field trial it may prove difficult to get maximum use of the file. Also, on a project of long duration you may have tens, if not hundreds, of playback files and you won't remember what the file "big_test.dat" contains or why it was

IF YOU USE:

ATI Nucleus™

Express Logic ThreadX®

Kadak AMX™

Precise MQX™

Wind River VxWorks®

Your own custom RTOS

ON:

ARM

StrongARM

ARC

PowerPC

**Contact
MetaWare for an evaluation
www.metaware.com/eval**

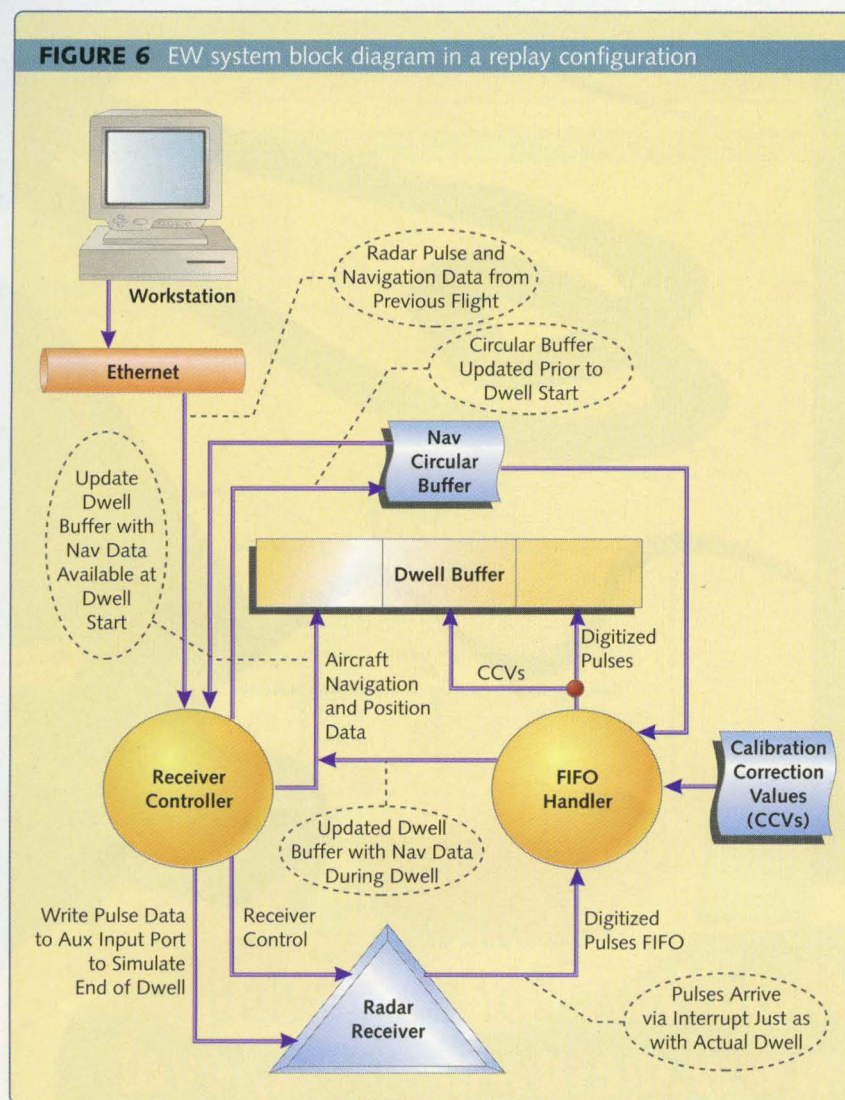
If time to market is important to you...
If performance, quality and excellent customer service matters to you...
then you need to evaluate the MetaWare tools!



2161 Delaware Avenue, Santa Cruz, CA 95060 • Toll-Free: 1.877-TOOLSET (866.5738)

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

FIGURE 6 EW system block diagram in a replay configuration



have to read everything you record to get any use of it.

Finally, a means by which the extracted data is actually recorded on a permanent storage device must be found. The recording channel must be able to handle the sustained peak recording load and have sufficient capacity to hold all the data from field trials of the longest estimated duration. This is critical since failure of your recording channel or media to meet these requirements will undermine your efforts by being unable to generate a complete and reliable playback file.

Use a debug channel for playback.

Your target platform may never be deployed with a network hardware device, but if you use one for debugging then you're in luck. It is hard to imagine a solution simpler than using TCP/IP sockets over Ethernet or other compatible media. If you're using a TCP/IP-based debugger then your target RTOS and development station almost certainly support socket programming (the API to TCP/IP). You'll need to host the playback file on the development station and modify your embedded code to log the file via sockets or perhaps the network file server (NFS) if that seems a better choice.

Each DX point will have one or more message types available to it. The format of each message must be specified in detail, in accordance with the principles set down in the “Playback Data Design” section. Estimate the number of messages of each type recorded per second for each DX point. Use these figures, along with duration of record mode for an estimate of your recording bandwidth and storage requirements.

Obviously, messages from the DX points must somehow be sent to a central point for actual logging to permanent storage. Your choices depend on the sophistication of your RTOS and

system architecture. In most cases, simply creating an inter-process queue will suffice. More complex systems may elect to use sockets as a transport mechanism.

You will want a way to enable or disable the recording of messages from the various DX points. At a minimum, you will want the scope of this function to include all DX points and all messages. Without this control your system will always be recording when operating. This almost certainly is not desirable. Despite the quantum leaps in disk storage capacity of recent years, it is not prudent to waste resources. And, regardless of disk capacities, you'll

Reuse an existing data channel for playback. Okay, so you don't have a

network interface card and can't use TCP/IP. When the going gets tough, the tough get expedient. Cast around for a data channel that you could either share or otherwise utilize, perhaps a medium to high-speed bus or channel only used for backup or other rare events or modes. We once used a MIL-STD 1553 bus to log DX data. It was slow (only 1 Mbit/s) but it was good enough. We equipped a PC with a MIL-STD 1553 PCMCIA card and were able to use the channel for both playback and record modes.

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!



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>

Flush often. Imagine this: you've just executed your field trial, collected 20 minutes worth of very valuable target environment data and your system crashes because of a power glitch on the aircraft (or whatever platform you're testing on). You reboot and, to your horror, the precious playback file has zero blocks allocated. Why? The file was neither closed nor flushed so the file allocation table wasn't updated. Suggestion: periodically issue a flush to the file during record mode. The period should be a little shorter than the maximum amount of data you can afford to lose.

Substituting playback data for sensor inputs

The sensor data from a playback file must be substituted for sensor device inputs and presented to the embedded software, while running in the target system. This is a fundamental

requirement for a playback-equipped system. It can also be one of the more interesting playback design challenges. If you're lucky, you can solve this problem in an elegant manner, but be prepared to do otherwise.

First, you need an input channel. The suggestions in the preceding "Recording Playback Files" section on reusing debug or other data channels apply equally to playback channels. Second, a way must be found to substitute recorded data for normal sensor inputs with minimum disruption to your existing software architecture.

Our EW system had been tested via simulated data files and so we already had a tested way of bypassing the sensor inputs and substituting the simulated data. Our embedded code ran under VxWorks while our simulation software ran under Unix. The two CPU boards were co-located in the same chassis. We had a TCP/IP chan-

nel implemented via shared memory and so socket communication was the answer. Our simulator sent its data to the embedded software via socket connections.

We leveraged the existing simulation channel for replay in the following manner. Figure 5 shows a block diagram of our EW system in normal operation. The central point of the pre-processing software was to join the radar emitter pulse data with the navigation data sampled during the dwell and put it all in one dwell buffer. Note that although the navigation data is continuously coming in at a 64Hz rate, it is only stored for processing during a dwell.

Figure 6 shows our system configured for replay. In this configuration the live 64Hz navigation data was replaced by the recorded navigation data from the playback file. The recorded navigation data was written to the circular buffer as if it had been received from the 1553 bus. The recorded radar pulse data was substituted for the real thing by writing it to an auxiliary input channel on the receiver hardware. The pulses then appeared in the FIFO and generated an interrupt just as if they had been received from the A/D converter. The great benefit of this technique was that only one process was modified: the Receiver Control process. None of the other processes required modification because all the interfaces remained the same.

The time investment we made to equip our EW system with playback was repaid many times over. Playback allowed us to both meet our flight test goals and to collect and record valuable target environment data. We could not have succeeded without it. I hope your experience with playback is as good.

esp

Wes Howl is a staff scientist at Litton Advanced Systems Inc., where he has been creating embedded software for air-traffic control voice communications systems and electronic warfare systems for nearly 20 years. Contact him at wes_howl@littonas.com.

Your RTOS Should Be *LEAN* and *MEAN*...



CMX-RTX™: Fastest Context Switch Time,
Lowest Interrupt Latency and Smallest Footprint.

- TCP/IP for 16-/32-bit CPU's
- No Royalties • Free Source Code

CALL TODAY FOR FREE WHITE PAPER:
"RTOS: Buy or Roll Your Own?"

**CMX
COMPANY**

8051
8051-XA
80C251
80196/296
80x86
80C16x
HC08/11/12
H8/H8S
68K/683xx
TLCS-900
AVR
M16C
SH
PowerPC
SHARC
TI DSP's
Many More!

508.872.7675

cmx@cmx.com

http://www.cmx.com

AIM TO DELIVER THE BEST PRODUCT ON THE MARKET?

So Do We. In Fact We Already Have.

THE BEST DEVELOPMENT ENVIRONMENT

ANYWHERE ASPEX is a fully integrated, completely open development environment offering you the tools to develop, integrate and optimize embedded software for today's complex products. If you've found other development environments slow, buggy and prone to crashes, you'll find ASPEX just the opposite.

A TOOLKIT FOR TODAY Mixed processors, multiple processors, systems-on-chips. Intensely increased integration. That's today's embedded software reality and ASPEX was designed from the inside out to seamlessly support it all. Make it smaller, make it cheaper, make it faster, make it better. ASPEX can make it happen.



SUPERIOR DEBUGGING

CAPABILITIES All this multiprocessing and integration vastly increases the complexity of software debugging. But ASPEX can handle it. From a single event-driven debugger, ASPEX lets you simultaneously debug multiple target systems whether the processors are all the same or of different architecture.

EASY TO LEARN, EASY TO USE ASPEX provides a uniform, graphical interface that tightly integrates all tools and lets you manage projects graphically instead of learning cryptic command strings. Our unique Extended Target Visibility feature actually lets you see more target resources. Our Trace and Analysis feature helps you track down hard to find problems.

ASPEX supports ARM, Analog Devices, DSP Group, Intel Strong ARM and Texas Instruments.

Let ASPEX give you the tools you need to meet the challenges you face. The tools to develop, integrate and optimize embedded software for today's increasingly complex products. We're experts in embedded development tools and it shows in every aspect of ASPEX. Test drive ASPEX for yourself and experience features our competition hasn't even dreamed of. Download your free, fully functional evaluation copy of ASPEX at www.allant.com.



Allant Software Corporation

1280 Civic Drive, Suite 206, Walnut Creek, CA 94596 • 925.944.9690 • fax 925.944.9612 • email info@allant.com • www.allant.com

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.

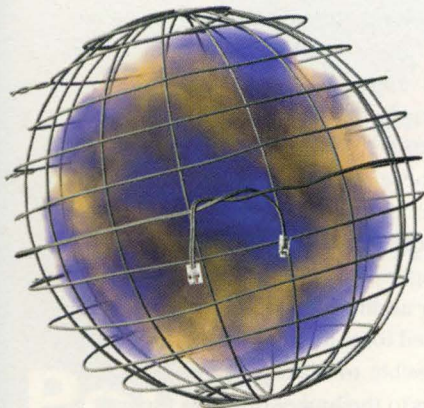
California Corporate Headquarters:
tel: 800.541.9508 fax: 805.499.5512
European Headquarters:
tel: 44.1494.432.735 fax: 44.1494.432.728
Japan/Asia-Pacific Headquarters:
tel: 81.3.5669.7722 fax: 81.3.5669.7723



Pacific Softworks
our wave... your Success.



Michael Barr



Hold Everything

One of the biggest challenges columnists face is coming up with something new to write about month after month, year after year. I thought I had solved that problem for myself, for a while at least, when I announced a protracted discussion of UDP/IP implementation details last month ("TCP/IP or Not TCP/IP?" April 2000, p. 49). I figure topics in and around UDP/IP are good for more than a half-dozen columns over the course of the next year. However, I did not properly anticipate a major problem with this topic before announcing it: the need for a plan of attack.

If I'm going to write an embeddable UDP/IP stack, I'll need an embedded platform to work with. Such a platform should have a processor, an Ethernet controller chip, a pair of serial ports (one for debugging, the other for SLIP and PPP, to be discussed later), and a couple hundred kilobytes of RAM and ROM (maximum, not minimum). I will also need to find development tools that will make it easy to write and debug this software in C. In addition, I should also make a decision about whether or not I will include an RTOS or kernel of some sort. All of these are things I should have decided before I announced the discussion! So, please bear with me—as I stall for time.

My thinking on the latter issue, the one about the operating system, is definitely to use one. In my experience, it is the rare embedded system that finds a need for networking but not multitasking. Given that I'm going to be using an OS and writing about the experience in a magazine that sells

advertising space to numerous RTOS vendors, I'd prefer to go with something "open source." This removes any potential for a claim of favoritism.

One possible choice for an open-source RTOS is eCos (sourceware.cygnus.com/ecos/). The problem with eCos is that with my current setup I require Windows-hosted development tools. eCos expects to be built with the GNU Compiler (GCC), which runs far better on Linux than on Windows (trust me, I've run it on both before), and is currently ported only to processors with relatively expensive Windows-based development tools (ARM and PowerPC, for example).

I'd much prefer to work with a 16-bit processor on a commercial off-the-shelf board, developing the software with a Windows-based compiler, assembler, linker, and debugger. That would keep my costs down, allow me to show the reasonableness of including networking support even in a mid-range embedded system, and make it possible for others (yourself included) to duplicate and verify my work.

One possible configuration—and the one I'm currently favoring—is to use a Net186 prototype board as the target (www.amd.com/products/lpd/186es/21780a.html) with µC/OS as the RTOS (see Jean Labrosse's *MicroC/OS-II*, R&D Books). But I definitely need more time to consider the issues involved, purchase the target and tools, and get my lab set up than I have before this month's column is due.

Fortunately, another great topic for a column just popped into my head a few days ago. It's best to get this one down onto paper while it's

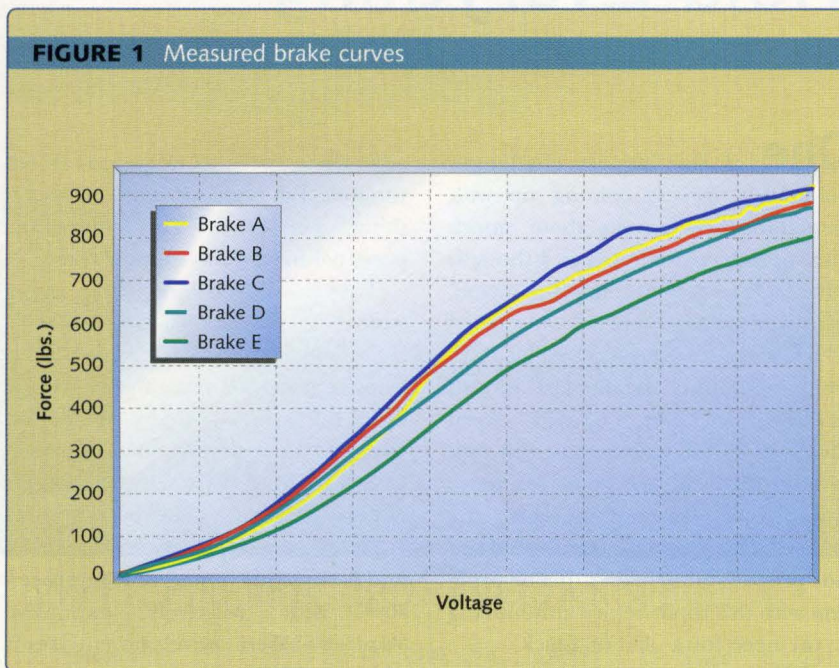
Internet Appliance Design

in this issue

- 61** Web by Proxy
- 69** Exposing MIB Data to a Web-Based Interface, Part 2
- 83** Real-Time Java Wars Yield Uneasy Truce
- 89** Embedded Internet Tools

One problem with such a closed-loop approach is that it takes time to achieve the desired results. If you close too quickly, you'll continually overshoot and undershoot the desired force.

FIGURE 1 Measured brake curves



still fresh, so this all worked out quite nicely.

Calibration

I once ran into a difficult problem involving run-time calibration of real world (that is, imperfect) components. A client was developing a piece of exercise equipment that was to provide a variable number of pounds of force feedback to a human operator. To support the complete range of possible force settings with a given worst-case accuracy, the product was correctly designed around a microcontroller. The embedded software that ran on this microcontroller was principally responsible for regulating the voltage going to a brake such that the brake's calipers would "slip" precisely at the selected force. In other words, the human operator must push or pull against the shaft with at least the selected amount of force in order to

make the shaft move. But the problem did not turn out to be as easy as it at first appeared.

What made this problem so difficult was that each brake performed differently in actual use. It turns out that there's no one equation that says: to achieve x pounds of force, apply $v = f(x)$ volts. (The input to the brake was actually a PWM signal and the important characteristic was current; I'm using voltage to make the discussion simpler.) Figure 1 contains five S-shaped curves. Each of these curves is the measured response of a particular brake. And, as if the differences between brakes were not enough to deal with, the curve for each particular brake is affected by its age (as the contacting surfaces wear), temperature, and even the velocity of the force applied against it. (All of these are factors that alter the amount of friction between the surfaces.)

Perhaps the first and most obvious

step to take is to create a feedback loop in the system. It's clear that simply setting the voltage on the brake and forgetting it is not going to be good enough. As you can see from Figure 1, the difference in force output for a given voltage between brakes can be extremely large. Some of the brake curves even cross, indicating that individual brakes are not simply stronger or weaker than the average.

By adding sensors that can read the applied force and the shaft's position, it is possible to make continuous adjustments to the brake voltage as the user is in the process of pushing or pulling on the shaft. If the shaft is moving (its position is changing) and the force on the shaft is too small, the brake voltage must be increased. If the force is too high, the voltage must be decreased. By "closing" toward the desired force like this, it's possible to achieve and maintain that force throughout each repetition of the exercise.

One problem with such a closed-loop approach is that it takes time to achieve the desired results. If you close too quickly, you'll continually overshoot and undershoot the desired force. So you must only make one small change in the voltage in a given time period, then wait for the brake to respond before making the next change. In our case, we found the ideal period to be right around 50ms. At that rate of closure, you can be relatively certain of achieving the desired force without much overshoot, and maintaining it without too much fluctuation.

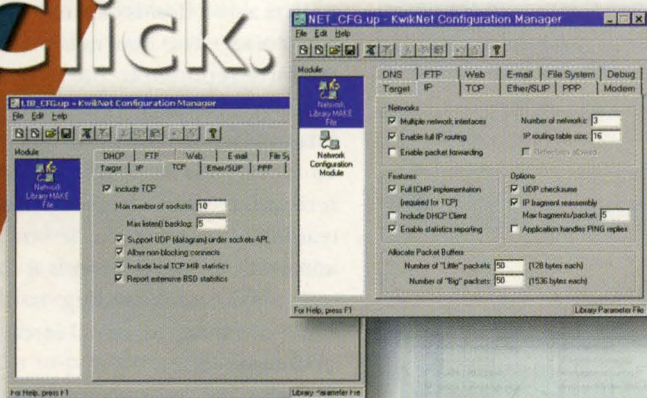
However, if you start out with an initial force that is so far from the desired force that it takes more than about half a second (500ms) to reach it, a human can generally feel the change in resistance. This seems to be true whether the incorrect initial force is above or below the desired force. So more than 10 increment or decrement operations had to be avoided at all costs. And the only way to achieve that was to make a fairly accurate initial "guess" about what the brake voltage should be for a given force. To get that

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

Before we can talk about calibrating the brakes, however, we must first talk about the force sensors. Each force sensor must also be calibrated.

FIGURE 2 Nominal brake curve

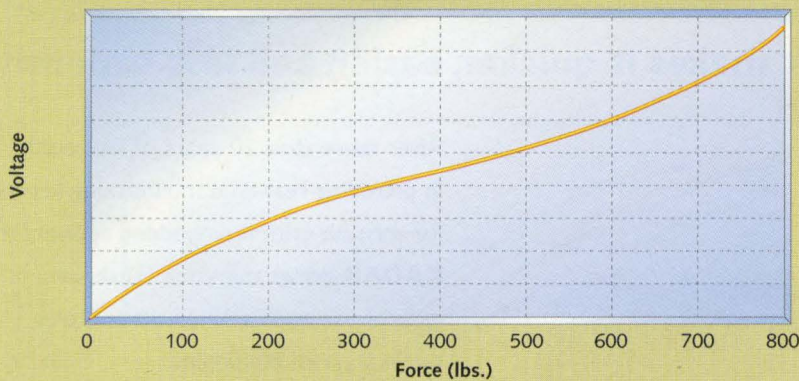
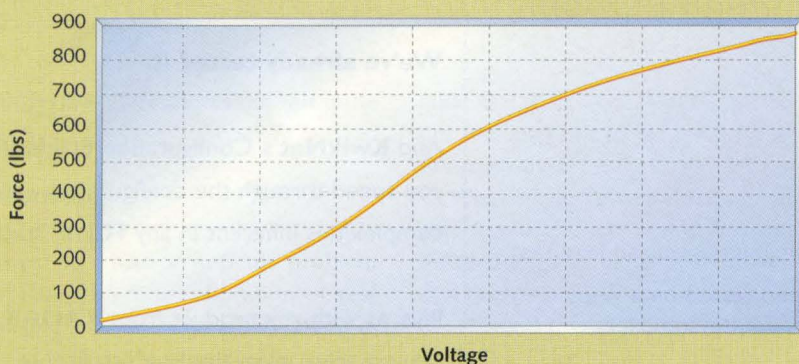


FIGURE 3 A polynomial approximation of the nominal brake



kind of accuracy, we had to calibrate each brake's S-curve when it was installed. And, because of the more subtle brake-specific factors mentioned previously, each brake must be periodically recalibrated in the field.

Linear simplicity

Before we can talk about calibrating the brakes, however, we must first talk

about the force sensors. Each force sensor must also be calibrated. Consider the example of a force sensor that always reads 10% below the actual force applied. The user will have to apply significantly more force (11.1% more, to be exact) than he or she expected. If you might be tempted to consider that a tolerable worst-case error, remember that it is possible the human user could injure (or, more

likely, reinjure) a muscle in the process of completing an exercise on such a strongly biased machine.

Fortunately, calibrating the force sensors is not terribly complicated. It turns out that these devices can only really be biased in one of three ways. (This from the manufacturer, of course.) First, they may have a non-zero reading even when no force is applied to them. We'll call this the sensor's zero offset. Second and third, the sensors may have a linear bias unique to forces applied in each direction. We'll call these the left and right gain factors.

The zero offset is easy to determine and correct for. You need only read the force reported by the sensor when no force is applied. That value is the zero offset for that particular sensor. To correct for it, simply subtract the zero offset from every future force reading. For example, if you determine that the zero offset is 5 lbs. and your next "raw" reading is 15 lbs., you'd compute the actual force on the shaft to be 10 lbs.

Once the zero offset, if any, has been subtracted out, determining the left and right gain factors is possible. The terms left and right are subjective, of course, but the general idea is that the sensor has one linear bias when forces are applied in one direction and a different linear bias when forces are applied in the opposite direction. The fact that these biases are linear simply means that the percentage error is the same, regardless of the actual force applied. So, for example, a sensor with a -10% bias would read 90 lbs. when 100 lbs. is applied and 180 lbs. when 200 lbs. is applied. It is, therefore, sufficient to take one calibration reading in each direction.

To determine the left and right gain factors, you might simply hang a 100-lb. weight from the shaft, first applying that force in one direction then the other. By comparing the actual readings to the expected readings in each direction, you can compute the

sensor's left and right gain factors. By dividing all future readings by the gain factor in the given direction, you can eliminate these biases from the sensor. The result is an accurate measure of force that can be compared across different machines and form the basis for calibration of the brakes.

Polynomial insanity

Calibrating a brake is much harder than calibrating a force sensor, for one important reason: each brake's S-curve is non-linear. You can't simply take one reading and compute a gain factor from it. This was possible for the sensors only because their error was a fixed percentage of their input value. The brakes, on the other hand, can diverge wildly from one another (and even from themselves, over time), so that there's really no such thing as a typical brake.

Despite the fact that there is no typical brake, we've still got to define one. After all, you need a starting point even just to perform a calibration. So we began by measuring the relationship between input voltage and output force for five randomly selected brakes. The results of these measurements were the five data sets shown in Figure 1. We then averaged the five forces resulting from each tested voltage setting to produce the nominal brake curve shown in Figure 2.

The only problem with the curve in Figure 2 is that it represents the inverse of the relationship the software needs to compute at run-time. The necessary relationship measures voltage as a function of desired force, telling the microcontroller how much voltage it should apply to the brake to achieve a particular desired force. Figure 3 shows the same nominal brake data, flipped on its axes and approximated as the third-order polynomial:

$$v = Ax^3 + Bx^2 + Cx + D \quad (1)$$

where A , B , C , and D are floating-point coefficients.

All we really needed to accomplish through calibration was to bring the curve within about 10 voltage increments or decrements of the optimal value, along as much of the curve as possible.

Jack Crenshaw has recently been looking to tackle a similar sort of problem in the pages of his column—that is, run-time calibration of a device with a polynomial response curve ("Curmudgeon Repercussions," September 1999, p. 19). However, his approach seems to be to actually determine new coefficients (A , B , C , and D) at run time, thus performing a complicated set of computations on the embedded processor. That sort of approach simply wasn't practical in this particular system.

What made more sense was for us was to try to "tweak" the nominal curve to make it a better fit for each particu-

lar brake. This was partly because we didn't have the computational resources to spare (it was only an 8-bit processor with a couple hundred bytes of RAM, after all) and partly because we were able to "close" to the desired force pretty quickly anyway. All we really needed to accomplish through calibration was to bring the curve within about 10 voltage increments or decrements of the optimal value, along as much of the curve as possible. In truth, given all of the other factors affecting the brake performance, we could never really hope for anything better. The actual brake curve will change, for example, as soon as the

Super Multimedia Power



PalmSize SBC Empowers TV-out Capability

\$433



Order Online
www.advantech.com

Automation with PCs
ADVANTECH

PCM-5822 Features

- Consumes 8 Watts power (STD mode) w/ 64 MB DRAM
- On board GXm LV-200 MHz CPU, fanless
- TV out function, NTSC & PAL format
- Supports 640 x 480 and 800 x 600 input resolutions
- On board VGA/LCD & Audio
- On board 10/100 Mbps PCI Ethernet
- One CompactFlash socket
- Size: 5.7" x 4"



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

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

For any given force, the actual brake curve may be above, on top of, or below the nominal curve.

surface of the brake pads heats up (even after just a few reps).

Here's how tweaking the curve might work in practice. Given a particular force setting, x , the nominal voltage, v , would be computed according to the polynomial in Equation 1. We would then apply a "tweak factor" that is also a function of the force. In other words, we'd apply a "local" tweak factor that is applicable only at or very near x lbs. Multiplying the nominal voltage by the tweak factor, we'd get a new voltage, v' , and apply that to the brake. The result, we hope, will be a force within 10 voltage increments or decrements of the optimal voltage. If it is, we've done good enough.

This, of course, begs the question of how to determine those "tweak factors."

Tweaking

The key to tweaking is the number of data points. If we had the luxury of setting the brake to every possible voltage and measuring the resulting forces, it is clear that we could produce an exact tweak factor for every point along the curve. For any given force, the actual brake curve may be above, on top of, or below the nominal curve. (Above means a higher voltage is needed to achieve that force on this particular brake at this particular time, below means a lower voltage.) If

it's above the nominal curve, the tweak factor would be a number greater than 1.0; if it's below, the tweak factor would be less than 1.0; and if the actual and nominal brakes concur at that force, the tweak factor would be exactly 1.0.

Unfortunately, measuring the force produced by every brake at every possible voltage is not realistic. (It was hard enough doing that once for each of the five representative brakes.) Since calibration must be done frequently and it is often the owner of the machine who will do it, it is necessary that the number of data points be significantly reduced.

At each calibration data point, a specific force (x_1) is requested, the microcontroller computes the nominal voltage for that force (v_1), the nominal voltage is applied to the brake, and the force produced is recorded. Interestingly, the processor now knows exactly what voltage to input to achieve the force that resulted—though it still doesn't know how to produce the requested force. In other words, if the next force requested were the one we had just produced (call it x_2) it would be very easy to "tweak" the nominal curve at that point, by simply multiplying the nominal voltage by the ratio of the two voltages (v_1/v_2). This ratio is the tweak factor at force x_2 .

By taking a number of these measurements and recording the ordered pairs ($x_2, v_1/v_2$) in a table, we gain the ability to produce perfect results at those specific requested forces. To extend these results to include other forces, we simply "draw" line segments between each known data point. In other words, we weight the two closest "precise" tweak factors to achieve estimated tweak factors for any force in between.

Listing 1 contains a function, `adjustVoltage()`, that performs the voltage scaling mathematics. The data structure is an array of calibra-



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

tion data points. Each `CalPoint` consists of a force (x_2) and a ratio (v_1/v_2) that was observed when the force x_1 was most recently tested on this brake. These calibration points are assumed to be ordered in the array from the smallest force to the largest. In addition, the function assumes that the array begins with a calibration point at force 0 and ends with a calibration point at `MAX_FORCE`. (We found that the ratios associated with these two points should be set to 1.0 and the same as the highest actual calibration point available, respectively.)

In effect, we multiply the nominal curve by a connected set of line segments. The more data points you have, the shorter the line segments will be, and, therefore, the more precise the resulting calibration. This is a really practical way to calibrate a complex device quickly. In our specific case, we were able to take just five measurements during brake calibration and to achieve impressive results.

I haven't found much literature about doing this sort of run-time calibration, so I wonder how others have accomplished similar tasks in their embedded systems. If you have a war story or a favorite reference on the subject, I'd love to hear about it. Please drop me a line.

Oh right, networking

As if to point out how obtuse the idea of including even UDP/IP in an embedded system is, Bill Gatliff weighs in this month with an article about using a PC and a serial port to solve the tricky problem of web-enabling legacy systems. His solution, complete with sample code for Linux, is extremely slick. Since Bill can hardly bring himself to say the words *commercial software* these days, I did the Windows port myself. (Both sets of code are available at www.embedded.com/code.htm.)

LISTING 1 A function that uses the calibration data

```
typedef struct
{
    unsigned short force;
    double        ratio;
} CalPoint;

CalPoint aCalPoints[NUM_CAL_POINTS];

unsigned short
adjustVoltage(unsigned short force, unsigned short vNominal)
{
    int i;
    double scaleFactor;

    /*
     * Find the relevant calibration range.
     */
    for (i = 1; i < NUM_CAL_POINTS && aCalPoints[i].force < force; i++);

    /*
     * Compute the weighted scale factor.
     */
    scaleFactor = ((aCalPoints[i].force - force) * aCalPoints[i-1].ratio
                  + (force - aCalPoints[i-1].force) * aCalPoints[i].ratio)
                  / (aCalPoints[i].force - aCalPoints[i-1].force);

    /*
     * Adjust the voltage.
     */
    return (vNominal * scaleFactor);
} /* adjustVoltage() */
```

Also in this month's Internet Appliance Design section, Kedron Wolcott finishes his two-part discussion of bringing legacy SNMP-enabled systems into the realm of web-based management. How did embedded programming get so complicated so quickly?

Next month I'll get started on that UDP/IP stack I've promised you. Honest. In the meantime, stay connected....

esp

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

SOLUTIONS TODAY... FOR TOMORROW'S EMBEDDED TECHNOLOGY

Let EBSnet help you build your next
embedded device!

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

Prompt Competent Support

- Consulting
- Satisfaction Guaranteed!

For More Information,
Visit Our Web Site

www.ebsnetinc.com

1-800-428-9340

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

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

EMBEDDED TCP-IP NETWORK STACK

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

INTERNET PROTOCOLS

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

DEVICE DRIVERS

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

KERNEL DRIVERS

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

PORTING LAYER

Ports easily to any realtime OS/CPU

DIAL-UP

SLIP, CSLIP, PPP, modem support

APPLICATION PROTOCOLS

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

WEB SERVER

Full featured embedded Web Server (with diskless option)

EMBEDDED WEB BROWSER

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



BILL GATLIFF

Web By Proxy

Adding a TCP/IP and web server stack to an embedded system is an expensive proposition. If your product already communicates, it may be better and cheaper to use a proxy.

It goes without saying that Internet-enabled devices are all the rage these days. A few short years ago, the only mainstream embedded users of the Internet were set-top boxes and network infrastructure equipment. Today, on the other hand, everybody wants to interact with every gadget they own via a web browser, and most of them can provide rational reasons for doing so.

But to most embedded devices, the Internet doesn't come easy. What do you do when your company's main product doesn't have a network port? How can critical applications like industrial controllers be placed online, without disrupting their primary functions? And what about the hordes of existing devices that don't have the resources to support TCP/IP and other Internet protocols?

In some of these situations, an HTTP proxy may come to the rescue.

What is a proxy?

Simply put, in a networking context, a *proxy* is any program that provides a communications bridge that other applications can use to exchange data. Proxies are widely used to help protect applications from each other, as in the case of a network firewall. Our situation, however, illustrates another popular use for proxies: as translators between applications with seemingly incompatible communications strategies. Such a proxy can bring the Internet to an embedded system, while allowing the embedded target to speak its native tongue.

Proxy implementations come in a variety of shapes and sizes, which makes them difficult to present comprehensively in a single article. The fundamental concepts are the same in almost all cases, however, so even the relatively limited treatment I provide here will be useful in a more general setting.

An HTTP proxy, as I present it here, is a program that implements a browser's HTTP requests for data using one or more proprietary message exchanges with the target embedded device.

LISTING 1 A simple "home page" for your product

```
int
parse_http_request(char *http_request, int connfd)
{
    const char *header = "http/1.0 ok\ncontent-type: text/html\n\n<html>";
    const char *footer = "</html>";
    char *target_timestr;

    write( connfd, header, strlen( header ));
    target_timestr = proprietary_localtime();
    write( connfd, target_timestr, strlen( target_timestr ));
    write( connfd, footer, strlen( footer ));

    return ;
}
```

For the remainder of this article, I will assume that the motivation for web-enabling a legacy device is to allow a customer to interact with the product using an ordinary web browser. This assumption allows us to focus on a single kind of proxy, one that can translate between HTTP and the target device's own, proprietary protocol.

What is an HTTP proxy?

An HTTP proxy, as I present it here, is a program that implements a browser's HTTP requests for data using one or more proprietary message exchanges with the target embedded device. Once this exchange is complete, the proxy returns the result to the client as an HTML document, or some other kind of browser-friendly format like PNG, JPG, or even raw ASCII text.

The proxy executable is placed at the most convenient point between the client and the target, depending on the desired capabilities of the overall solution. In most cases, the best location for the proxy is on the PC running the browser, especially the case when the target doesn't support Ethernet, or access is needed only when the client is standing next to the product.

When something resembling true Internet-wide connectivity is necessary, however, the proxy can be installed on an inexpensive, single-board computer located between the target and the target's link to the network.

Why a proxy?

The traditional approach to putting a device "on the Internet" is to add TCP/IP and various other capabilities to the target itself. While this approach has its advantages, it is usually an unreasonable option for mature embedded products, particularly those that lack the necessary hardware interfaces, spare memory, or processor cycles.

Proxies enable Internet-style communications with legacy hardware without modification of the target application (of course, the target must support some kind of communications capability beforehand). The proxy application runs on a computer located somewhere between the client's browser and the target, and uses the target's native tongue to extract information to send back to the web browser. As a result, the target device has no idea that it has been Internet-enabled.

A proxy-based solution is more flex-

ible than an embedded system that speaks IP directly. Because it doesn't need to physically coexist with the target application, a proxy can support the overhead necessary to present a uniform user interface for multiple target versions. In addition, the target device's visual interface, as shown on the client's browser, can be changed without taking the target system out of service simply by upgrading the proxy application.

A proxy also permits communication with targets that don't offer a connection medium normally associated with IP protocols. For example, an HTTP-to-CAN proxy could be used to provide browser access to a target that had only a CAN port.

HTTP 101

Obviously, an understanding of how web browsers communicate is needed before we can use a browser to interact with an embedded target via its HTTP proxy. I'm not going to try to train you for a new career as a web server designer in this section, but I will try to cover all the basics.

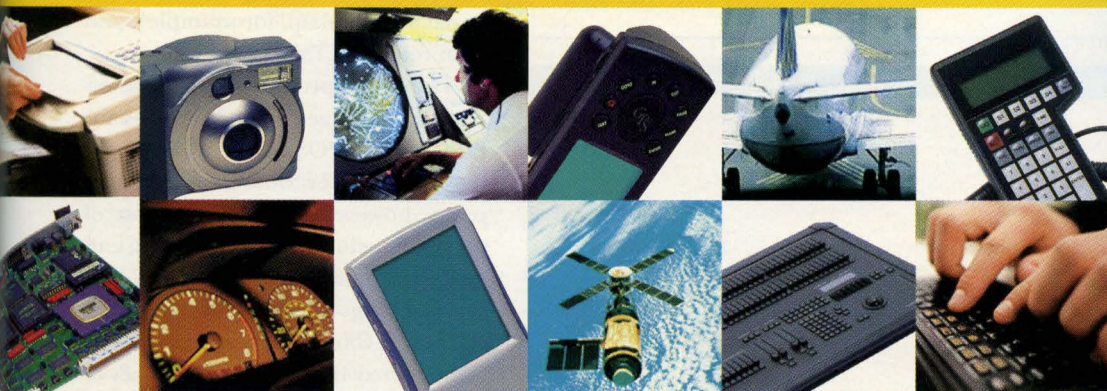
Contrary to popular belief, your web browser's primary language is actually HTTP, not HTML. When you type in a URL like <http://www.embedded.com/index.html>, for example, your browser sends the following HTTP message to the web server on the machine named *www.embedded.com*.¹

```
GET /index.html
```

A typical web server's response to this message is to return the contents of an HTML file named *index.html*, but this isn't always the case. In fact, the particulars of the response are left entirely to the server, and sophisticated ones like Zope (www.zope.org) routinely break the conventional notion of a one-to-one mapping between URLs and file names on the serving machine (in Zope's case, this divergence is a good thing).

Moving on, when you fill in some text and then click on a button in an

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

With a proxy-based solution, the key to connecting an embedded device to a browser lies in the ability to translate between HTTP and whatever language and media the target system supports.

LISTING 2 A more sophisticated page

```
typedef struct
{
    char *method;
    char *object;
} http_request_T;

int
parse_http_request(char *http_request, int connfd)
{
    http_request.method = strtok( http_request, " " );
    http_request.object = strtok( 0, " " );

    if ( strcmp( http_request.object, "/query?press=value" ) == 0 )
    {
        send_other_page( connfd );
    }
    else
    {
        send_home_page( connfd );
    }
}
```

LISTING 3 An HTML page with hidden values

```
<html>
<form method="get" action="help"
    <input type="text" name="textfield" value="type here" size=40>
    <input type="submit" name="pressme" value="Help!">
    <input type="hidden" name="state_id" value="1234">
</form>
<form method="get" action="more_info">
    <input type="submit" name="pressme" value="Conclusions">
    <input type="hidden" name="state_id" value="5678">
</form>
</html>
```

result is that the web server passes the message to a standalone application that performs a host-specific function (a database lookup, for example), and then returns HTML to the browser.

The HTTP protocol contains several other messages, including ones for PUTting and POSTing data. We don't need to consider those for our simple proxy, however, so in the interest of space I'll include references at the end of the article for further reading.

A basic example

With a proxy-based solution, the key to connecting an embedded device to a browser lies in the ability to translate between HTTP and whatever language and media the target system supports.

To illustrate one way to do this, I have developed a very basic HTTP proxy (supplied in `proxy.c`, available at www.embedded.com/code.html). To use this code, you must enhance the included `parse_http_request()` function to decode an HTTP message in a manner most suited to your needs, and then use the information the message contains to decide what to do next.

For example, let's say that all you want to do is provide a simple "home page" for your product that shows calendar time at the target device. To do this, you don't need to look at the arguments supplied with the HTTP request at all, because the response will be the same in all cases. Listing 1 shows how to do this, assuming you can use a function called `proprietary_localtime()` to get time information from the target.

To see this example in action, simply compile `proxy.c`, launch the resulting executable, and then supply the following URL to your browser:

`http://localhost/`

If your workstation already has a web server installed, try changing the definition of `LISTENPORT` in the example code to an unused port number

HTML form (the home page for an Internet search engine, for example), your browser sends a slightly different HTTP message to the server:

```
GET /query?textfield
=textdata&pressme=press_here
```

This message tells the server that you typed the word *textdata* into a field called **textfield**, and then clicked button **pressme** (which was showing the text *press_here* at the time) in an HTML form called **query**. As with the previous message, what happens next is entirely up to the server. Often the

(for example, 8000). Recompile, then connect using this URL instead:²

`http://localhost:8000/`

In any case, here is what the code in Listing 1 does:

- Provides an initial “okay” response to the client’s browser
- Calls the function that gets the local time from the target
- Builds an HTML page that contains the response, and
- Sends that response back to the client’s browser

A more sophisticated example

Let’s now suppose that we want the target’s home page to contain a button that the user can click to get more information from the target. This requires more intelligence in `parse_http_request()`, because we have to:

- Send the client the home page with the button, and
- Determine which button the user pressed and respond accordingly

Code to demonstrate this is included in `proxy.c` as well. The general idea is shown in Listing 2.

This code is doing essentially the same thing as the previous example, except that it is choosing which page to return based on whether the HTTP message says the user clicked on the button labeled **value** or not.

Hidden values and proxy simplification

The previous examples are straightforward, but they won’t scale very well to applications with more than a handful of pages. The reason is that `parse_http_request()` requires specific parsing code for each page, something that quickly becomes tedious and error-prone for anything beyond the simplest functionality.

When an application with many

LISTING 4 A state-based implementation

```
const http_state_T http_states[] =
{
    { 1, home_state },
    { 1234, page_1234 },
    { 5678, page_5678 },
    { 0, 0 }
};

void
parse_http_request(int connfd, char *http_request_buf)
{
    http_request_T http_request;
    char *state_idstr;
    int state_id;
    int wstate;

    /* make sure it's a "GET" message; if it isn't,
       we don't know what to do with it right now */
    http_request.method = strtok( http_request_buf, " " );
    if( strcmp( http_request.method, "GET" ) == 0 )
    {
        /* crack apart the rest of the request */
        http_request.object = strtok( 0, " " );
        http_request.protocol = strtok( 0, " \r\n" );

        /* find the "state_id=" portion of the message */
        state_idstr = strstr( http_request.object, "state_id=" );
        if( state_idstr )
        {

            /* get the number that follows "state_id=" */
            state_idstr = strchr( state_idstr, '=' ) + 1;
            sscanf( state_idstr, "%d", &state_id );

            /* Look it up */
            for( wstate = 0; http_states[wstate].id; wstate++ )
            {
                if( http_states[wstate].id == state_id )
                {

                    /* found it! invoke the state function */
                    http_states[wstate].state( connfd, http_request.object );
                    break;
                }
            }

            if( http_states[wstate].id == 0 )
            {
                error_state( connfd );
            }
        }

        /* there wasn't a "state_id=" in the message;
           default to the home page */
        else home_state( connfd, 0 );
    }

    return;
}
```


HTTP proxies are a simple and powerful way to get a legacy product onto the Internet, but they do have their limitations.

pages is desired, HTML's hidden values are the preferred way to manage the complexity without the drudgery of lots and lots of parsing code.

Listing 3 shows the source for an HTML page with two forms, each containing a unique hidden value and a single button. When the user clicks one of the buttons, the browser includes the associated hidden value in the HTTP message, which makes it a convenient way for the proxy to determine what to do next.

When the user clicks on **Help!**, the browser sends `state_id=1234` to the proxy. Likewise, when the user clicks **Conclusions**, the browser sends `state_id=5678`. The code in Listing 4 extracts the value of `state_id` from the HTTP message, and then looks up and invokes the state's associated function to generate the proper response. This code is included in `proxy2.c` (also available on www.embedded.com/code.html).

To add a new page to the application just add its associated `state_id` value and function to `http_states[]`, and then adjust the contents of the referring page to deliver this value to the proxy at the proper time (when the user clicks on a button, for example). In other words, you no longer need to modify `parse_http_request()` when a page is added.

The `http_states[]` table is a kind of "site map" for the entire application that `parse_http_request()` uses to move the client through pages in the appropriate order. From another perspective, `http_states[]` is a state machine that drives the behavior of the proxy in response to user events encoded in `state_id` values.

Whatever your interpretation, it should be clear that a state-driven proxy architecture makes it far easier to manage applications with a lot of pages than anything else I've shown you so far.

But can't I do all of this with CGI?

Yes and no. The examples shown here include portions of web server functionality that most CGI applications don't have, in particular the ability to receive HTTP requests from an IP port via `bind`, `accept`, and `read`. As such, our proxies can run on machines that lack a web server, which would be the case for most of your client's PCs.

On the other hand, a CGI-based approach makes sense when you need a proxy that can run on different kinds of hosts, or there is the possibility that the proxy will run on a host that is already running a web server. In the event that you produce a CGI proxy but a web server isn't available, the example proxy in this article can serve as a minimal web server that forwards HTTP to the proxy via an `exec` or similar system call.

Disadvantages of proxies

HTTP proxies are a simple and powerful way to get a legacy product onto the Internet, but they do have their limitations. To begin with, the proxy must be properly installed and running somewhere before communication with the target system is possible. In contrast, for targets with integrated Ethernet and HTTP/TCP/IP capabilities, the user only needs to plug in a cable and type in a URL.

A standalone proxy also does nothing to assure that the target interfaces it uses are properly maintained. A compiled-in HTTP server, in contrast, will likely produce compilation or link errors if a function it needs is accidentally removed from a new version of the target's application.

Finally, successful proxies require some knowledge of the host's networking and other APIs, which may present problems for developers with no skills in this area. I consider this an item of minimal concern, however, given the number of excellent TCP/IP

and other networking books available in the mainstream press today.

Flexibility and frugality

When you need to get a legacy system talking to the Internet, a proxy is probably the best way to go about it. In addition to their simplicity, proxies offer flexibility and frugality that's tough to match using any other approach.

HTTP proxies are not difficult to implement, and they don't require modification of target software. As a result, the Internet appliances your customers want tomorrow could very well be the devices you are already building today. **esp**

Bill Gatliff is an independent consultant who specializes in solving difficult embedded problems using open-source tools and techniques. He is also a member of the Embedded Systems Conference Advisory Panel and a frequent contributor to Embedded Systems Programming. Comments and questions are always welcome at bgat@open-widgets.com.

References

1. The actual HTTP message is a bit longer than this because it also includes information on the type of browser and operating system you are using and the identity of your machine. The GET is the essential text, however.
2. In some cases, you'll need to use raw IP addresses, for example, `http://127.0.0.1/` or `http://127.0.0.1:8000/`.

Resources

Gundavaram, Shishir. *CGI Programming on the World Wide Web*. This book is only available on-line now, at www.oreilly.com/openbook/cgi.

Guelich, Scott, Shishir Gundavaram, and Gunther, Birznieks. *CGI Programming with Perl, 2nd ed.* This book will be published by O'Reilly Associates in July.

Stevens, W. Richard. *Unix Network Programming*. Upper Saddle River, NJ: Prentice-Hall, 1997.

www.webmonkey.com/backend/protocols/ Just about everything on www.w3.org, if you want the gory details.

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.

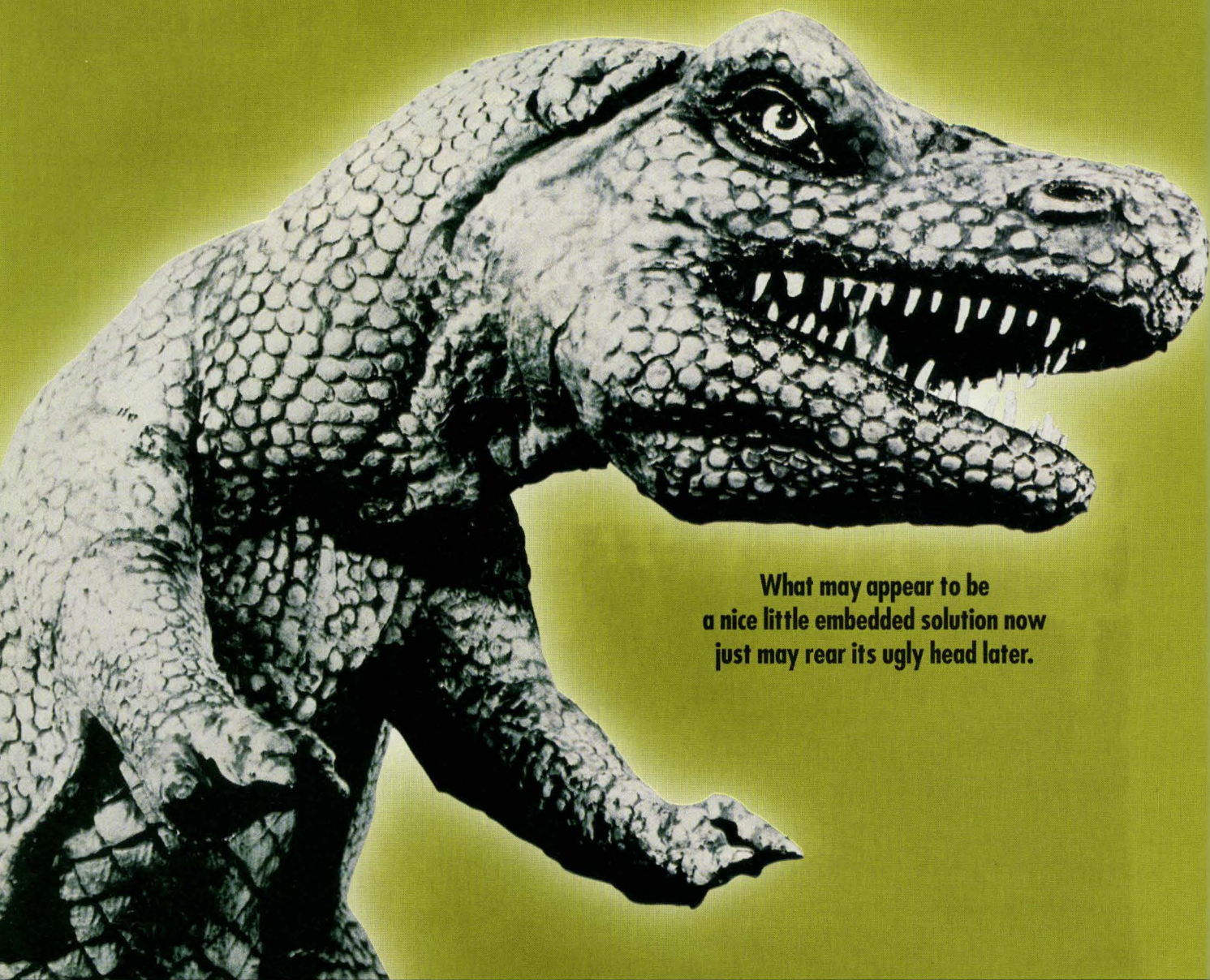
Call us today to see how Treck Inc. can help you!

Treck Inc.

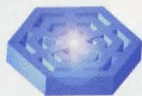
(800)340-6648 or (513)688-0553

or visit our web page at www.treck.com





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

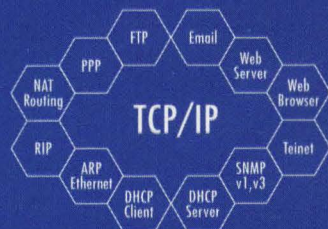


PROVEN SOLUTIONS: Market needs and competitive situations will continue to change. That's why we have provided solutions over the last ten years that are modular so you can quickly adapt and easily meet these challenges with a minimum of engineering overhead.

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

Exposing MIB Data to a Web-based Interface, Part 2

This month we conclude this discussion of how web-based management can benefit from the right architecture and an SNMP MIB inheritance library.

Last month, we introduced the idea of coupling an SNMP MIB inheritance library with a backplane-based architecture to leverage the advantages of each. This month, we'll present examples and look at some even more powerful advantages of adding WBM support this way.

Operation with scalar MIB objects

Again, assume that you have the following HTML found in `MYFILE2`, where `MYFILE2` is stored on the embedded network device using such an architecture:

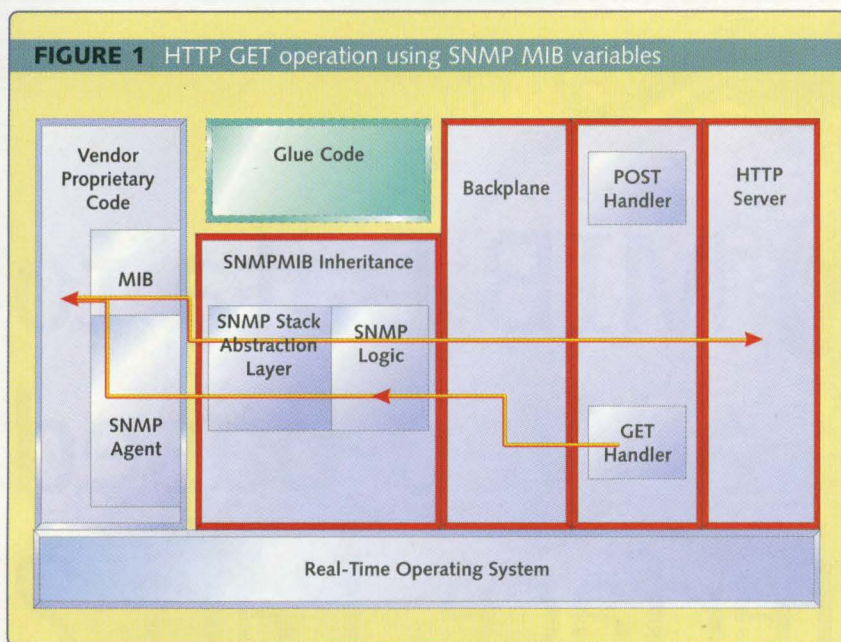
```
<INPUT TYPE="text" SIZE="20" NAME="sysName" VALUE="%sysName%">
```

As previously described, this HTML is a `FORM` element that describes a one-line text box, which allows an end user to both read and write data.

Now as before, when an HTTP request is made for `MYFILE2`, it gets loaded into the Web-based management solution's GET Handler. The GET Handler parses through the HTML until it runs into `%`. From there, it knows that `sysName` is a special tag.

At that point, the GET Handler again consults the backplane in an effort to get the data associated with `sysName`. However, unlike the pre-

FIGURE 1 HTTP GET operation using SNMP MIB variables



suitable for viewing via HTML, SMTP e-mail alerts, command-line interfaces, Java applets, and so on—and passes it into the HTML being streamed out of the device. This whole process is shown in Figure 1.

When the backplane can't find an entry for `sysName`, it passes it off to the SNMP MIB inheritance library. The library, in turn, formats the request for `sysName` (depending on the needs of the SNMP stack), and makes a call to the GET function either via the SNMP agent, or by the MIB directly. Once retrieved, the relevant data is sent back to the SNMP MIB Inheritance library, converted, and copied into the HTML that is being streamed out to the browser.

An HTTP POST is processed in a similar fashion to an HTTP GET. A POST operation is shown in Figure 2. The POST handler parses through the CGI stream, which might look something like:

. . .&sysName=James+Blaisdell&. . .

After that, the POST Handler consults the backplane for `sysName`. When the backplane is unable to locate an entry for `sysName`, it then passes the data and the HTML tag off to the SNMP MIB inheritance library, which formats the data. Lastly, it accesses the MIB variable's SNMP SET routine, and the data is updated to "James+Blaisdell."

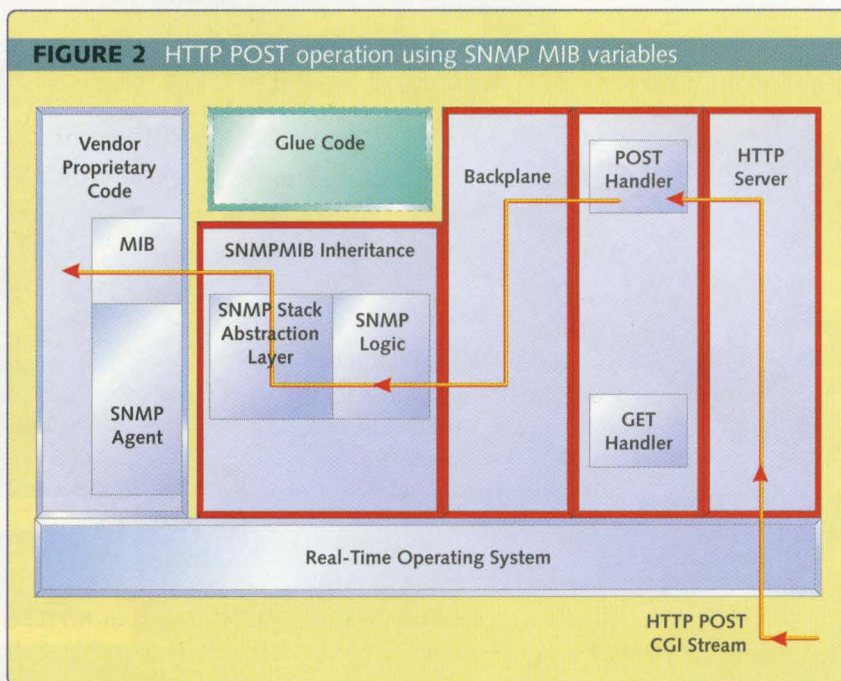
Operation with tabular MIB objects

Despite the elegance of the design so far shown, there is far more to the SNMP MIB inheritance technology than just the proper formatting and conversion of MIB variables. The library also needs to provide extensive support for SNMP tables. Since most SNMP MIB objects are in table format, this support is undoubtedly the most sophisticated part of such a library.

To show how the SNMP MIB inheritance library works with tables of data, an in-depth example is provided here.

As shown in Figure 3, the Structure

FIGURE 2 HTTP POST operation using SNMP MIB variables



vious example, where the backplane had an entry for `userName`, the backplane will simply not have an entry for `sysName`. And when the backplane can't find an entry for `sysName`, it assumes that it is an SNMP MIB variable and passes it off to the SNMP MIB inheritance library.

At that point, the library takes over. Depending on the SNMP stack, the module will either access the

Management Information Base directly, or it will build up a PDU and pass it off to the agent, whichever is appropriate.

Regardless, the SNMP GET routine associated with the MIB variable `sysName` is called, and the data represented by `sysName` is returned in numerical notation. The SNMP MIB Inheritance module then takes that data and converts it back into text—

When the largest companies in the world are faced with a challenge, who do they turn to?

- Challenge 1:** ORACLE wanted to use an Internet-enabled vending machine in their CEO's keynote at OpenWorld.
- Challenge 2:** SUNBEAM wanted to show networked consumer appliances at the National Housewares Show.
- Challenge 3:** GE wanted to demonstrate Internet-enabled smart concept appliances at the International Builders' Show.
- Challenge 4:** INVENSYS wanted a local area device-networking solution for their home and light industrial ControlServer™ and SmartModule™ products.
- Challenge 5:** GALILEO wanted a networking management solution for low-cost Ethernet switches.
- Challenge 6:** AT&T wanted to demonstrate wide area device-networking for their global network.

One Solution: e-smart device networking from emWare.

When the largest companies in the world need device-networking solutions, they turn to emWare. In this age of "e" everything, we can help you be e-smart. Look us up on the Web at www.emware.com or call us toll-free today at 1.877.436.9273 to find out how emWare can solve your device-networking challenges.



© Copyright 2000 emWare, Inc. The ETI logo, e-smart and the e-smart logo are trademarks of emWare, Inc. All other trademarks are the property of their respective owners.



ESPS02

of Management Information (SMI) consists of a lot of groups and sub-groups. However, as you wend your way through the SMI hierarchy, eventually you wind up at a leaf node. Leaf nodes are the members of the SMI that have actual GET and SET routines associated with them. Leaf nodes

are where the “rubber meets the road” in the world of SNMP.

The variable we have discussed so far, `sysName`, is a leaf node, but it represents a scalar object. That is, `sysName` is an object that has one and only one instance—hence the “instance” variable that was appended

to the numerical specification of `sysName`, 1.3.6.1.2.1.1.5, was 0 for a final, unique object identifier (OID) of 1.3.6.1.2.1.1.5.0.

However, leaf nodes can represent table objects as well—objects that have more than one instance. In this case, the instance variable can be more than a simple 0. In fact, the instance variable can be far more complex.

For example, suppose we want to get a particular value of `tcpConnLocalAddress`. Now `tcpConnLocalAddress` is an SNMP MIB object that fits within the SMI as a member of `tcpConnEntry`, which is part of `tcpConnTable`, which is part of the `tcp` group of MIB II. (And since it is part of MIB II, both consoles and agents will have a priori knowledge of `tcpConnLocalAddress`.)

The arrangement of `tcpConnLocalAddress`, within the SMI, is shown in Figure 4. Thus to access `tcpConnLocalAddress`, we need to specify (at a minimum):

```
iso.org.dod.internet.mgmt.mib-2.tcp.tcpConnTable.tcpConnEntry.tcpConnLocalAddress
```

or, in numerical form:

```
1.3.6.1.2.1.6.13.1.2
```

This is the base OID of `tcpConnLocalAddress`. However, as shown in Figure 3, `tcpConnLocalAddress` is actually part of a table, `tcpConnTable`, that is made up of `tcpConnEntries`. Furthermore, as shown in Figure 4, there are four separate instances of `tcpConnLocalAddress`—the instances which have values of 170.1.81.4, 170.1.81.5, 0.0.0.0, and 144.5.66.13, respectively.

But how do you specify, via SNMP, which instance you are interested in getting or setting? As shown in Figure 4, four variables in this table are indexable—`tcpConnLocalAddress`, `tcpConnLocalPort`, `tcpConnRemAddress`, and `tcpConnRemPort`. According to the

Flash ISP/IAP

Now for 16-bit MCUs and DSPs

The diagram illustrates the internal architecture of a PSD (Programmable System Device) chip. It features a central 'Main Flash' and '2nd Flash or EEPROM' connected to a 'Decode' block. Below these are 'SRAM', 'Programmable Logic', 'Security', and 'PMU' blocks. The chip is connected to an 'MCU Interface' and 'I/O Ports' on the top and bottom. A 'FLASH PSD' label is on the right side. A laptop and a mouse are shown connected to the chip via an 'ISP Port' and a 'Mouse' respectively.

PSDsoft Express

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

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

Waferscale-USA
Tel. 800-832-6974
Fax 510-657-5916
info@waferscale.com

Waferscale-EUROPE
Tel. 33-1-69320120
Fax 33-1-69320219
wsifrance@wsiusa.com

Waferscale-ASIA
Tel. 82-2-761-1281
Fax 82-2-761-1283
james@mail.wsiasia.co.kr



Kiss your ASIC good-bye.

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

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

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

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

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

The unbeatable low-risk, high-volume solution

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



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

Priced to go, go, go

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

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

Why would you choose anything else?

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

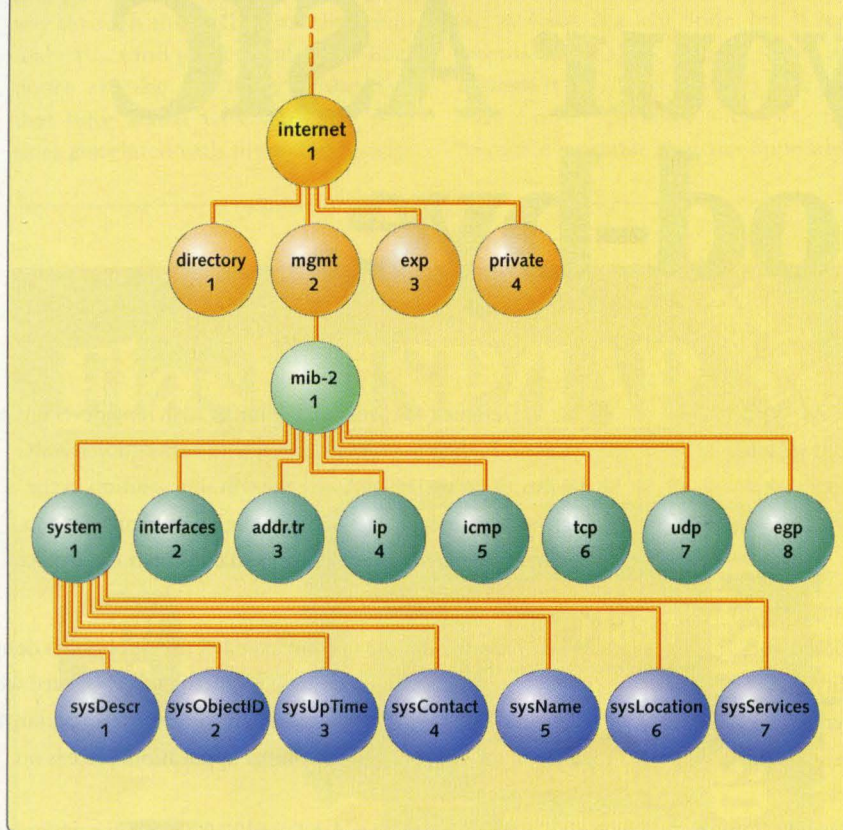
Find out more about Spartan-II FPGAs today. Visit our website, and we'll prove to you that it's time to kiss *your* ASIC good-bye.

www.xilinx.com



The Programmable Logic Company™

FIGURE 3 Structure of management information through the system group



SNMP protocol, all indexable variables in a table are used to index the table entries. Thus, to access a particular table entry for the `tcpConnTable`, you would have:

```
a.b.(tcpConnLocalAddress).(tcpConnLocalPort).(tcpConnRemAddress).(tcpConnRemPort)
```

where:

a = value for `tcpConnEntry` (1.3.6.1.2.1.6.13.1)
 b = value for specific column in `tcpConnEntry` (for example, 2 for `tcpConnLocalAddress`)
 (name) = value in the object name. For example (`tcpConnLocalAddress`) means 170.1.81.4, if we are talking about the first row of `tcpConnEntries`

Thus, in order to fully specify the OID of the first instance of

`tcpConnLocalAddress`—the instance in the first row of the table—we would need to append 170.1.81.4.80.170.1.81.28.2049 to the base OID of `tcpConnLocalAddress`, 1.3.6.1.2.1.6.13.1.2, for a total specification of:

```
1.3.6.1.2.1.6.13.1.2.170.1.81.4.80.170.1.81.28.2049
  \object identifier      \instance
```

As shown in this example, instances can be incredibly complex entities.

When an NE receives a request for the OID above, it will first look up, in its Management Information Base, the entry for the OID 1.3.6.1.2.1.6.13.1.2, as shown in Figure 5.

Then, after it gets the relevant GET or SET routine, the SNMP agent will pass the instance variable, 170.1.81.4.80.170.1.81.28.2049, into the routine, and await the appropriate response. (It is up to the GET or SET routine to deal

with the massive, and sometimes hideous, instance variables.)

This is essentially the way that tabular data is used in SNMP. As highlighted in this example, tables can be quite complicated.

So how, then, does the MIB inheritance library deal with such complexity? Can it work with really complicated instance variables like the one described above? And can it work in a really general way? Lastly, is the complete separation of HTML and C still maintained? The answer to all three questions is a resounding “yes.”

The SNMP MIB inheritance library is able to deal with such complexity through the use of special tags. These special tags are essentially “meta-tags,” in that they do not necessarily display or alter datum individually.

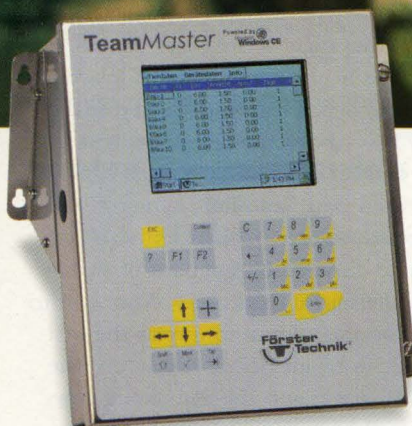
Instead, the glue code that is associated with each meta-tag serves to “prime” the SNMP table via GET NEXT and GET ROW calls to the specific SNMP stack. Also the glue code for the meta-tags, in case you are wondering, constitutes the “Table Logic” and the “Advanced Features.” The logic is provided as part of the library.

For example, to display our table of `tcpConnEntries` (and thereby display `tcpConnLocalAddress`), we would have the following HTML shown in Listing 1.

As you can see from this HTML example, four of the meta-tags are worth explaining—`createTable`, `REPEAT`, `endRow`, and `endTable`. These meta-tags all have glue code logic that serves to work through the SNMP table. (The other meta-tags—`tcpConnState`, `tcpConnLocalAddress`, and so on—are meta-tags that do not have any glue code associated with them. They are meta-tags that would be referenced in an identical fashion to the `sysName` example given earlier, in that they would be passed directly into the SNMP MIB inheritance library as shown in Figure 1.)

The “API” for each of these four meta-tags is described in the following paragraphs. As you can see from these API specifications, meta-tags can take

Windows CE is feeding cows.



m&f
Engineering Team

When you've got hundreds of hungry bovines to feed, you don't have time to think about whether your operating system can run entirely in CompactFlash. Or if it supports the Microsoft® Win32® API. But luckily for busy dairy farmers, Förster Technik does. And that's why, with the assistance of Mettler and Fuchs AG, they powered their TeamMaster industrial automatic nursing station with Microsoft Windows® CE.

Windows CE brings the ease-of-use of a PC to rugged embedded devices—in any industry. With it, dairy farmers can experience the familiar Windows environment from their animal's stalls, feed calves in the field and monitor the herd's health in the office—all while streamlining their business.

Windows CE and the powerful Win32 API facilitate code reuse and feature expansion, so Förster Technik can add modules to the TeamMaster to dispense feed and weigh cows. And the robust Windows CE operating system can run entirely from RAM and ROM or CompactFlash, making it the perfect RTOS for ruggedized devices.

Dairy farmers don't design embedded systems. But you do. So why not use the environment that's familiar to both users and developers?

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**

arguments. This is a general property of meta-tags and is *not* limited to these tags in particular. That is, you can use arguments in your own (non-SNMP) tags as well.

```
$%createTable(  tableName,
                lengthInstance,
```

This meta-tag, based on its arguments, creates a "Table Descriptor," which is

```
firstInstance,
skipAhead,
filterType,
filter,
ListOfColumns)#$
```

an internal data structure that is used to control the display and alteration of SNMP tabular data. The arguments to this meta-tag are explained below as follows:

tableName = the ID of the table. This ID is not just a standard OID. Instead, it needs to be a table entry (a column) that is fully populated—it cannot have any holes. For example, if you are building a table of **ifEntries**, you should probably have **ifIndex** be the name of the table. It is the only row entry that is guaranteed to have a value at all times (the other entries might change if you have a system of "hot-swappable" cards, for example).


lengthInstance = the length of the instance identifier. This can be either left blank (in which case it is assumed to be 1), it can be an integer, or it can be a wildcard '*', so that its value can be calculated at run time.

firstInstance = specifies the first instance in the table to be displayed. This can be left blank, in which case the very first row in the table is sought by the **createTable** glue code, or it can be specified with another value of the instance variable. This is useful if you want to display different parts of a table on different HTML pages. For example, you might want to display the first 100 rows of an SNMP Table on one HTML page, and the remaining rows on another. In the **createTable** meta-tag on the second HTML page, then, you would specify the first instance to be the 101st row of the table.

skipAhead = determines the amount the table should be advanced from the starting row in **firstInstance**.

filterType = this is a string to determine the type of instance filtering to be applied to this table. It should be either "include" or "exclude."

filter = specifies how you want to filter the table display based on



It's Proven — RABBIT 2000™


Microprocessor Outperforms Floating Point Competition

The **RABBIT 2000** delivers high-performance floating point, leaving 8-bit competition in the dust. The outstanding performance is a result of the **RABBIT 2000**'s powerful number crunching ability and optimized Dynamic C® software libraries. The Dynamic C® software development system includes an interactive editor, compiler and debugger. Integration of hardware and software, and numerous on-chip peripherals, simplifies your design effort.

Floating Point Operation	RABBIT 2000 @29.49MHz	Zilog Z180 @24.58MHz	Dallas DS80C320 @33.18MHz	Phillips @33.18MHz	AMD 188ES @36.86MHz
Add	9.6µs	26	32	78	194
Multiply	12µs	42	34	85	184
Square Root	32µs	343	334	805	355
Sine	94µs	1238	452	1112	804


Full information on benchmark tests available at www.rabbitsemiconductor.com/benchmark.html. Clock speeds reflect maximum permitted clock speed for 55 nS memory and standard baud rates.

- **Glueless interfacing**
- **1 megabyte of code space**
- **4 serial ports**
- **Remote cold boot**
- **40-plus I/O pins**
- **Slave port**
- **"Sleepy" mode allows 32 kHz operation at 100-200 µA**
- **7 timers, battery-backable time/date clock, watchdog**



**Rabbit 2000™
Development Kit
Only \$139**

Includes:
Single-board computer with Rabbit 2000™ processor, prototyping board, Dynamic C® development software on CD ROM, power supply, and PC serial cable to perform real-time debugging.



Visit our web site and order your Rabbit 2000™ Development Kit!
www.rabbitsemiconductor.com
 2932 Spafford Street, Davis, CA 95616 • Tel 530.757.8400 • Fax 530.757.8402
Dynamic C® is a trademark of Z-World, Inc.

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/esp.

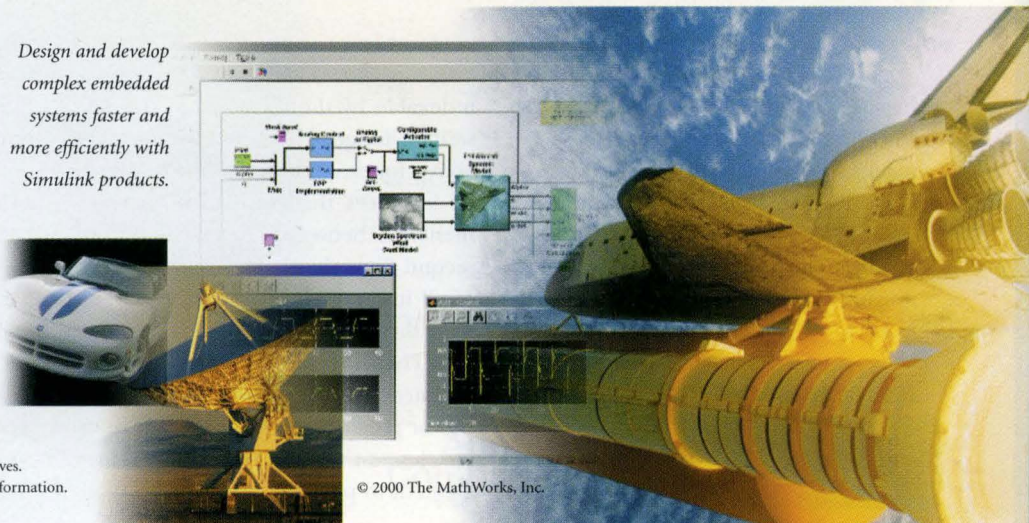
SIMULINK®

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



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.

FIGURE 4 Structure of management information through tcpConnEntry

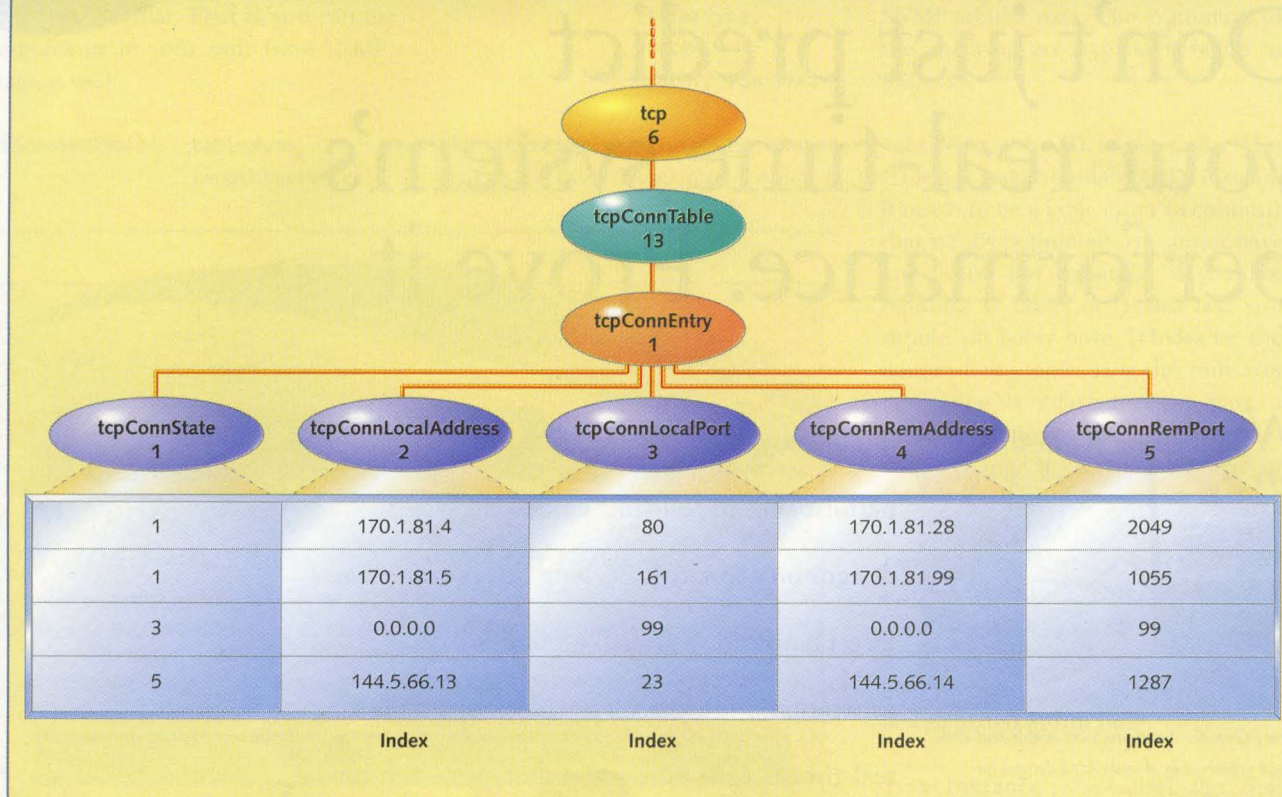
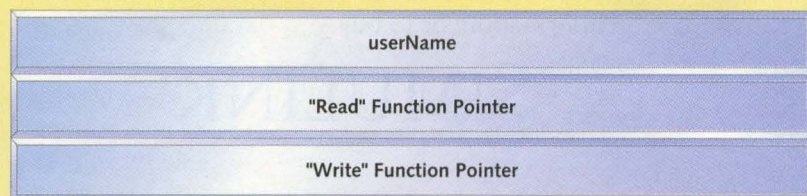


FIGURE 5 MIB object for tcpConnLocalAddress



requires the instance fifth element to be 80, and so on.

Thus, the row given by the instance 171.2.4.56.80.170.1.34.27.19 would be retrieved and displayed via the relevant `tcpConnState` or `tcpConnLocalAddress` special tags as appropriate.

However, the row given by 171.2.1.99.81.170.1.34.27.19 would not be retrieved, and hence its objects would not be displayed, because this instance failed on elements three and five (1 is less than 2*, and 81 doesn't equal 80).

`listOfColumns` = a listing of the columns in the table—in this example, it is `tcpConnState`, `tcpConnLocalAddress`, `tcpConnLocalPort`, `tcpConnRemAddress`, and `tcpConnRemPort`.

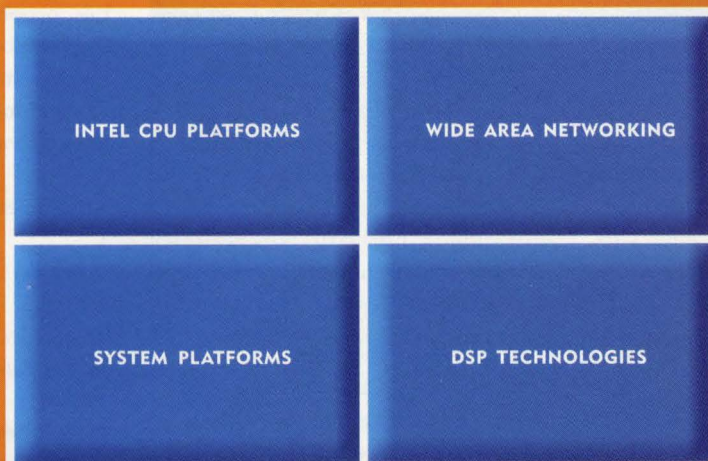
`$(REPEAT(tableName, m, n)#$`
This meta-tag is used to cycle through the rows of a table. It takes the

instances. As described in the example for `tcpConnLocalAddress`, these instances can be extraordinarily complex. Hence, filters are a wonderful way of managing this complexity. For example, in our `tcpConnLocalAddress` illustration, you could specify a filter to be:

170-200.*-5.2*.*.80.170.1.*.25*.*

This would allow, if the filter type is "include," the display of all instances

indexable by the elements in the prescribed range. The first filter element above, for example, allows the display of all instances that have their first element fall between 170 and 200. The second filter element allows the display of all instances with a second element from the lowest allowable value through five. The third filter element allows all instances with a third element equal to two or greater to "pass." The fourth filter element allows any instance fourth element to pass, the fifth filter element



**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

LISTING 1 Code to display table of tcpConnEntries

```
<! %createTable(tcpConnLocalAddress,*,  
include, 170-200.*-5.2-*.80.170.1.*.25-*.*,  
tcpConnState,  
tcpConnLocalAddress,  
tcpConnLocalPort,  
tcpConnRemAddress,  
tcpConnRemPort)#$>  
<! %REPEAT(tcpConnLocalAddress,1,numCurrTcpConnEntries)#$>  
  <tr>  
    <td><font face="Arial, Helvetica">${tcpConnState}#</font></td>  
    <td><font face="Arial, Helvetica">${tcpConnLocalAddress}#</font></td>  
    <td><font face="Arial, Helvetica">${tcpConnLocalPort}#</font></td>  
    <td><font face="Arial, Helvetica">${tcpConnRemAddress}#</font></td>  
    <td><font face="Arial, Helvetica">${tcpConnRemPort}#</font></td>  
  </tr>  
<! %endRow(tcpConnLocalAddress)#$>  
<! %REPEAT(END)#$ >  
<! %endTable(tcpConnLocalAddress)#$ >
```

tableName, which is used to identify the proper "Table Descriptor," and retrieves the row specified by the firstInstance argument in the createTable tag. Then, it essentially does a loop of the form:

```
for (index = m; index <= n; index++)  
{  
  code  
}
```

The arguments are:

tableName = the table ID. This is the same tableID as given in createTable
m = the starting index
n = the ending index

Both m and n can be tags themselves (either normal glue-code based tags or MIB variables). That way, the table can have a dynamic number of entries. The tag numCurrTcpConnEntries, given in the HTML example above, is a normal HTML tag that has "Read" and "Write" glue code routines, which are accessed in the normal way.

The %REPEAT()#\$ meta-tag is delimited by the %REPEAT(END)#\$ meta-tag. That is, all the HTML, and all of the tags, that lie between the first call to %REPEAT()#\$ and the call to %REPEAT(END)#\$ are repeated (displayed) n-m times.

%endRow(tableName)#\$

This tag tells the Table Logic to advance to the next row, based on the filter characteristics. Namely, every time %endRow(tableName)#\$ is encountered, the next conceptual row in the SNMP Table is retrieved, based on the filter logic. From there, the individual tags can be used to retrieve the data. Note that as per the operation outlined for the REPEAT meta-tag, %endRow(tableName)#\$ must be located between the %REPEAT(...)#\$ and the %REPEAT(END)#\$ tags.

tableName = the table ID. This is the same table ID as in createTable and REPEAT.

What do the leading silicon vendors know about BIOS?

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

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

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

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

Embedded BIOS™

ADAPTATION KIT:

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

CORE BIOS FEATURES:

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

CHIPSETS:

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

IDEAL FOR:

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



GENERAL
SOFTWARE

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

© 2000 General Software, Inc. All rights reserved. General Software, the GS Logo, and Embedded BIOS are trademarks of General Software Inc.


```
}%endTable( tableName )#%
```

This tag clears the "Table Descriptor" for `tableName` and frees up all the relevant memory, mutex semaphores, etc. It must come after the `%REPEAT(END)#%` tag.

`tableName` = again the standard table ID mechanism.

Last Notes

This is just about the end of our my description of the SNMP MIB inheritance library technology. However, two last features should be mentioned.

First, the SNMP MIB inheritance library should support the concept of overrides. Namely, the SNMP data type conversion methods built-in to the library can be overwritten within the HTML. For example, you could have the HTML tag:

```
}%MACADDR:ifEntryMacAddr.0#%
```

While the default method would print a string in dotted decimal notation, the overload method (`MACADDR`) would display hexadecimal values delimited by ':' characters, such as `03:15:a2:ff:24:73`.

Second, the library should support inline instances. For example, you could have the tag:

```
}%TrpRcvrEventLast.{[TrpRcvrIpAddr  
]}#%
```

where each element is a dynamic variable in its own right. These elements, in turn, can be nested with no real limit to the depth of the nesting. The code first looks up `TrpRcvrIpAddr` in the normal way (see Figure 1), and then uses that to instance `TrpRcvrEventLast`. Also, the inline capabilities can be used with any of the above features.

UI flexibility

I've outlined a pair of technologies: a backplane-based architecture for extensible and scalable Web-based management and an SNMP MIB

inheritance library that together give a network equipment vendor unprecedented power and leverage in creating a WBM interface.

First, by completely separating the HTML from the C code, the a backplane-based architecture allows UI development to be completely separated from the back-end engineering. This opens up tremendous opportunities for rapid UI prototyping, for localizing your product to foreign markets, and for OEM deals. In short, by separating the HTML from the C, the backplane makes WBM not just another "checkbox item," but a strategic business imperative.

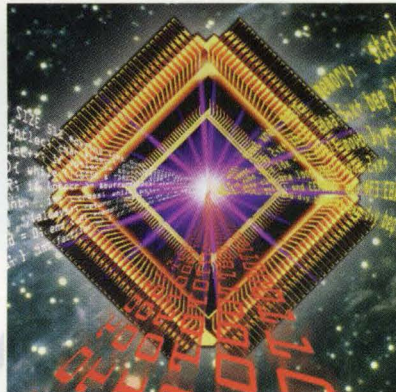
Second, through a properly designed SNMP MIB inheritance library, it's extraordinarily easy to leverage all of the engineering effort that went into specifying the MIB variables in a network element (NE). In

fact, it requires little or no additional engineering effort.

Thus, using these two technologies in tandem gives a company the capability to have a rich, full-featured WebUI that is easily customizable, scalable for OEM and foreign market opportunities, and yet requires almost no engineering effort to achieve. **esp**

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

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.

Can your Java™ make this ball float?



Small AND
fast Jbed can.



And every swing is a home-run!

Jbed

Pure Java™
based RTOS
for embedded
systems

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.jbed.com, info@esmertec.com, F +41 1 445 37 30, X +41 1 445 37 34

esmertec

your partner for embedded real-time java

Real-time Java Wars Yield Uneasy Truce

Real-time Java is not an oxymoron, according to the two groups presently developing competing sets of specifications.

Sometimes the most effective way to wage a war is to declare victory and go home. That's what's seemed to have happened in the battle between Sun Microsystems, and its rivals over a series of real-time extensions to the Java specification.

To understand where real-time Java stands today, this tangled, fractious story is best told backwards. We'll start with the latest developments—which could result in some useful implementations by year's end. Then we'll work our way back through to the history of the full-blown fights over real-time Java that flared in late 1998.

The initial fracas flared up between Sun and Hewlett-Packard. The sticking point was and continues to be control of the spec. While Sun is willing to accept input from other companies—indeed, it is developing its spec under the rubric of its "Java Community Process"—Sun nevertheless wants to maintain firm control of the final document. That is, they have decided not to cede the spec over to one of the industry's neutral specification-maintenance organizations.

That position has stuck in the craw of the many other players, both large and small, that have been involved. For one, IBM has been playing a prominent role in attempting to get past the conflict and move real-time Java forward. Indeed, many believe IBM may yet emerge as the preeminent purveyor of Java in the embedded and real-time arenas.

"I call IBM the stealth company," said Jerry Krasner, director and research editor of Electronics Market Forecasters. "They have been superb at pushing forward the standard and at launching tools for deeply embedded development."

More good news is that this year's JavaOne conference, to be held in June in San Francisco, should yield some hard information of interest to developers. Further information—perhaps the announcement of some of the first real-time

TABLE 1 Java traits**All developers of real-time Java have agreed with the following principles by NIST:**

Java's higher level of abstraction allows for increased programmer productivity.
 Java is easier to master than C++.
 Java is secure by keeping software components (including the VM itself) protected from one another.
 Java supports dynamic loading of new classes.
 Java is highly dynamic, supporting lots of object/thread creation at run time.
 Java support component integration and reuse.
 The Java language and platforms support application portability.
 The Java technologies support distributed applications.
 Java provides well-defined execution semantics.

Source: National Institute of Standards and Technology (NIST)

Java implementations—could be on tap this fall at the Embedded Systems Conference.

Lay of the land

A little background is required to understand the lay of the real-time Java land. It's confusing, hard to navigate, and involves a cast of players which are often hedging their bets by sitting on both sides of the real-time Java fence. Even those involved in the work find the shifting sands of alliances and development efforts confusing. Indeed, it is the objective of this article to recount as clearly as possible what has happened and where things are going, even though no single engineer in the industry is privy to every facet of the multi-sided Java jewel.

To recap, the conflict over a real-time Java specification began simmering at ESC in November 1998 between Sun and Hewlett-Packard. That's about the time Sun corralled a group of its Java licensees and began to develop a spec of real-time extensions. The Sun-led group was called the "Java Experts Group."

Unhappy with what they felt was a heavy hand by Sun in its control of that spec effort, a competing collection of Java vendors led by Hewlett-Packard and NewMonics Inc., formed its own group, known as the "J Consortium."

Both parties wanted to control extensions that have to be added to

the Java Virtual Machine to enable it to support real-time operation. The extensions help ensure that a given Java implementation can deliver guaranteed responses to interrupts—a tenet of real-time operation.

Interestingly, both groups worked to create their real-time spec using the same baseline document: "Requirements for Real-time Extensions for the Java platform," published by the Information Technology Laboratory of the National Institute of Standards and Technology (NIST).

Adding to the inbred aspect of the two opposing efforts is the fact that representatives from both the Experts Group and the J Consortium had a hand in creating the NIST requirements document.

Winning the war?

Today, according to most knowledgeable observers, Sun and the Experts Group have won the spec battle.

Currently, version 0.9.2 of the Experts Group's spec is available on Sun's Web site (see Resources). It's expected that version 1.0 will be published on or around June 1 by Addison-Wesley as part of its well-known series of Java books. The 1.0 spec itself is currently set to be formally unveiled at the Embedded Systems Conference this fall.

"This specification, which is close to beta release, will benefit companies who are anticipating an industry standard application programming inter-

face (API) that extends the Java platform with hard real-time capabilities," said Vicki Shipkowitz, an embedded product manager at Sun.

More important, several companies are believed to be hard at work on reference implementations of real-time Java. These include QNX, Agile, and another company I'm not allowed to mention, having been sworn to secrecy. The only hint I can drop is that it is definitely *not* a couple of engineers working in a Silicon Valley garage. (Quite the contrary, in fact.) Also expected to field a real-time Java is a company called TimeSys.

For its part, IBM is thought to be writing the conformance test suite through which future, third party, real-time Java implementations can be validated.

Diverging approaches

In terms of technology, the two real-time Java specs provide a fascinating view into the different ways engineers believe that they can take advantage of Java's higher level of abstraction—despite its trade-off of poorer runtime efficiency.

Sun's (that is, the Experts Group's) spec is somewhat looser, in that it gives developers more freedom as to how they can implement its required features. In contrast, the J Consortium document, which is also available online (see Resources), is much more specific as to mandated practices, including class structures and the handling of different types of events.

"The garbage-collection behavior is very much unspecified by Sun," said a Java-savvy engineer. This could lead to charges that the Sun spec does not strictly mandate determinism; that is, all implementations hewing to the document would be embedded, but some would be more "real-time" than others. (Sun says it has laid down firm provisions for hard real-time performance.)

Sun's initial rationale for its real-time spec was laid out in a document which read, in part: "This extension is necessary because the guarantees and

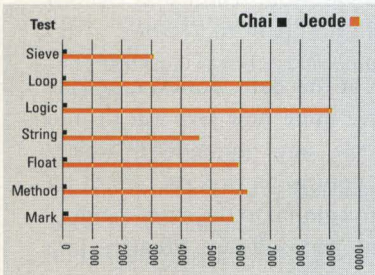


ACCELERATED JAVA TECHNOLOGY OUTPACES
"JAVA WANNABE'S" FOR INTERNET APPLIANCES



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

Embedded CaffeineMark 3.0



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



Accelerated Java Solutions
for Internet Appliances and
Embedded Devices.

APIs provided by the standard Java platform do not meet the needs of real-time systems. We propose to develop a real-time Java standard extension specification and reference implementation. The underlying technology needed to implement this spec is essentially a Java virtual machine (VM) that is built to honor deterministic guarantees of its run-time behavior. A central component of this VM is a real-time garbage collector. This VM would only be implementable atop suitable target platforms, such as a real-time operating system." The spec was to have profiles aimed at both "hard" and "soft" real-time systems.

As for the real-time operating system (RTOS) community, most RTOS vendors have steered toward Sun's side of the fence. That's because the competing J Consortium run-time doesn't necessarily even need an RTOS. It can run on the bare metal. That's not true of Sun's approach, which requires that the real-time APIs be interfaced through an RTOS.

Game's not over

Meanwhile, the J Consortium made a decision to focus on a smaller group of real-time requirements than were laid out in the broad-based "Requirements for Real-time Extensions for the Java platform" document published by NIST.

"We decided to pare back the ambitious list in the NIST requirements document and focus on core extensions, which address the traditional embedded developer who's used to C or assembly language," said Kelvin Nilsen, chief technology officer at NewMonics.

"It's unclear which group Sun is trying to address," Nilsen charged. "They're not addressing issues of memory footprint and aren't fully addressing the issues of performance and latency."

Regardless, it's not currently clear how many J Consortium-based implementations will make their way to market. "The J Consortium is continuing

to make tweaks to their spec, but there's no plan that I'm aware of for anyone to implement it," one industry pundit claimed.

Not so, according to NewMonics's Nilsen. "NewMonics and some other companies like Aonix are looking at ways to bring the J Consortium spec to market, but no companies other than Omron have announced anything officially," he said.

Indeed, the game may be far from over. One fascinating rub is that nearly every Japanese Java vendor is supporting the work of both groups. In the United States, companies have been more apt to take sides. While Sun constitutes the backbone of the Experts Group, the J Consortium is populated by Hewlett-Packard, Microsoft, NewMonics, Siemens, Omron, and Aonix (the latter three are members of both groups.)

For example, the APIs specified by the J Consortium have been added to Japanese vendor Omron's proprietary implementation of real-time Java. For that reason, the Omron code effectively constitutes a superset of the J Consortium spec.

Siemens, which is one of the fence sitters, prepared an interesting technical presentation to the J Consortium (it's not known if the documents were shown to the Experts Group, too). The slides add some quantitative meat to a proposed series of real-time profiles, which would address a range of real-time requirements from soft to hard. Specifically, the Siemens engineers proposed an initial profile which requires "the fewest extensions and which reflects existing real-time practice."

This profile is based on the use of call-back routines with separate execution contexts (that is, events are processed distinctly). "Such a profile can provide for extremely short worst-case response" in the tens of microseconds, the Siemens engineers noted in their slides.

They also suggested limiting priority levels to no more than ten. Finally,

garbage collection should fit within the regular thread-priority scheme and should be pre-emptable only by threads that don't themselves require memory clean-up.

Where's Waldo?

Despite such assiduous specmanship by both committees, a strange fate seems to have befallen real-time Java. While the technology was clearly the star of the fall 1998 Embedded Systems Conference, lately, an eerie silence has settled upon the community.

Indeed, earlier this year, perhaps the biggest shock at the Spring 2000 edition of the Embedded Systems Conference in Chicago was the relative quiet surrounding real-time Java. In contrast, last year the conference was abuzz with advance word on great things to come.

One reason for this year's relative silence may be a split in viewpoints among embedded developers. That is, developers anxious to find an alternative to Microsoft's Windows CE (a real-time version of which has not yet been completed) are hoping to get their hands on some meaty real-time Java implementations. (Still to weigh in is the wild card of embedded Linux; it's currently an open question as to whether that OS is ready for prime time in real-time applications. More about embedded Linux next month.)

Still, Java developers appear to be plugging away quietly, especially in the Internet appliance realm. The trend has already started, as a host of embedded systems developers rush to ready an emerging generation of Web browsers, pagers, smart phones, and a bunch of other small-form-factor products—the latest in downsized, low-power, consumer-oriented gadgets.

Even at this date, however, no one is positive quite how Java will fit into this brave new embedded world. Indeed, contrarian experts are not yet willing to commit to the extra memory footprint and software libraries Java tends to require.

That's why the potential entrance of an embedded or real-time Linux opens up a whole new can of worms. The full-blown implementation of Linux is available essentially for free (that is, it's open source); however, that version is more appropriate for desktop and enterprise markets. The dynamics of Linux in embedded apps are not yet clear, although product announcements are imminent.

Should that come to pass, it could force Sun and its allies to place real-time Java, too, into the open source realm, if only to maintain the OS's position as a potent competitor. Indeed, reports to that effect have been circulating for months, though no officials will provide definitive comments.

Stop-gap measures

For the time being, Sun appears to have an interim strategy up its sleeve. The company is poised to launch something called the "Java Community Process 2.0 (JCP 2.0)." That will supersede the current JCP under which the Experts Group's spec is being developed.

However, Sun won't yet say exactly what JCP 2.0 is or how it's different from today's JCP. A company spokesperson did provide a nebulous explanation to the effect that "there is a great dialogue going on between all the members of the Java community. They are exchanging ideas very proactively. I think you can expect something in the near future, but it's not imminent."

What exactly that means, no one in the Java community is quite sure. Pundits believe JCP 2.0 will be more open than the current process—though how much more open is anyone's guess—but that Sun will still have the final say when it comes to the Java spec.

Vigorous adolescence

Moving forward, most Java supporters believe the language is entering an adolescence from which it will emerge with renewed vigor.

"The embedded stuff is clearly a longer haul than the Enterprise was," James Gosling, one of the original developers of Java at Sun, said. While things are now looking much brighter, that still seems to be the case.

In addition, everyone agrees that a rapprochement is in order if Java isn't to suffer the same fate that fragmented Unix into dozens of flavors.

"For a real-time Java spec to be effective, it's going to have to include the major players coming together in a totally open standard," said Krasner of EMF. HP, Sun, and IBM all say they want to agree on a single spec for real-time Java.

That's why the engineer with his or her real-time design on the chopping block might be wise to consider that it may well be the marketplace—rather than a that will have the final say. **esp**

Alexander Wolfe is managing editor for

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

Resources

Many of the Java specs and related documents are available on the Web:
www.rti.org (Java Experts Group)
www.j-consortium.org (J Consortium)
www.nist.gov/itl/div897/ctg/real-time/ (NIST requirements document)
java.sun.com/products/ (Every API document ever released by Sun)
java.sun.com/javaone (The annual Java-fest, slated for June 6-9)
www.newmonics.com (A different approach)



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
 FAX972-312-1159

The Future

Is Ahead

of Schedule

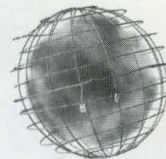
Convergence. It's happening. Now. How do you get there? Trillium. We provide the software and support to enable you to develop equipment for the convergence of telephony, Internet, wireless and cable networks. Call us at +1-310-442-9222 or visit www.trillium.com. Now. Because the future is ahead of schedule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



The World's Source for Communications Software™

Trillium Digital Systems, Inc. Los Angeles, California

www.trillium.com/ESP



Embedded Internet Tools

Embedded Linux development platform

Linux Planet is a software development platform integrating hardware and firmware with Hard Hat Linux. Linux Planet is comprised of computing engines, I/O modules, firmware, and peripheral products. The platform includes RPX Lite which features Motorola's MPC823 microprocessor, a 10Mbps Ethernet port, PCMCIA, an RS-232 interface, and an on-board ambient temperature sensor. The I/O card provides an MP3 performance-capable CODEC, video out for NTSC/s-video, multiple serial ports, IrDA, and connections for the touch screen. The CPU engine and the (H I/O X) expansion module are connected by the RPX Bus, composed of two standard 120-pin connectors. Linux Planet is available for \$5,995 based on Hard Hat Linux subscription service. RPX Lite and the Multimedia I/O are available in quantities of 1,000 for \$199 and \$99, respectively.

Embedded Planet

Highland Heights, OH

(602) 957-4500

www.embeddedplanet.com

Real-time version of Linux

TimeSys Linux/RT is a real-time version of the Linux operating system and extends the open-source Linux kernel, allowing the kernel and other processes to continue running even if a single real-time process crashes. TimeSys Linux/RT also allows users to incorporate a Real-Time Applications Interface (RTAI) layer. The TimeSys Linux/RT operating system is scheduled for release in April and will be available for free from the Internet

and for a fee on CD-ROM.

TimeSys Corp.

Pittsburgh, PA

(412) 681-6899

www.timesys.com

Networking stack for 8- and 16-bit chips

The CMX-MicroNet is a TCP/IP stack for 8- and 16-bit embedded processors. The CMX-MicroNet works with most 8- and 16-bit processors and uses between 2K and 12K of ROM, depending on the processor and the configuration. The CMX-MicroNet supports TCP, UDP, IP, SLIP, PPP, and HTTP protocols and allows for direct or dial-up connections. No proprietary protocols are required to run the stack and it can run stand-alone or with a real-time operating system. The CMX-MicroNet is available now. The base package costs \$5,500.

CMX Systems Inc.

Framingham, MA

(508) 872-7675

www.cmx.com

Software tracks real-time CORBA standards

The VisiBroker builds Common Object Request Broker Architecture-compliant (CORBA) distributed real-time applications. VisiBroker supports the Object Management Group's (OMG) Real-Time CORBA specification. New features of the Real-Time CORBA standard include the ability to map CORBA operations to real-time priority models as supported by real-time operating systems and the ability to manage a CORBA implementation's resource utilization. The Real-

Time CORBA software implements functionality from the OMG Minimum CORBA and from the new real-time CORBA standards. The Real-Time CORBA first-seat development license is \$8,000.

HighComm

Lakeland, FL

(941) 686-7767

www.highcomm.com

Java technology in embedded devices

The JSTAR co-processor is an accelerator for Java technology; its hardware accelerator can be added to any RISC or CISC microprocessor design. JSTAR interfaces with the native microprocessor core and its cache or memory subsystem. It acts as a Java interpreter in silicon, retrieving byte code instructions from memory and executing them with the native processor. JSTAR operates on Java byte codes and is designed to be integrated into any Java Virtual Machine (JVM). Adding JSTAR requires no changes to the microprocessor's instruction set or pipeline architecture. Operating systems, native applications, and software components and tools can run on a JSTAR-enabled processor. JSTAR provides a Java technology-based extension that can be added to processor designs. The first implementation of the JSTAR accelerator technology is on an R3000-class processor supporting the MIPS I instruction set architecture and Sun's PersonalJava 3.0 technology.

JEDI Technologies Inc.

Santa Clara, CA

(408) 654-8988

www.jeditech.com

ANDREY JIVSOV

A Generic Protocol Engine for Synchronous Protocols

Despite the diversity of communications protocols, there are many similarities among them. You can build one protocol engine to handle the commonality and then adjust it to the needs of each project.

Computer systems seem to evolve like groups in Darwin's evolutionary theory. As Darwin's theory shows, there cannot be only one set of "mainstream" species. Like living organisms, computer systems will always be diverse in the ways they can speak with each other. The emergence of generic protocols like TCP/IP cannot substantially reduce the number of specialized protocols.

Programmers who have experience dealing with link-layer protocols have probably noticed similarities in the protocols. Headers, trailers, and payloads can be found in the packet structure; confirmations and packet assembling and disassembling are found in the protocol scenarios. When attempting to discuss protocols or when reading their specifications, you are likely to use a limited set of entities, as well as verbs, to describe them. The presence of definable object domains and associated actions are good conditions for an object abstraction.

Brute-force approach

Protocol handling is considered to be an error-prone implementation task. Due to the asynchronous nature of protocols and uncertainty in their

Protocol handling is considered to be an error-prone implementation task. Due to the asynchronous nature of protocols and uncertainty in their responses, assuring implementation correctness is difficult.

responses, assuring implementation correctness is difficult. Special-purpose languages such as Estelle or PROMELA exist to help in protocol construction. However, implementation using these special-purpose languages has limitations. Moreover, the decision to use protocol languages for validation is worthwhile only if you have a full-featured asynchronous protocol.

Today, the C language is widely used for protocol implementation. Not surprisingly, maintaining protocol software is a difficult task. Bug fixes or additions of new protocol commands are a nightmare.

Consider the example in Tables 1 and 2. The receiver expects two packets: Response with data or not-acknowledge, NAK. If these are the only responses, the protocol implementation will first read seven bytes, because the LEN field can be zero. Based on the third byte, the protocol implementation will identify the packet it is receiving: NAK, if the third byte is 02, or Response, if the third byte is 01. If it is the Response packet, the case protocol implementation will extract the LEN field and correct the length accordingly. Understanding the protocol is difficult due to this method of implementation. In the given example are two “magic” protocol-specific constants: 7 (initial length to read) and 2 (offset to find the packet type). Suppose there is a change to the protocol of the packet shown in Table 2. Now the “magic” constants become 6 and 1, respectively.

The best way to implement receiving of the packet in C is by following the structure in Listing 1.

You need a **buffer** to fill the parts of the packets. You cannot enforce C-typing (C++, Java, and so on), because you can’t identify the packet at compile time. You will use **ifs** with the

magic constant of three for determining the **Packet**’s field.¹

Specific protocol implementation is so obscure that it represents a huge challenge to implementing a group of protocols in one source project. Copying and modifying is a common practice for protocol implementation reuse. Later, maintenance is obviously a duplication of efforts. If, in the previously mentioned example, matching for the third packet would be added to the second source tree, then visual differences for protocol implementation will be frustrating (that is, the visual

ed protocol stack is an issue. If it were possible to abstract the code for packet handling, then at some number of protocol messages, the generic approach would yield a more compact size.

Describing the engine

The data-link layer protocol abstraction can be presented with the interface in Listing 2. The **Params** type is the collection of **Param** objects. A **Param** object is the proxy for user data. It can either manage data by itself by allocating and freeing the data

TABLE 1 The sample protocol replies

Packet	Content
NAK	02 00 02 STAT1 STAT2 CRC1 CRC2
Response	02 00 01 LEN1 LEN2 DATA[LEN] CRC1 CRC2

TABLE 2 The sample protocol message

Packet	Content
Busy	02 03 STAT1 STAT2 CRC1 CRC2

difference tool will show changes in the source dealing with initially present packets). Code reuse is difficult.

Another shortcoming of the usual implementation methodology is that the packet structure is scattered throughout the source code. You can’t change a packet structure without changing the associated code. Understanding the exact protocol this implementation handles is difficult for the programmer. There is an issue of consistency between implementation and protocol documentation.

If you need to handle a series of similar protocols or a protocol with large numbers of packets in one resource-constrained device, the bloat-

LISTING 1 The C structure for parsing sample protocol replies

```
union Packet {
    char buffer[EMAX_PACKET];
    NAK nak;
    Response response;
    InProgress inProgress;
};
```

buffers, or simply be a wrapper for the buffers provided by the caller. I use a collection of data buffers for user data, thereby preserving the structure of the user’s data. In addition, the **Params** object provides synchronous access to the user’s data. The interface will:

Most of the data-link layer protocols in the industry are block protocols (byte-count-oriented protocols).

LISTING 2 The data link interface

```
class DataLinkInterface
{
public:
    Communicate(int msgId, Params &outParams, Params &inParams);
};
```

TABLE 3 ANSI 12.29 protocol's fields

Field ID	Description
LENGTH	The length of the data field.
PACKET_MULTI	This packet is a part of multiple packet transmission.
PACKET_FIRST	This is the first packet of multiple packet transmission.
PACKET_SEQUENCE_BIT	This bit is toggled for each new packet sent.
PACKET_SEQUENCE	Packet number from max-1 to 0
STATUS	Status field.
CRC	CRC field
DATA	Data field of the message. User will supply data information to the protocol.
DONT_CHECK	Any value.
Any character	Explicit value for specified byte.

LISTING 3 Example of the packet definition

```
class Response: public Packet
{
public:
    Response ()
    {
        SetName("Response");
        Insert( 3, "\x02\x00\x01" );
        Insert( 1, Field::LEN );
        Insert( 0, Field::DATA );
        Insert( 2, Field::CRC );
    };
};
```

TABLE 4 Typical communication session fragment

N	PC	Meter	Channel distortion	State of Engine
1s	Give me the table X			INITIAL
1r		Sends first chunk		
2s	Give me more		?	MORE_RECEIVE
2r		NAK		
3s	Give me more			MORE_RECEIVE
3r		Sends last chunk	?	
4s	Re-send last packet			ERROR_LOST
4r		Sends last chunk		FINAL

- Provide a pure data-link channel, that is, payload-only data with no dependencies on selected frames. A virtual channel can be routed between two media and the most efficient frames can be selected (a function of channel distortion) without affecting the clients of this layer
- Hide the unrelated details of the protocol. Every `msgId` represents a group of messages for external use
- Provide error handling: the calculation of error-detecting codes, retransmission, and reordering control

Most of the data-link layer protocols in the industry are block protocols (byte-count-oriented protocols). This means that the structure of the packet has fixed binary fields, and the actual length of the variable-length fields is determined by the value of preceding fixed fields (normally in the same packet).

In this article, a generic implementation of the `DataLinkInterface`, used for block protocols, is presented.

For simplicity, consider the synchronous block protocol. A final goal is to provide a generic implementation for the `DataLinkInterface`, the so-called protocol engine. To achieve this goal, two core protocol attributes—packet format and procedural rules—must be examined.

The following entities of the packet format are of primary interest:

- Data fields—fields containing the user data
- Frame fields—header and trailer, like a checksum

Procedural rules include:

- Packet assembly/disassembly
- Actions in error conditions, like request for retransmission

User parameters must be mapped from the `Params` collection to the packet's data fields. To do that, two differ-

Check out our easy
migration path from
pSOS™, VxWorks®, and iTRON™
to OS-9®

Get to market first™

Get a jump on the competition.

If you're about to jump into the market with a new product, make sure you've got an embedded solution that goes the distance. Our solution includes powerful development tools, proven software components like the OS-9® RTOS, and experienced consultants who'll help you succeed. Support for Java™ technology, IEEE 1394, USB, and other connectivity is already built in.

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.



State machines can help represent the procedural rules of the protocol. In every protocol, one can select logical groups of messages, representing part of a protocol functionality.

Scripting languages and packet matching

We can further look at the similarities of regular expressions and protocols. Consider scripting languages in which regular expressions are an inherent part. User data in the send packets are language variables and user data in the received packets are language-predefined names, representing match sub-patterns in the whole regular expression.

In the Perl scripting language, the send packet can be:

```
"\x02".$outParams[0]."-".$outParams[1]."\x03",
```

while receive code for this packet is:

```
if( /\x02 ([^\-]*) - ([^\x03]*)\x03/ ) {
    $inParams[0] = $1;
    $inParams[1] = $2;
}
```

col described in the ANSI Std C12.21-1998 Protocol Specification for Telephone Modem Communication, for an electricity metering device has the **field types** shown in Table 3.²

The packet template is represented by a **Packet** object in the protocol engine. The **Response** packet from Table 1 is shown in Listing 3. The **DATA** field's length is unspecified, which means that it has to be taken from the **LEN** field, which should occur earlier in the same packet template. It is possible to have bit fields in the protocol. Hence, template objects may have bit fields too.

Presented field types in the template objects make assembly and disassembly of packets possible if the protocol engine knows how to ask for the next packet. This is an area of procedural rules of protocol.

Procedural rules

State machines can help represent the procedural rules of the protocol. In every protocol, one can select logical groups of messages, representing part of a protocol functionality. For example, if the communicating peer is designed to give an acknowledgement for every request, then there are at least two packets: request and response. In most protocols dealing with the transmission of large amounts of data, there is a need for a command like "continue." Error handling requires an additional command like "repeat."² For **DataLinkInterface**, these groups of messages are externally represented by the integer **msgID**. This group is called the *message set*. Message sets in the protocol engine are defined by the **Packets** class. This is a collection of packet templates. Think about the message set as a subset of protocol messages, which implements some defined functionality of the protocol. The message sets can be described by the Harel statechart in UML.³

Let's assume that each packet template in the message set has an assigned state, receiving or sending

FIGURE 1 The state machine for synchronous protocols supported by the implemented version of the protocol engine

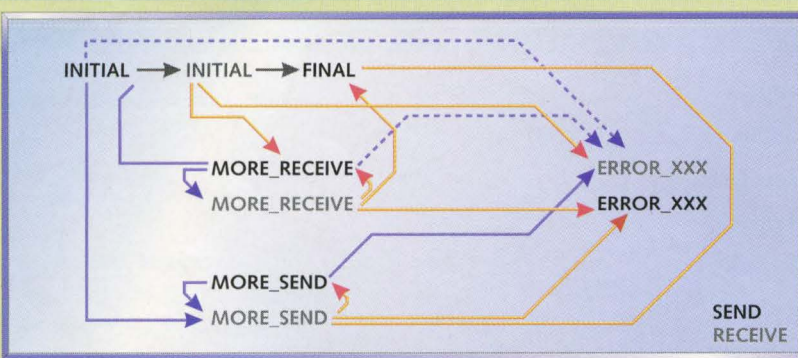


TABLE 5 State transition table corresponding to Figure 1

Event/State		INITIAL (I)		MORE_RECEIVE (MR)		MORE_SEND (MS)	
		S	R	S	R	S	R
INITIAL	S	Ir					
	R		F		F		F
MORE_RECEIVE	S			MRr			
	R		MRr		MRs		
MORE_SEND	S	MSr				MSr	
	R						MSs

ent types of information must be captured. The first is about the location of the data field in the packet. The second is about how to extract the data stream from the sequence of packets.

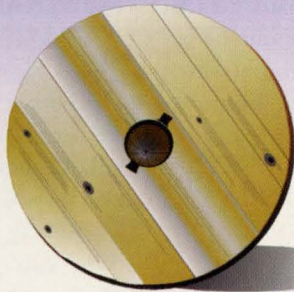
This is done by introducing the concept of the *packet template*. The packet template is a collection of tuples (**field length**; **field type**; or **field value**). Using a communication proto-

DSP Kernel Evolution



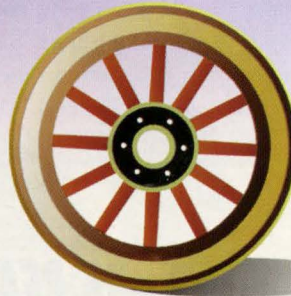
In the beginning. . .

- A small loop
- ASM language
- No standards
- No support



Custom

- A bigger loop
- ASM language
- Primitive scheduler
- No support



1st Generation Commercial

- Multitasking
- ASM & C languages
- Large footprint
- Cumbersome support



RTXCDSP™ RTOS

- Modular Scheduler Architecture
- Written mostly in C
- Stable path to product maturity
- Fast, effective support

This selection's a natural.

We're Embedded Power Corporation. Our new **RTXCDSP™ RTOS** gives embedded application developers a winning head start in four distinct ways:

First, next-generation DSP-based designs require a commercial RTOS. Our **RTXCDSP** frees you from the terrors of tinkering to redesign your legacy custom scheduler, or from the harmful effects of headbanging against the limits of a big-footed slow commercial leviathan from yesterday. Work on what really matters: your application.

Second, **RTXCDSP** replaces design headaches with pleasant surprises like a consistently-shared API. Its flexible scheduler architecture accomplishes both round-robin and multitask scheduling in a remarkably powerful yet small footprint. You can scale the scheduler resources you need for your design.

Third, you'll appreciate that **RTXCDSP** makes it easy to preserve legacy code and maintain legacy test regression suites. It's easy to port to **RTXCDSP** from your current design. We even offer a **SPOX™** Conversion Guide.

As you add application capabilities to support your customer's needs, we'll be there to support you, even if your programming staff changes.

RTXCDSP is a modular design, built to meet all your DSP application development needs:

- A single-stack module provides multithreading and level-specific thread priorities for superfast linear processes using static kernel objects
- A multiple-stack, multitasking module provides intertask communications and synchronization for efficient management of diverse event-source inputs using dynamic kernel objects.

- Dual-mode provides the properties of both single and multiple stack modules together in a single kernel.

Fourth, **RTXCDSP** combines blazing flat-out speed with a footprint so small that several major high-performance portable designs are already underway using **RTXCDSP**.

To get your next DSP design rolling call us today!

Supported Processors

Motorola™ 8101, 563xx, 566xx
StarCore™ SC140

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.

Therefore, the protocol State Machine is actually a Deterministic Finite Automata (DFA). I call it a *scenario's DFA*. The message set defines the scenario's DFA identified by **MsgID**.

the particular packet switches the protocol engine to the corresponding state. Similarly, the protocol engine selects packets to send if the packet

has the current engine's state in its list of valid states. There should be an unambiguous determination of which packet to send. Therefore, the proto-

col State Machine is actually a Deterministic Finite Automata (DFA). I call it a *scenario's DFA*. The message set defines the scenario's DFA identified by **MsgID**.

Consider the following example. This scenario is one of reading a data block from an electricity meter to the PC through the optical infrared protocol. All messages in Table 4 constitute one message set.

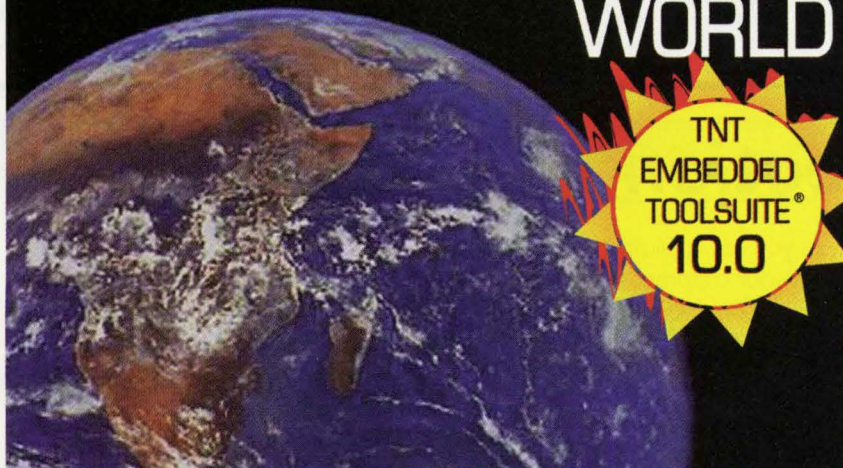
Initially the engine is in the **INITIAL** state, sending request 1s and receiving response 1r. After matching the response with one of the packets in the message set, this response can switch the engine to state **MORE_RECEIVE** or **FINAL**. The details regarding how it is done are in the next section. Suppose receiving a packet is the first of the two-packet sequence. In that case the engine will continue the session by sending a "give me more" command. Only one send command should be possible in the **MORE_RECEIVE** state. As Table 4 shows, this request has been distorted. Suppose the meter's CRC validation failed. At the 2r step, the PC received **NAK** from the meter. The **NAK** message template does not contain a data field, so the accrual data buffer will not be changed at this step. The engine is still in the **MORE_RECEIVE** state and it repeats the request in the 3s. Suppose the meter received a 3s packet and sent a response, but the response is lost at 3r. The appropriate timeout engine will switch into an **ERROR_LOST** state. Again, in that state the only valid request is "Re-send last packet." The PC receives the packet at the 4r, then the appropriate field in this packet indicates that there is no more data to receive, so the session ends at the **FINAL** stage.

The State Machine for this protocol is shown in Figure 1. The black nodes represent the engine before the packet is sent—"send" nodes. Sending the packet when the engine is in this state brings it to one of the "receive" states, which is marked in gray. Receiving the packet in the "receive"

RTOS^{Win32} with REAL realtime

Phar Lap SoftwareTM

making our mark on the embedded
WORLD



- 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



– Conventional RTOS? ...sorry!

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



OSE THE NEW GENERATION RTOS!

FASTER

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

TOUGHER

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

LONGER-LASTING

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

MORE POWERFUL

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

SIMPLER

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

SAFER

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

PROVEN

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

ENEA OSE SYSTEMS INC

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

www.enea.com

TABLE 6 Templates for the replies of the sample protocol

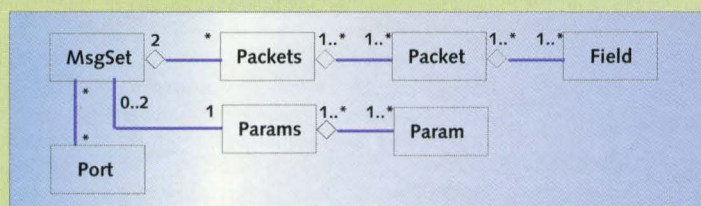
Packet name	Template definition (word, length)
Busy	(02, 1) (03, 1) (STATUS, 2) (CRC, 2)
NAK	(02, 1) (00, 1) (02, 1) (STATUS, 2) (CRC, 2)
Response	(02, 1) (00, 1) (01, 1) (LEN, 2) (DATA, ?) (CRC, 2)

TABLE 7 Example of packet parsing by the protocol engine

N	Byte stream	Fetch length	Unprocessed bytes in buffer	Backtrack length	Currently matched packet
1	02	6=min{6,6,7}	0+5=5		Busy
2	00		4	1	NAK
3	01		3	1	Response
4	00		2		Response
5	02		1		Response
6	19	3	1+2=3		Response
7	73		2		Response
8	xx		1		Response
9	xx		0		Response

TABLE 8 Protocol engine object descriptions

Object	Description
Field	Field object, represented one field of protocol message template
Packet	Protocol message template, collection of Fields
Packets	Collection of Packets
Param	User-specified parameter for protocol message
Params	Set of Params
MsgSet	Two Packets collection, for send and receive messages
Port	The interface with communication medium. Declares Send() and Receive() methods for byte stream

FIGURE 2 Protocol engine class diagram

state moves the engine into one of "send" states, and so on. The DFA handles the packet assembly/disassembly via **MORE_SEND** and **MORE_RECEIVE** states. All errors are represented by the **ERROR_XXX** state; error recovery has not been shown for simplicity purposes. The processing of the packet, sending or receiving, generates an event. For instance, it can be a **MORE_SEND** event when the user data is too large to be sent in one packet. For receive, it can be **MORE_RECEIVE**, if the field

PACKET_SEQUENCE is not zero.

The well-known way to implement the scenario's DFA is through a transition table. However, such tables are difficult to read, especially taking into consideration different table compression methods. I decided to use the association of event and packet template to specify the scenario's DFA. Thinking about objects/events is easier than dealing with indices in a three-dimensional array. The example of the transition table, corresponding to

Figure 1, is shown in Table 5 and can be derived from associated events in the message set.

The State Machine of the whole protocol is a DFA, created by parallel combinations of DFAs for all message sets of the protocol.

Packet matching

The last issue in the protocol engine is the processing of received packets. While the scenario's DFA is at the packet level, there is a *frame's* DFA at the packet's bytes level (for block protocols). The task of the frame's DFA is to match the received packet with the template from the message set.

In theory, the receiving part of protocols is more complicated than the sending part. The engine's receive part deals with packet matching logic.

Packet matching is:

- Receiving a byte stream from a communication channel
- Finding the packet template, which can represent the read byte stream or report about a failure to match

Packet matching is similar to regular expression matching. However, there are three differences:

- The number of bytes in the set, in some cases, is defined by the value of the previously encountered set⁴
- The *-closure is absent for block protocols. This simplifies implementation: no loops are hanging on the input stream
- Better control on the input stream is required. Words are not homogeneous; sometimes their sizes are not known at some stage of the packet reading. The packet's template matching cannot rely on the end-of-stream event. There is always uncertainty: either there is an end of the packet or the tail is lost

The exact alphabet for packet matching is byte values [0,255] plus all Field IDs from the message set.



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

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

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

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

**And that, makes you
stand out from the crowd.**

Lynx—the Leader in Embedded Linux

www.lynx.com

Lynx



BlueCat™
Embedded Linux from Lynx

The viable prefix term is borrowed from compiler theory, where it is a set of symbols on the stack of an LALR parser.

- Backtrack length

Examples of Field IDs are shown in Table 3.

The structure of the packet from the matching point of view has the following attributes:

- Fetch length
- Viable prefix length

The *fetch length* is the maximum possible length in bytes, which can be read from the input stream before analyzing the data and refining the knowledge about packet field lengths.

The algorithm for finding the fetch length for a message set is as follows:

For each packet template:

- Given the number of bytes already read, determine the field F of the template to be matched with the next byte from the input stream
- From the field F, give the smallest possible number of bytes, which can match remaining fields (For the fields with unknown field, set the least possible length)

The minimum value obtained on step two is the fetch length. The *viable prefix* term is borrowed from compiler theory, where it is a set of symbols on the stack of an LALR parser. In the case of the protocol engine, it is a common suffix of packet templates, which is not yet matched for the given input.

Backtrack length is how far we proceeded from the end of the viable prefix in an attempt to match the packet with the wrong template. Fetch and backtrack lengths are measured in raw bytes; viable prefix is a set of template fields. The template's word following the end of viable prefix determines which packet is matched. Therefore, backtrack length is equal to the size of this word.

The purpose of the viable prefix calculation is performance improvement. Failure to match the packet at the viable prefix means that the packet cannot be matched for any template in the message set. Knowing the viable prefix also reduces the backtrack length. In our implementation, the backtrack length is not greater than one byte.

The finding of the packet template in the message set is done in the order the packets are stored. Storing order is important. The algorithm is as follows:

Applied Microsystems Knows Why CAD-UL is #1.

Applied Microsystems knows who to count on when they need a partner for x86 embedded development tools. And it looks like they're not alone. According to VDC, CAD-UL is the number one revenue generator of x86 embedded development tools worldwide.

Why? Because we're committed to providing a seamless development solution from the day you start writing your code to the day you finish testing it. And, oh yeah, CAD-UL tools work!

Do You?



Applied Microsystems
CORPORATION

- Supports all Intel Architectures
- Full Software Tool/RTOS Support
- Robust Debug, Test & Analysis Tools
- Superior Price/Performance

www.amc.com
(800) 426-3925



- C/C++ x86 Compiler Systems
- Symbolic Debugging Systems
- IDE: CAD-UL Workbench
- CodeCoverage Tools

www.cadul.com
(877) GO CAD-UL

THREADX

Being Royalty Free Is not Enough.

FREE
pSOS+[®]
Evacuation Kit!

Contact Us
Today

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

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

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

Processors Supported

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

the comfortable RTOS



eL
Express Logic, Inc.

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

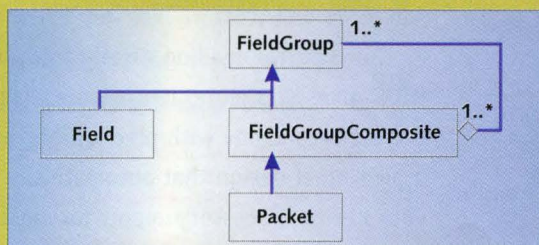
Future development

Consider how the techniques described here can deal with packets, which have groups of fields repeated more than once in the packet. Even if the protocol doesn't have recursively-defined packet field, they can be hidden in the **DATA** field. Later, the application might extract the data from variable length records.

The task can be efficiently resolved with an enhanced version of the protocol engine by applying the Composition design pattern to Field and Packet (see Figure 3). An additional abstract class FieldGroup is needed for the Field and Packet to retain their meaning. The clients still access the Packet object. Clients will use the FieldGroup abstract class to work with the collection. However, since the Packet is not expected to contain only one Field, we can inherit Packet from FieldGroupComposite. The Params and Param objects shall mimic this structure.

The technique was described for synchronous protocols. However, besides the asynchronous queuing system, packet processing is actually synchronous. Therefore, the technique can be applicable to asynchronous protocols.

Figure 3



- Store packets in the ascending order of their initial fetch lengths
- If packets have equal fetch length, store the one with the longer viable prefix first

The purpose of the second rule is to cope with **if/else** logic in packet matching. Message sets can have packets, which are different in only one field, saying "O-ACK, nonzero-NAK." Two template packets might define the actual packet "on the wire": the first should have a constant value of 0 in the status field, and another template will have a field with the undefined data value **DONT_CARE** in place of a status field. If the packets are equivalent in the meaning of the described algorithm, they are stored in the requested order. This order reflects the knowledge about the probability of receiving the packet.

Consider the example in Table 6 (combined from Tables 1 and 2). In the current implementation of the

protocol engine, there is a constraint for only one field with an undefined **DATA** field. At the time this field is matching, the protocol engine has to encounter the **LEN** field in the same packet. The example of matching the byte stream is shown in Table 7.

Table 7 shows the incoming raw byte stream, fetch length calculated by the protocol engine, unprocessed bytes in the read buffer of the protocol engine, backtracking and currently satisfying the input stream template. Every time the engine retrieves the fetch length, it reads the same amount of bytes from the input stream. While the input stream is strictly a forward-moving stream, the internal engine buffer is used for backtracking.

Analysis of three packet templates gives the fetch length equal to six, which is a minimum value in a set {6,6,7} for all packets in the message set, respectively. Packets are matched in the same order as their templates

are sorted in a message set collection. The first byte satisfies the **Busy** packet, but the second byte contradicts the packet template. Two bytes do not contradict the second template; the first packet is removed from consideration. Similarly, the third byte removes the second packet from the candidates. The fourth and fifth bytes give the value for the **DATA** field. All of the fields now have the known length, so the fetch length is three. Although three new bytes are read, one byte is left in the engine buffer. The last step is to match four bytes in the engine's buffer with **Response** template.

Implementation details

Table 8 briefly describes each template object and Figure 2 shows the object hierarchy.

It's logical to implement the protocol object as a separate binary module. Generic **DataLinkInterface** is independent of the data-link protocol, so these protocol modules can be dynamically loaded and registered. Currently, protocol modules are implemented as DLLs. Every protocol DLL has C-style function, which creates a **Protocol** object of class implementing **DataLinkInterface**. For block protocols, implementation is done via the protocol engine.

STL has been used as a generic class library. Since there is no need to insert objects into the collection in a random order, the general collection object is the STL **vector**.

To use the protocol engine:

- Define the **Packets** message set object by inserting **Packet** template objects
- Add the entry to the message set map, assigning **MsgID** to the created **Packet** objects

This map is, in fact, a definition of the whole protocol. There is an instance of the class of **Factory** design pattern to get the instance of message object for a particular **msgID**.

Listing 4 is an example of a defini-




**everything you need to know about
embedded computer design...online**

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
 - And much, much more!

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.

VerticalNet™
Leading Business to E-Business

LISTING 4 Definition of two message sets for the communication protocol IEC 1107

```

class IEC1107ABB::ClassReadShort : public MsgSet
{
protected:
    class Request : public Packet
    {
    public:
        Request()
        {
            SetName("ClassReadRequest");
            AssignEvent(INITIAL);
            Insert( 3, "\x02\x05\x00" );
            Insert( 2, "\x00\x00" );           // The Length: default
            Insert( 2, "\x00\x00" );           // offset for read: default.
            Insert( 1, Field::DATA );           // Class #
            Insert( 2, Field::CRC );
        }
    };
    class ClassReadContinue : public Packet
    {
    public:
        ClassReadContinue()
        {
            SetName("ClassReadContinue");
            ActAt(MORE_RECEIVE);
            AssignEvent(MORE_RECEIVE);
            Insert( 2, "\x02\x81" );
            Insert( 2, Field::CRC );
        }
    };
    class ClassReadNak : public Packet
    {
    public:
        ClassReadNak()
        {
            SetName("ClassReadNak");
            AssignEvent(NAK);
            Insert( 2, "\x02\x05" );
            Insert( 1, Field::DONT_CHECK );
            Insert( 1, Field::STATUS );
            Insert( 2, Field::CRC );
        }
    };
    class Response : public Packet
    {
    public:
        Response()
        {
            SetName("ClassReadResponse");
            AssignEvent(FINAL);
            Insert( '\x02' );
            Insert( 1, Field::DONT_CHECK ); // 0x05 or 0x81
            Insert( '\x00' );
            Insert( 1, Field::STATUS );
            InsertBit( 1, Field::IF_NOT_LAST, MORE_RECEIVE );
            InsertBit( 7, Field::LEN );
            Insert( 0, Field::DATA ); // 0 - don't know data's length
            Insert( 2, Field::CRC );
        }
    };
};

```

tion for the message sets `CLASS_READ` and `CLASS_READ_SHORT`. Listing 5 is a fragment describing registering the protocol object in the protocol plug-in as well as a definition for the protocol templates map.

The usage of `Protocol` object, which implements `DataLinkInterface`, is shown in Listing 6. This code fragment has been taken from a real project. If the appropriate meter is connected to the PC, this code will read data into the `buffer` from the meter's internal memory.

Many benefits

Modern systems have increasing lag in the performance between the I/O subsystem and the CPU. The minimal overhead added by the engine's abstract objects in most cases is not an issue. Communication between the embedded system and the PC is unlikely to be a bottleneck.

In the end, what we have is better, readable code, and easy maintenance of enhancements. Documentation can be kept up to date through automation. In addition, the GUI protocol analyzers are straightforward. Protocol scenarios with events and states can be presented along with the packet structure.

Moreover, the idea of separating the definition of the protocol from the imperative code can be turned into further advantage. Implementation may process the message sets at the creation time of the protocol object and generate efficient state machine tables.

It is worthwhile to consider the generic approach to implementation, even though your protocol may not have any of the following attributes:

- Large number of packets
- Complicated structure of the packets
- High throughput is not an issue, compared to the maintenance burden

The protocol industry is a rapidly growing market niche. New languages and tools are emerging. However, they

CodeWright® 6.0

The Programmer's Editing System™



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

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

Your development environment.

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

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

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

\$299

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

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

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

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

1-800-675-7793

premia

the source code intelligence company™

LISTING 4, continued

```

public:
    ClassReadShort() : MsgSet(Request(),Response() )
    {
        SetName("ClassReadShort");
        m_responses.Insert(ClassReadNak());
    }
};

class IEC1107ABB::ClassRead : public ClassReadShort
{
protected:
public:
    ClassRead() : ClassReadShort()
    {
        SetName("ClassRead");
        m_sends.Insert(ClassReadContinue());
    }
};

```

LISTING 5 Fragment of the protocol definition, registering two message sets

```

DEFINE_PROTOCOL_ENTRY( MeterCom::IEC1107ABB::ProtOpt )

MeterCom::BCProtocol::MsgSetInfos::INFO ProtOpt::m_packetInfos[] =
{
    DECLARE_MSGSET_MAP(CLASS_READ,          IEC1107ABB::ClassRead),
    DECLARE_MSGSET_MAP(CLASS_READ_SHORT,    IEC1107ABB::ClassReadShort),
    // ...
    { UNKNOWN, NULL }
};

```

LISTING 6 "Hello World!" using the protocol engine

```

#include "MeterCom.h"

using namespace MeterCom;

int main()
{
    Session session;

    int ret = session.Open();
    ASSERT(ret==0);

    session.Protocol().Handshake();

    Param cls(1);
    Param buffer(128);

    cls = 9;
    session.Protocol().Communicate(CLASS_READ, &cls, &buffer);

    return 0;
}

```

have not overgrown their initial academic segment. For now, you can stay with your programming language and bring many useful features of specialized protocol tools to mainstream-language implementation. **esp**

Andrey Jivsov is a senior software engineer at Network Associates Inc. He implements various security protocols. Prior to this, he worked for DEC as a system projects lead and ABB Power T&D Co. where he designed and implemented PC software interfaces for intelligent embedded devices. He received his master's degree from Moscow State University of Radio Engineering and Automatics. You can e-mail him at andrey_jivsov@nai.com.

References

1. Certainly, good protocol design simplifies implementation. Generic headers make parsing easy. However, in many cases either protocols overgrew their original designs, or minimizing the overhead was a key design issue. If the payload content can vary, you still have the same problems of mapping it to your run-time structures.
2. Sliding window protocols are resource-consuming for embedded systems.
3. In Reactive Modules, it has an analogous entity called an *atom*.
4. The most advanced regular expression I have seen is in the compiler tool flex, where you can specify the number of characters in the set, but this number is a constant.

Resources

- Aho, Alfred V. *Compilers, Principles, Techniques, and Tools*. Reading, MA: Addison-Wesley, 1985.
- ANSI Std C12.21-1998 Protocol Specification for Telephone Modem Communication.
- Holzmann, Gerard J. *Design and Validation of Computer Protocols*. Englewood Cliffs, NJ: Prentice-Hall, 1990.
- Gamma, Erich et al. *Design Patterns. Elements of Reusable Object-Oriented Software*. Reading, MA: Addison-Wesley, 1995.

Testing Special Cases

by Michael Aivazis

You'll recall that in our last column, we built a scaffolding class to test how our **BankAccount** class would handle simple withdrawals and deposits. Now we need to make sure that our class will perform properly under unusual circumstances. Bank accounts must be able to handle situations in which someone withdraws the entire balance or tries to withdraw more money than they have in the account. The scaffolding class below will test how our **BankAccount** class handles these situations:

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

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

        account.deposit(20);
        account.deposit(30);
        account.withdraw(7);
        account.withdraw(43);

        if (account.balance() == 0) {
            cout << "Your balance is 0" << endl;
        } else {
            cout << "Error!" << endl;
        }

        try {
            cout << "\nwithdrawing $3";
            account.withdraw(3);
            cout << "\nreturn from withdraw";
        }

        catch(BankException) {
            cout << "\ncatching BankException";
        }

        cout << "\ndone ";
    }
};

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

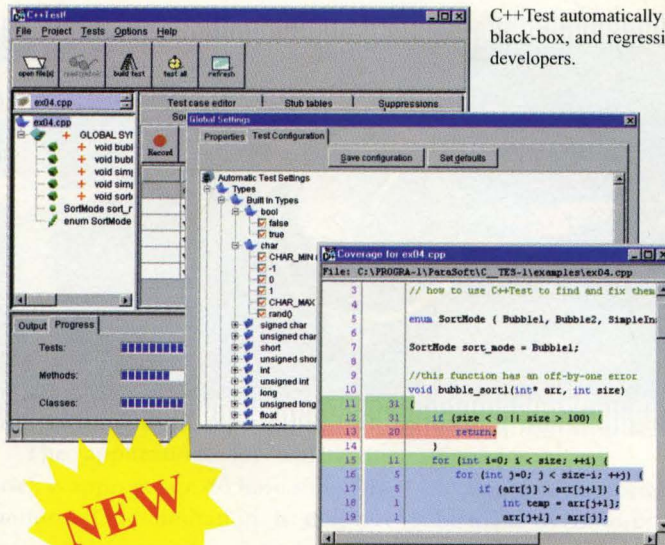
    return 0;
}
```

In the code above, we brought our balance up to \$50, withdrew \$7 and \$43 to bring our balance back down to 0, and then instructed the class to print one of two messages. We then tried to withdraw another \$3, but the class caught an exception. Our printouts tell us that our class can handle a balance of 0 and that it will prevent us from withdrawing more money than we have available:

```
Your balance is 0
withdrawing $3
catching BankException
done
```

Perhaps you are thinking, "Why must I write a whole new class to test my class?" You may think this is a redundant exercise because the class will be tested with the rest of the application later on. However, when you test at the class level, the errors are easier to find and fix.

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



C++Test automatically performs white-box, black-box, and regression testing for C and C++ developers.

With C++Test, you can specify test parameters globally or for the current project.

C++Test's coverage analysis feature graphically displays the lines of code that are being tested.

It's finally here... automatic C/C++ unit testing!

Customizable error prevention for C/C++ developers

C++Test™ is a new tool from ParaSoft that automates C and C++ unit testing, making it easier to find and fix errors before they mushroom into more serious problems. Testing with C++Test is automatic, but it leaves you in control. For the quickest testing, you can click once to have C++Test open, build, and perform white-box testing on your files one after another. For more complete testing, you can modify test parameters and let C++Test perform automatic white-box, black-box, and regression testing.

White-box testing tests the construction of your code. C++Test performs white-box testing by automatically generating and executing test cases to see how files, classes, or methods behave when they are passed unexpected inputs. You can control what types of test cases C++Test generates by specifying what kinds of arguments to use, how many test cases to run, and how deeply to search your embedded classes. C++Test will also build stub functions to test methods that call other functions.

Black-box testing verifies that each class behaves according to specifications. C++Test performs black-box testing by automatically generating and executing test cases and displaying the outcomes, which you can verify with one click of

the mouse. You can also enter your own black-box test cases to be executed by C++Test.

Regression testing verifies that your modifications corrected problems and did not introduce new problems into your code. C++Test performs regression testing by executing the test cases that were run the previous times the class was tested and checking to see if any outcomes have changed. If outcomes have changed or if C++Test finds exceptions, it will report errors. You can replay the test cases each time you modify your code.

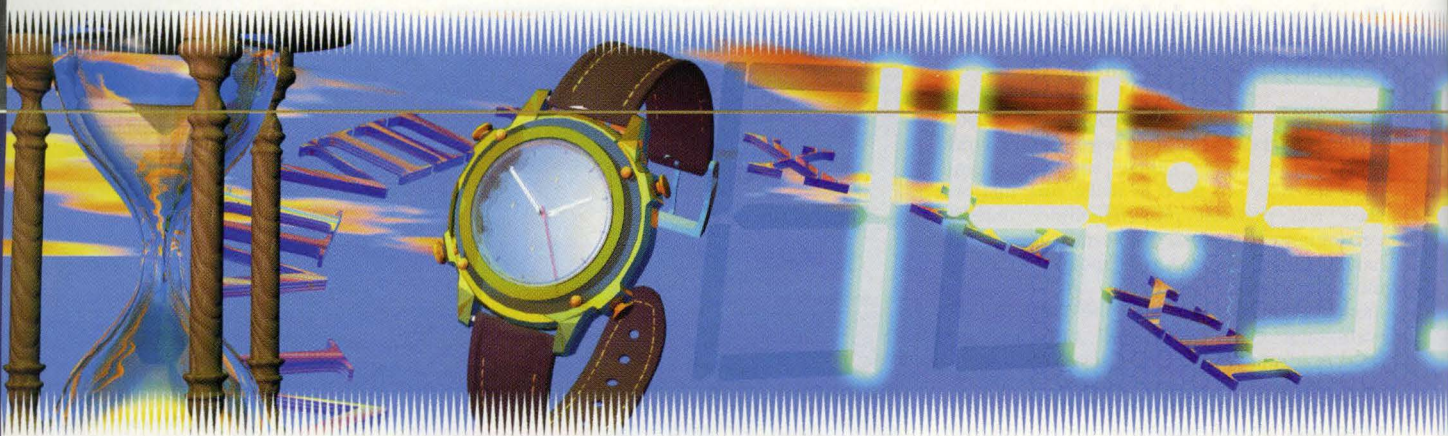
Of course, unit testing is just one stage of the error prevention process for C and C++ developers. C++Test works with ParaSoft's CodeWizard® to enforce coding standards automatically each time you build a file. C++Test also works with ParaSoft's Insure++® to give you automatic runtime error detection each time you test a class or method. Ask about ParaSoft's specialized Developers Kits for your development team.

If you would like the satisfying experience of testing automatically at the unit level, please take a moment to download a fully-functional demo copy of C++Test today at www.parasoft.com/best.htm, or call us at (888)305-0041 for more information.



www.parasoft.com/best.htm

ParaSoft is a registered trademark, CodeWizard and Insure++ are registered trademarks of ParaSoft, and C++Test is a trademark of ParaSoft.



Timing Tricks for PICs

The upside of using a microcontroller with single-cycle instructions is the ease and precision with which your code can synchronize its timing. This article presents five timing tricks that will help you do just that.

Because of their single-cycle instructions, microcontrollers in Microchip's PIC family are often good choices for applications involving precise timing using instruction execution times as part of the solution. Here's a list of useful tricks that can help you implement timing functions in the lowest level Microchip controllers. The code samples will work with most members of the PIC family, but they were originally developed for the 12C5XX and 16C5XX parts.

First, a little background on the PICs and timing is in order. The processor's clock frequency is internally divided by four to develop the basic instruction clock cycle. Each instruction takes one of these cycles to execute, unless that instruction modifies the program counter, in which case an extra cycle is used. This makes counting cycles almost as easy as counting instructions. The code samples that follow all assume a 4MHz processor clock, so each instruction cycle is one microsecond. At least one timer (called Timer 0) can be configured to count instruction cycles, with or without a prescaler. This is an 8-bit counter that can be read or written, and it can be configured to cause an interrupt when it wraps around to zero.

In the following examples, instruction timing is a central theme. In many cases, the use of instruction timing can

avoid the need for additional hardware, such as input capture, pulse width modulation, or a serial port. In some cases this additional hardware is available in higher level PICs, but these examples have been designed to work on the simplest PICs.

Synchronizing to a timer

Let's suppose that you have configured Timer 0 to count instruction cycles, and that you want the code to take a certain action exactly when Timer 0 reaches a specific value. For example, you might want to pulse an output pin when Timer 0 reaches 7. A straight-forward polling loop can synchronize your code to within four counts of the correct timer value:

```
movf    TMRO,w
movwf   Temp
movlw   4
L1:     addwf  Temp,f
        btfss STATUS,C
        goto  L1
```

After falling out of this loop, Timer 0 will have one of the values 1, 2, 3, or 4. These values are derived by starting with Timer 0 reading back as hex FC, FD, FE, and FF in the first

instruction and then counting forward by instruction cycles. These are the only four values that cause an immediate exit from the polling loop. Any other value of Timer 0 will cause additional iterations that take exactly four cycles each to execute, thus allowing Timer 0 to advance by the same amount.

The overall effect is that the loop always exits with Timer 0 at 1, 2, 3, or 4. Corresponding to each of these Timer 0 values, we have values for Temp of 0, 1, 2, and 3, respectively. We can use the precise value of Temp at the loop exit point to remove the Timer 0 ambiguity. Provided the code is in page 0—where adding to the program counter (PCL) works properly—you can follow the polling loop with

```
movf Temp,w
addwf PCL,f
nop
nop
nop
```

After this code executes, Timer 0 can only be 7. Note that this degree of accuracy cannot be achieved with interrupts, since some PIC instructions take two cycles. Because interrupts occur on instruction boundaries, there is an unavoidable ambiguity in the interrupt time. However, this ambiguity could be removed by reading Timer 0 once more within the interrupt service routine and using the same code synchronization trick.

Because of instruction pipelining, it is perhaps a little unclear what is meant by Timer 0 having a specific value when a certain instruction executes. For example, when running the Microchip simulator, MPLAB, you may try to single-step through a `movf`

`TMR0,w` instruction with TMR0 showing 2. After the instruction, `w` will contain 3, because the timer increments before its value is loaded into `w`.

The applications for which this trick is appropriate do have some limitations. One limitation is that the setup time sets a lower limit on the minimum time between output events that are synchronized in this fashion. If this limitation is unacceptable, consider the use of pulse width modulation (or output compare) hardware.

Synchronizing to an input event

If an input event causes an interrupt, your interrupt service routine is synchronized to the event to the resolution of one instruction, which may be two cycles. If this resolution is good enough, the interrupt solution is the simplest (if the PIC you want to use supports interrupts). But can you synchronize code to an input event to the resolution of one cycle? The answer is yes if you can tailor the input event to be of a certain form. Let's start with the brute-force approach:

```
;Wait for PORTB.0 = 0
L1: btfsc PORTB,0
    goto L1
```

This polling loop will drop out when bit 0 on PORTB reads 0 and the synchronization ambiguity will be three cycles (the round-trip loop time). Of course, this loop suffers from the lack of a time-out—in case the event never happens. But putting a time-out check within the loop would complicate what I want to do next (and obscure the principle). Suffice it to say that the following implementation could be modified

to include a software time-out if you really needed one. In my application I use the watchdog timer to get me out of the loop if the event never happens.

Suppose the form of the event can be made to conform to:

. 1 1 1 1 0 0 0 1 1 1 0 1 1 . .

Each value represents the input value for an instruction cycle. If the input event consisted of a single transition from one to zero then our software polling loop could never do better than synchronize to the input event to within three cycles, as shown by the simple three-cycle polling loop above. But this “event” has been made to be more elaborate. The steady stream of ones changes to zeros for three cycles, changes back to ones for three more cycles, and then has one more blip of zero before settling back down with steady ones. This pattern would not be hard to create if it were being generated by another processor that was running off of the same clock source.

Now consider what happens when this bitstream is sampled with the following code:

```
; Wait for a '0'
L1: btfsc PORTB,0
    goto L1
; Then check for '1'
L2: btfss PORTB,0
    goto L3
; Then check for '0'
L3: btfsc PORTB,0
    goto L4
L4:
```

This seemingly meaningless series of instructions will synchronize to the specific event bitstream to within one cycle. Note that a skipped `goto` instruc-

Whenever you write code that relies entirely on instruction times, consider all program branches within the time-critical code to ensure that branches are equalized.

FIGURE 1 Timing relationships between the special input event data and the code at labels L1, L2, L3, and L4

1	1	0	0	0	1	1	1	0	1	1
	L1			L1		L2		L3		L4
		L1		L2			L3			L4
L1			L1		L2		L3			L4

tion takes one less cycle than a `goto` that is not skipped, even if the resulting program counter is the same. To see how this code works, just consider the three possible phases in which the L1 loop could be run with respect to the input bitstream, as shown in Figure 1.

Each of the three rows of labels represents a different phase relationship between the software and the input event. The L1 loop continues to execute until the instruction at L1 first encounters a zero, at which time control passes to L2. But depending on exactly when the transition from one to zero took place, the L2 instruction may see a zero or a one. If it sees a zero it will take three cycles to get to L3. If the instruction at L2 sees a one, it will take only two cycles. Finally, another two/three cycle adjustment at L3 brings the software into a single well-defined phase relationship with the input bitstream at L4.

To be fair, I must point out that this trick may not be necessary with a higher-level PIC with input capture capability. Input capture solves the problem of timing an input event, and it will work with a single transition instead of a specially constructed pattern. But if you want to save money and use the PICs without input capture, then this trick might be just what you need.

I once used this technique to pre-

cisely synchronize several PICs that were all running off the same crystal. One PIC generated the special bitstream shown above synchronized to its own Timer 0 (using the technique of the first trick) and another PIC would synchronize to that bitstream and set its Timer 0 accordingly. Once synchronization was established, the timers on the two PICs maintained that synchronization indefinitely. The application was a multi-channel fuel injector pulse generator. Each channel had its own PIC, but each pulse train needed to be in a precise timing relationship with the other pulse trains.

Error-tolerant serial receive

Many examples have been published by Microchip and others showing how to implement serial communications in software. Timing plays a critical role in all such implementations. Listing 1 shows yet another implementation that is especially tolerant of errors induced by electrical noise. This implementation assumes that the actual RS-232 signal has been buffered and inverted before being connected to the RxD bit of PORTC.

Note that this code doesn't use the timer at all but relies solely on instruction times. The timing has been hardwired to operate at 9,600 baud. To implement other baud rates takes a careful analysis of the design principles so that you know what to change.

The start bit is detected with some error tolerance by waiting for 10 TRUE samples. A FALSE sample merely cancels a TRUE sample by directing the program flow up one notch in the sampling sequence. Using the program counter to keep track of how many TRUE start-bit samples have been detected allows for the fastest possible sample rate—one sample every two or three cycles. The drawback of this method of start bit detection is that it takes more code to wait for more samples. The number 10 was picked as a reasonable compromise. And if the start-bit edge is so obscured by noise that even more checking is required, we would be dealing with noise pulses that last over 20µs, which is a substantial fraction of the bit cell time (104µs). If we have such long noise pulses to deal with, the uncertainty in the start bit's timing would become the limiting factor in the program's noise immunity. Start-bit detection is unavoidably the most vulnerable part of serial data recovery.

After waiting out the remainder of the start bit, the code then integrates the serial bitstream over intervals that correspond roughly to the received bit cells. For each bit, the integral is compared to a threshold to determine the value of the bit. Since each bit cell has 19 samples, the threshold was set at 9 1/2.

Whenever you write code that relies entirely on instruction times, consider all program branches within the time-critical code to ensure that branches are equalized. Fortunately, this was easy to do in this example. The use of the `subwf` instruction to set the carry, which is then shifted into the SerialSR, does not involve any branches. Thus it takes the same time to execute whether the serial data bit is a zero or a one.

The time delay after the start bit is detected, and before data bit cell integration can begin, is picked so that the 19 samples for a bit cell are aligned as well as possible within that

RESUME

Name: HI-TECH C
Address: 7830 Ellis Rd., Suite 105, Melbourne, FL 32904
Phone: 800 735 5715
Fax: 407 722 2902
Email: hitech@htsoft.com
Web site: <http://www.htsoft.com/>
DOB: Jan 1st, 1984
Salary: Peanuts. Really. But a small signing bonus would be nice.

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

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

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

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

Languages: C, assembler.

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

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

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

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

LISTING 1 Error-tolerant serial receive (PIC 16C505)

```

; Getchar: (leaves results in SerialSR)
; (Hardwired for 9600 baud, assuming 4MHz XTAL)
Getchar:
    btfsc PORTC,RxDbit ; Test for start bit
    goto  Getchar      ; Back to the top
GetC2: btfsc PORTC,RxDbit ; Test for start bit
    goto  Getchar      ; Back up the ladder
GetC3: btfsc PORTC,RxDbit ; Test for start bit
    goto  GetC2        ; Back up the ladder
GetC4: btfsc PORTC,RxDbit ; Test for start bit
    goto  GetC3        ; Back up the ladder
GetC5: btfsc PORTC,RxDbit ; Test for start bit
    goto  GetC4        ; Back up the ladder
GetC6: btfsc PORTC,RxDbit ; Test for start bit
    goto  GetC5        ; Back up the ladder
GetC7: btfsc PORTC,RxDbit ; Test for start bit
    goto  GetC6        ; Back up the ladder
GetC8: btfsc PORTC,RxDbit ; Test for start bit
    goto  GetC7        ; Back up the ladder
GetC9: btfsc PORTC,RxDbit ; Test for start bit
    goto  GetC8        ; Back up the ladder
    btfsc PORTC,RxDbit ; Test for start bit
    goto  GetC9        ; Back up the ladder
GetSP: movlw 28          ; Start bit detected
    movwf Temp
GetB0: decfsz Temp,f      ; Wait out the remainder
    goto  GetB0          ; of the Start bit.
    movlw 8              ; Read in all 8 bits.
    movwf Bitcounter
GetcharBitLoop:
    clrf Samplecounter
    movlw 19             ; Each bit cell is
    movwf Temp           ; sampled 19 times.
    nop ; Makes total round trip = 104 cycles
    ; which is 9600 baud
GetcharSampleLoop:
    btfsc PORTC,RxDbit ; If serial data = 1,
    incf Samplecounter,f ; then increment.
    decfsz Temp,f      ; Do it 19 times.
    goto  GetcharSampleLoop
    movlw 10           ; threshold value
    subwf Samplecounter,w ; 0-9 is called '0'
    rrf SerialSR,f     ; 10-19 is called '1'
    decfsz Bitcounter,f ; ..all 8 serial bits
    goto  GetcharBitLoop
    retlw 0            ; Results in SerialSR

```

bit cell, assuming the average start bit detection latency of 1 1/2 cycles. Before the first start bit sample, the code samples the input once every three cycles. Thus, in the case of a noiseless start pulse, the code will be at GetSP at 20, 21, or 22 cycles into the start bit.

The wait loop at GetB0 waits for the remainder of the start bit so that the first of the data bit samples happens about seven cycles into the bit cell. The 19 samples for each bit cell are taken at five cycle intervals, and the time between the last sample of one bit cell and the first sample of the next bit cell is 14 cycles. Thus, the ideal time to begin sampling is seven cycles into a bit cell. The delay at GetB0 has been chosen for exactly that purpose. Under ideal average latency conditions, the bit cell samples are at least five cycles from the bit-cell boundaries. Therefore the three-cycle start-bit detection uncertainty can only lower this margin to about two cycles; all 19 samples are normally from the correct bit cell.

Since the zero/one data bit determination is made based on whether the number of one samples is from zero to nine or 10 to 19, we have nine samples of noise margin. This margin can be used to correct for noise-induced timing uncertainty in the start bit detection, or it can be used to correct for noisy samples in the data bits themselves.

Dithered frequency synthesis

Another technique that is applicable to a wide variety of processors besides PICs involves the synthesis of a particular frequency when the period may not be an integer multiple of the processor clock. A processor can only generate events synchronized to the nearest clock cycle, but sometimes the long-term average period is more important than each individual period. In that case it might be sufficient to dither the generated periods in order to make the

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:

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:

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:

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

DProbe167 and DBox167

DProbeHC12

MX51/AX51

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

The process of repeatedly adding PeriodLo to AccumLo will create just the right number of carries to increment [Thi,Tlo] often enough to make the average period work out perfectly.

LISTING 2 Subroutine to extend Timer 0

```
; ExtendTmr0: Read TMRO and extend it into
; 24-bit tmr0_exhi,md,lo. It works OK if no more
; than 255 instruction cycles elapse between calls.
ExtendTmr0:
    movf    TMRO,w
    movwf   Temp
    subwf   tmr0_exlo,w ; exlo - read of TMRO
    btfss   STATUS,C    ; C if wrap-around
    goto    ExA         ; since the last call
    incf    tmr0_exmd,f ; When wrap-around is
    btfsc   STATUS,Z    ; detected, increment
    incf    tmr0_exhi,f ; 16-bit _exmd & _exhi
ExA:    movf    Temp,w
    movwf   tmr0_exlo   ; _exlo = TMRO as read
    retlw   0
```

long-term average come out right. Of course, it would be best if the dither could be kept to a minimum. In the case of the PIC, as we have seen already, synchronizing to within one cycle is possible. So we use that capability to generate a desired frequency so that the phase difference between the generated signal and an ideal version of the signal with perfectly uniform periods is never more than one cycle.

Suppose that the desired period is expressed as a 24-bit number in the 16.8 format. This means there are 16 bits to the left of the implied binary point and eight bits after it. This will define periods as long as 65535.99 cycles with a resolution of .0039 cycles. For example, hex 23 1C 40 represents 8988.25 cycles. (The .25 comes from taking hex 40 = 64 divided by 256). Suppose PeriodHi and PeriodMid represent the 16-bit integer part of the desired period and PeriodLo represents the 8-bit frac-

tional part. For each generated event, do the following:

```
    movf    PeriodHi,w
    movwf   Thi
    movf    PeriodMid,w
    movwf   Tlo
; [Thi,Tlo] = [PeriodHi,PeriodMid]
    movf    PeriodLo,w
    addwf   AccumLo,f
; If add overflows, then
; increment [Thi,Tlo]
    bcf     STATUS,Z
    btfsc   STATUS,C
    incf    Tlo,f
    btfsc   STATUS,Z
    incf    Thi,f
```

The result of this code is to leave the 16-bit quantity [Thi,Tlo] equal to either [PeriodHi,PeriodMid] or one count higher, depending on whether or not AccumLo overflowed. Then use one of the many techniques to cause the next generated event to occur [Thi,Tlo] cycles after the last such

event. This will result in a long term average period of:

$$\text{PeriodHi} * 256 + \text{PeriodMid} + \text{PeriodLo} / 256.$$

The process of repeatedly adding PeriodLo to AccumLo will create just the right number of carries to increment [Thi,Tlo] often enough to make the average period work out perfectly. This method can be generalized to even higher resolutions by adding more bytes to the right of the binary point. Just be sure to increment the effective period every time there is a carry-out of the accumulated fractional parts.

Extending Timer 0

The timer in the low-end PICs is only eight bits wide. Often it is necessary to extend this timer by synthesizing higher-order bytes. This can be done in software with the subroutine in Listing 2.

If the ExtendTmr0 subroutine is called at least once every 256 cycles, then tmr0_exhi, tmr0_exmd, and tmr0_exlo will be guaranteed to be a consistent extension of Timer 0. It is important to use tmr0_exlo and not any other reading of Timer 0. Only tmr0_exlo is guaranteed to be consistent with the higher-order bytes. This routine works by recognizing a wrap-around in Timer 0 by a transition from higher to lower unsigned values.

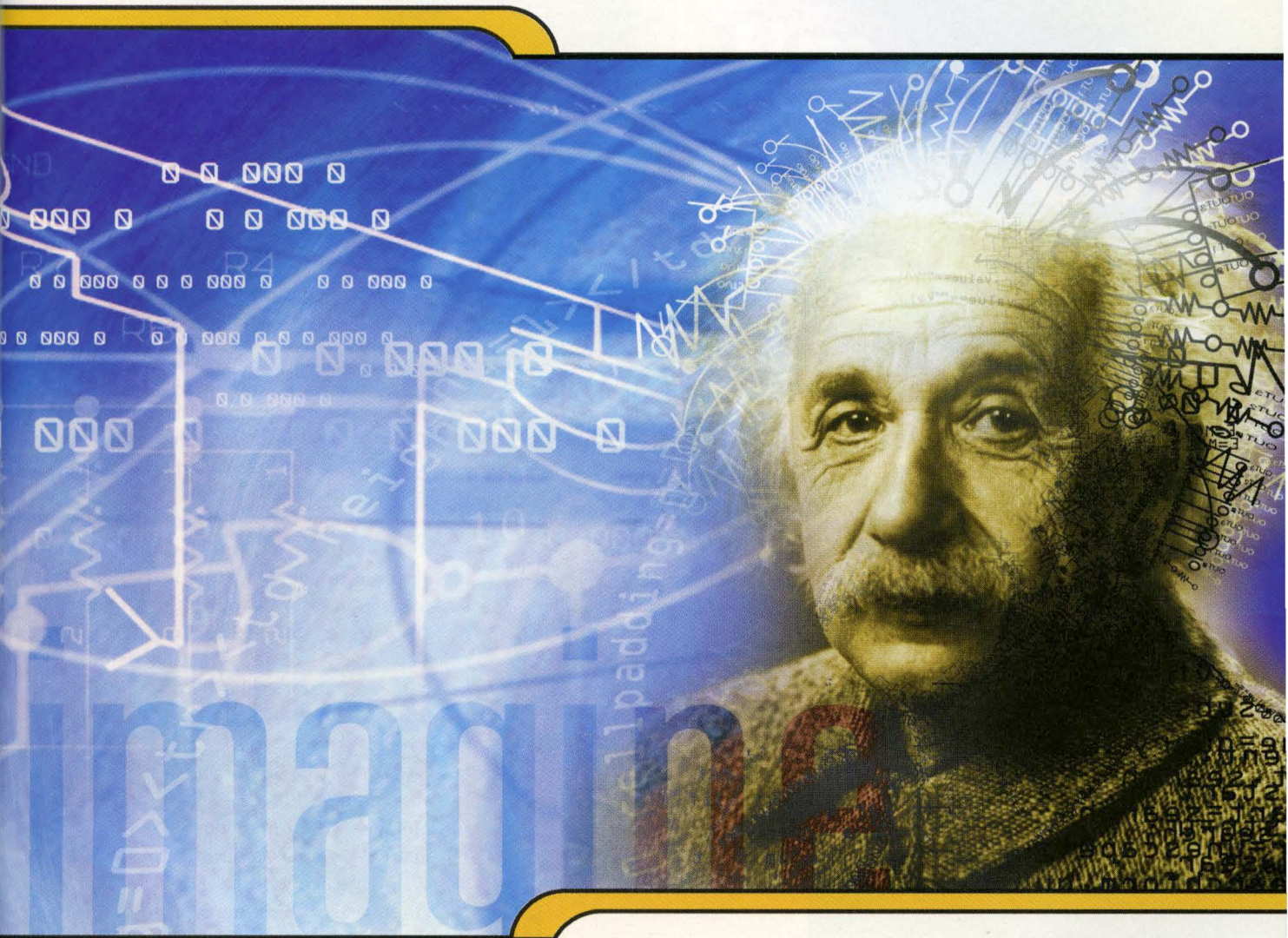
This routine can be combined with the method of synchronizing to Timer 0 to produce a means of waiting for a timer-defined event and synchronizing to that event to within one cycle, as shown in Listing 3.

This routine uses the 24-bit NextTime[0, 1, and 2] as the negative of the desired tmr0_ex value. Keeping NextTime as the negative of the desired time makes it easier to check when tmr0_ex has advanced far enough, because 24-bit addition is slightly faster than 24-bit subtraction.

The maximum round-trip time for the polling loop that checks 24-bit

"Imagination is more important than knowledge"

...Until you have to **build** what you've imagined.



If You Need or Provide These Services:

- I concept
- I design
- I manufacturing
- I support / consulting

Visit and Find:

- I product development RFPs
- I pre-qualified solution providers
- I emerging technology communities
- I imagination and knowledge morphing into reality

Then it's time for Web Product Realization Network.

Where Electronic Product Developers
and Service / IP Providers connect
to make ideas real. Faster.

Are you in or are you out?



(877) 7-WebPRN | getintoit@webprn.com

The Fastest Way to Make Ideas Real.

Of course, modifications to programs that depend on instruction timing are much harder and more error-prone.

LISTING 3 Waiting for a timer-defined Event

```
WaitNextEvent:
; -Wait for TIMER + NextTime to be positive....
    call    ExtendTmr0
    movf    tmr0_exhi,w
    movwf   Temp2
    movf    tmr0_exmd,w
    movwf   Temp1
    movf    tmr0_exlo,w
    addwf   NextTime0,w
    movwf   Temp
    btfsc   STATUS,C
    goto    wne2

wne1 movf    NextTime1,w
    addwf   Temp1,f
    btfsc   STATUS,C
    incf    Temp2,f
    movf    NextTime2,w
    addwf   Temp2,f
    btfsc   Temp2,7      ; Test if Temp2 is '+'
    goto    WaitNextEvent ; [38] max round trip
    movf    TMR0,w       ; [27-35] higher than
    addwf   NextTime0,w   ; in ExtendTmr0
    movwf   Temp         ; so Temp = 27,...,72
    movlw   -73
    addwf   Temp,f        ; Temp = -46,...,-1
    movlw   4

wne4 addwf   Temp,f
    btfss   STATUS,C
    goto    wne4         ; fall out on 0,1,2,or 3
    movf    Temp,w
    addwf   PCL,f
    nop
    nop
    nop
    retlw   0            ; **Exactly Synchronized**

wne2 incf    Temp1,f
    btfsc   STATUS,Z
    incf    Temp2,f
```

time is 38 cycles, so when `Temp` first becomes positive, it is in the range 0...37. After the 24-bit comparison has been satisfied, the timer is read again. This second read of the timer can be anywhere from 27 to 35 cycles after the read that occurred in `ExtendTmr0`. Therefore, when that read is added to the low byte of `NextTime`, the result must be in the range 27 to 72. After adding -73, this range is reduced to -46 to -1, which is then treated just like the `Temp` value in the first trick on this list. When this function returns, the code is exactly synchronized to Timer 0 in a 24-bit sense, with the understanding that a fixed amount of time has elapsed since the Timer 0 was actually equal to the negative of `NextTime`.

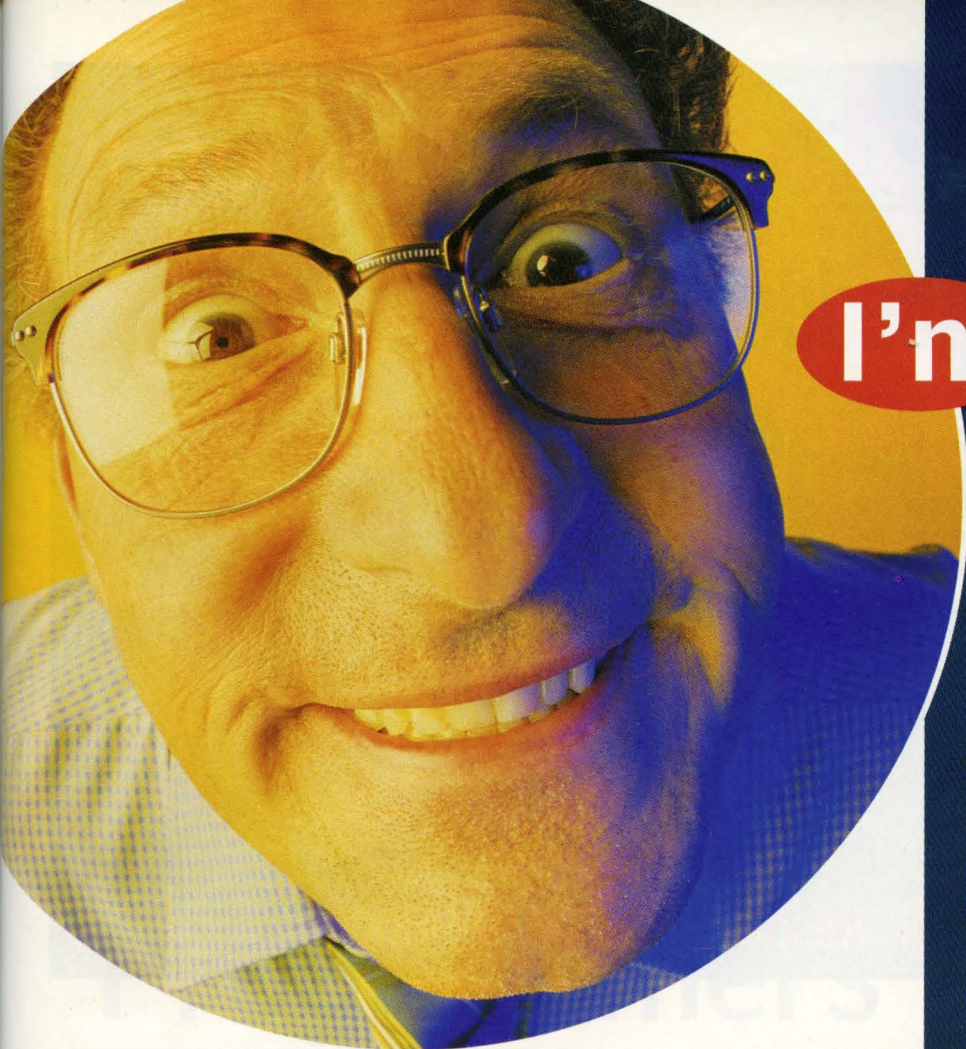
The application may have to take this fixed delay into account if absolute timing is required. In many pulse-generation applications, the absolute time of each pulse is not as important as the relative time between pulses, so this may not be an issue.

Comment, please

You can do a lot of things with a simple timer in conjunction with some instruction cycle counting. The one-cycle instruction timing in the PIC family makes this synergy if not painless, at least tolerable. Of course, modifications to programs that depend on instruction timing are much harder and more error-prone. But that is often the case when you try to squeeze the maximum performance from inexpensive hardware. For maintainability, I recommend using comments liberally to document the timing ramifications of your code.

esp

Robert Scott is a consulting developer of embedded systems for clients in automotive-related industries. When he is not practicing instrument approaches to runway 23L at Willow Run, he can be reached at rscott@wvwnet.net.



I'm flippin'
out!

**Check out the new
technical content!**

You will:

- Learn numerous tricks and tips from over 25 QNX technical workshops
- Discover new third-party products to make your job easier
- Hear about the hottest new embedded trends from our partners, including IBM, Intel, National Semiconductor, Motorola SPS, and more
- Hear first-hand about exciting new QNX developments currently under wraps
- Take advantage of hands-on post-conference training courses
- Enjoy the spectacular natural beauty of British Columbia and cosmopolitan Vancouver
- All this and more where the realtime embedded market meets - QNX's 10th International Technology Conference!

QNX2000

TENTH INTERNATIONAL TECHNOLOGY CONFERENCE

May 14-17, 2000

**Vancouver Convention & Exhibition Centre
Vancouver, Canada**



Platinum Sponsor

For more information on QNX2000,
or to register for the conference, visit

www.qnx.com/qnx2000

or call **1 800 676-0566**



© 2000, QNX Software Systems Limited. QNX is a registered trademark, and 'Build a More Reliable World' and QNX2000 are trademarks, of QNX Software Systems Limited. IBM is a registered trademark of International Business Machines Corporation. All other trademarks belong to their respective owners.

Rent the **Hottest List** around!

Embedded Systems

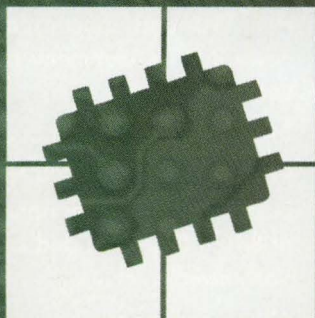
PROGRAMMING

Over 55,000 Qualified Subscribers



Call Rubin Response **847-619-9800**

Your strategic link to the future of embedded technology.



EMBEDDED EXECUTIVE CONFERENCE

May 8-10, 2000
Silverado Country Club & Resort
Napa, California

This exclusive networking and industry-shaping event for top management from the leading embedded vendors is by invitation only. To nominate a VP or chief officer from your company, visit www.embedded.com/exec.

CMP

A 'C' Test: The 0x10 Best Questions for Would-be Embedded Programmers

Pencils up, everyone. Here's a test to identify potential embedded programmers or embedded programmers with potential

An obligatory and significant part of the recruitment process for embedded systems programmers seems to be the "C test." Over the years, I have had to both take and prepare such tests and, in doing so, have realized that these tests can be informative for both the interviewer and interviewee. Furthermore, when given outside the pressure of an interview situation, these tests can also be quite entertaining.

From the interviewee's perspective, you can learn a lot about the person who has written or administered the test. Is the test designed to show off the writer's knowledge of the minutiae of the ANSI standard rather than to test practical know-how? Does it test ludicrous knowledge, such as the ASCII values of certain characters? Are the questions heavily slanted towards your knowledge of system calls and memory allocation strategies, indicating that the writer may spend his time programming computers instead of embedded systems? If any of these are true, then I know I would

seriously doubt whether I want the job in question.

From the interviewer's perspective, a test can reveal several things about the candidate. Primarily, you can determine the level of the candidate's knowledge of C. However, it's also interesting to see how the person responds to questions to which they don't know the answers. Do they make intelligent choices backed up with good intuition, or do they just guess? Are they defensive when they are stumped, or do they exhibit a real curiosity about the problem and see it as an opportunity to learn something? I find this information as useful as their raw performance on the test.

With these ideas in mind, I have attempted to construct a test that is heavily slanted towards the requirements of embedded systems. This is a lousy test to give to someone seeking a job writing compilers! The questions are almost all drawn from situations I have encountered over the years. Some of them are tough; however, they should all be informative.

This test may be given to a wide range of candidates. Most entry-level applicants will do poorly on this test, while

seasoned veterans should do very well. Points are not assigned to each question, as this tends to arbitrarily weight certain questions. However, if you choose to adapt this test for your own uses, feel free to assign scores.

Preprocessor

1. Using the `#define` statement, how would you declare a manifest constant that returns the number of seconds in a year? Disregard leap years in your answer.

```
#define SECONDS_PER_YEAR
    (60 * 60 * 24 * 365)UL
```

I'm looking for several things here:

- Basic knowledge of the `#define` syntax (for example, no semi-colon at the end, the need to parenthesize, and so on)
- An understanding that the preprocessor will evaluate constant expressions for you. Thus, it is clearer, and penalty-free, to spell out how you are calculating the number of seconds in a year, rather than actually doing the calculation yourself
- A realization that the expression will overflow an integer argument on a 16-bit machine—hence the need for the `L`, telling the compiler to treat the variable as a `Long`
- As a bonus, if you modified the expression with a `UL` (indicating unsigned long), then you are off to a great start. And remember, first impressions count!

2. Write the “standard” `MIN` macro—that is, a macro that takes two arguments and returns the smaller of the two arguments.

```
#define MIN(A,B)
    ((A) <= (B) ? (A) : (B))
```

The purpose of this question is to test the following:

- Basic knowledge of the `#define` directive as used in macros. This is important because until the inline

operator becomes part of standard C, macros are the only portable way of generating inline code. Inline code is often necessary in embedded systems in order to achieve the required performance level

- Knowledge of the ternary conditional operator. This operator exists in C because it allows the compiler to produce more optimal code than an if-then-else sequence. Given that performance is normally an issue in embedded systems, knowledge and use of this construct is important
- Understanding of the need to very carefully parenthesize arguments to macros
- I also use this question to start a discussion on the side effects of macros, for example, what happens when you write code such as:

```
least = MIN(*p++, b);
```

3. What is the purpose of the preprocessor directive `#error`?

Either you know the answer to this, or you don't. If you don't, see Reference 1. This question is useful for differentiating between normal folks and the nerds. Only the nerds actually read the appendices of C textbooks to find out about such things. Of course, if you aren't looking for a nerd, the candidate better hope she doesn't know the answer.

Infinite loops

4. Infinite loops often arise in embedded systems. How does you code an infinite loop in C?

There are several solutions to this question. My preferred solution is:

```
while(1)
{
}
for(;;)
```

Many programmers seem to prefer:

```
for(;;)
```

```
{
}
```

This construct puzzles me because the syntax doesn't exactly spell out what's going on. Thus, if a candidate gives this as a solution, I'll use it as an opportunity to explore their rationale for doing so. If their answer is basically, “I was taught to do it this way and I haven't thought about it since,” it tells me something (bad) about them.

A third solution is to use a `goto`:

```
Loop:
...
goto Loop;
```

Candidates who propose this are either assembly language programmers (which is probably good), or else they are closet BASIC/FORTRAN programmers looking to get into a new field.

Data declarations

5. Using the variable `a`, give definitions for the following:

- An integer
- A pointer to an integer
- A pointer to a pointer to an integer
- An array of 10 integers
- An array of 10 pointers to integers
- A pointer to an array of 10 integers
- A pointer to a function that takes an integer as an argument and returns an integer
- An array of ten pointers to functions that take an integer argument and return an integer

The answers are:

- `int a;` // An integer
- `int *a;` // A pointer to an integer
- `int **a;` // A pointer to a pointer to an integer
- `int a[10];` // An array of 10 integers
- `int *a[10];` // An array of 10 pointers to integers
- `int (*a)[10];` // A pointer to an array of 10 integers
- `int (*a)(int);` // A pointer to a

function a that takes an integer argument and returns an integer

h) `int (*a[10])(int);` // An array of 10 pointers to functions that take an integer argument and return an integer

People often claim that a couple of these are the sorts of thing that one looks up in textbooks—and I agree. While writing this article, I consulted textbooks to ensure the syntax was correct. However, I expect to be asked this question (or something close to it) when I'm being interviewed. Consequently, I make sure I know the answers, at least for the few hours of the interview. Candidates who don't know all the answers (or at least most of them) are simply unprepared for the interview. If they can't be prepared for the interview, what will they be prepared for?

Static

6. What are the uses of the keyword static?

This simple question is rarely answered completely. Static has three distinct uses in C:

- A variable declared static within the body of a function maintains its value between function invocations
- A variable declared static within a module, (but outside the body of a function) is accessible by all functions within that module. It is not accessible by functions within any other module. That is, it is a localized global
- Functions declared static within a module may only be called by other functions within that module. That is, the scope of the function is localized to the module within which it is declared

Most candidates get the first part correct. A reasonable number get the second part correct, while a pitiful number understand the third answer. This is a serious weakness in a candidate, since he obviously doesn't understand the

importance and benefits of localizing the scope of both data and code.

Const

7. What does the keyword const mean?

As soon as the interviewee says "const means constant," I know I'm dealing

with an amateur. Dan Saks has exhaustively covered `const` in the last year, such that every reader of *ESP* should be extremely familiar with what `const` can and cannot do for you. If you haven't been reading that column, suffice it to say that `const` means "read-only." Although this answer doesn't



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>



really do the subject justice, I'd accept it as a correct answer. (If you want the detailed answer, read Saks' columns—carefully!)

If the candidate gets the answer correct, I'll ask him these supplemental questions:

What do the following declarations mean?

```
const int a;
int const a;
const int *a;
int * const a;
int const * a const;
```

The first two mean the same thing, namely `a` is a `const` (read-only) integer. The third means `a` is a pointer to a `const` integer (that is, the integer isn't modifiable, but the pointer is). The fourth declares `a` to be a `const` pointer to an integer (that is, the integer pointed to by `a` is modifiable, but the pointer is not). The final declaration declares `a`

to be a `const` pointer to a `const` integer (that is, neither the integer pointed to by `a`, nor the pointer itself may be modified). If the candidate correctly answers these questions, I'll be impressed.

Incidentally, you might wonder why I put so much emphasis on `const`, since it is easy to write a correctly functioning program without ever using it. I have several reasons:

- The use of `const` conveys some very useful information to someone reading your code. In effect, declaring a parameter `const` tells the user about its intended usage. If you spend a lot of time cleaning up the mess left by other people, you'll quickly learn to appreciate this extra piece of information. (Of course, programmers who use `const`, rarely leave a mess for others to clean up.)
- `const` has the potential for generating tighter code by giving the opti-

mizer some additional information

- Code that uses `const` liberally is inherently protected by the compiler against inadvertent coding constructs that result in parameters being changed that should not be. In short, they tend to have fewer bugs

Volatile

8. What does the keyword `volatile` mean? Give three different examples of its use.

A `volatile` variable is one that can change unexpectedly. Consequently, the compiler can make no assumptions about the value of the variable. In particular, the optimizer must be careful to reload the variable every time it is used instead of holding a copy in a register. Examples of `volatile` variables are:

- Hardware registers in peripherals (for example, status registers)

WWW.ARCHIMEDESSOFTWARE.COM

NEW!
8051
VERSION 6

"I have never used a piece of integrated development software as thoroughly integrated as the Archimedes development environment. It has saved me more money in paid labor than the cost of the software. What makes Archimedes product is the simulation capability & profiling. I simply cannot say enough about Archimedes products!"

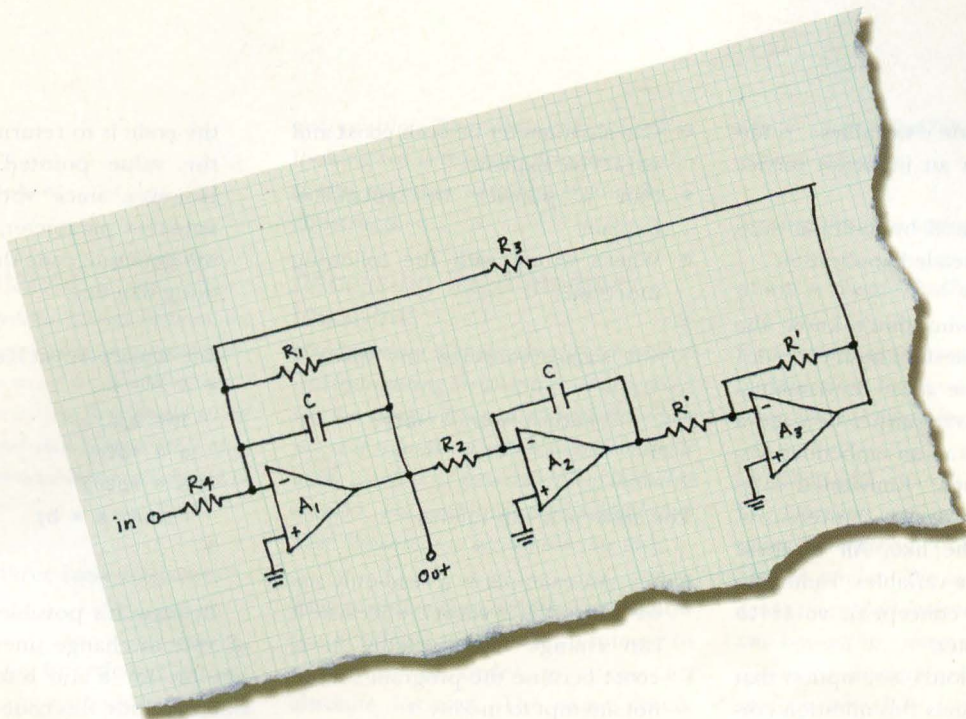
-Paul Buckley
Applied Sciences Group

ide
68HC05
68HC08
68HC11
68HC12
68HC16
683XX
8051
80251
8051XA



1-800-338-1453

303 Parkplace Center Suite G125 Kirkland WA 98033



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.

- Non-automatic variables referenced within an interrupt service routine
- Variables shared by multiple tasks in a multi-threaded application

Candidates who don't know the answer to this question aren't hired. I consider this the most fundamental question that distinguishes between a C programmer and an embedded systems programmer. Embedded folks deal with hardware, interrupts, RTOSes, and the like. All of these require **volatile** variables. Failure to understand the concept of **volatile** will lead to disaster.

On the (dubious) assumption that the interviewee gets this question correct, I like to probe a little deeper to see if they really understand the full significance of **volatile**. In particular, I'll ask them the following additional questions:

- Can a parameter be both **const** and **volatile**? Explain.
- Can a pointer be **volatile**? Explain.
- What's wrong with the following function?:

```
int square(volatile int *ptr)
{
    return *ptr * *ptr;
}
```

The answers are as follows:

- Yes. An example is a read-only status register. It is **volatile** because it can change unexpectedly. It is **const** because the program should not attempt to modify it
- Yes, although this is not very common. An example is when an interrupt service routine modifies a pointer to a buffer
- This one is wicked. The intent of

the code is to return the square of the value pointed to by ***ptr**. However, since ***ptr** points to a **volatile** parameter, the compiler will generate code that looks something like this:

```
int square(volatile int *ptr)
{
    int a,b;
    a = *ptr;
    b = *ptr;
    return a * b;
}
```

Because it's possible for the value of ***ptr** to change unexpectedly, it is possible for **a** and **b** to be different. Consequently, this code could return a number that is not a square! The correct way to code this is:

```
long square(volatile int *ptr)
{
```

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


```
int a;
a = *ptr;
return a * a;
}
```

Bit manipulation

9. *Embedded systems always require the user to manipulate bits in registers or variables. Given an integer variable `a`, write two code fragments. The first should set bit 3 of `a`. The second should clear bit 3 of `a`. In both cases, the remaining bits should be unmodified.*

These are the three basic responses to this question:

- No idea. The interviewee cannot have done any embedded systems work
- Use bit fields. Bit fields are right up there with trigraphs as the most brain-dead portion of C. Bit fields are inherently non-portable across compilers, and as such guarantee that your code is not reusable. I recently had the misfortune to look at a driver written by Infineon for one of their more complex communications chips. It used bit fields and was completely useless because my compiler implemented the bit fields the other way around. The moral: never let a non-embedded person anywhere near a real piece of hardware!
- Use `#defines` and bit masks. This is a highly portable method and is the one that should be used. My optimal solution to this problem would be:

```
#define BIT3 (0x1 << 3)
static int a;

void set_bit3(void) {
    a |= BIT3;
}

void clear_bit3(void) {
    a &= ~BIT3;
}
```

Some people prefer to define a mask together with manifest constants for the set and clear values. This is also

acceptable. The element that I'm looking for is the use of manifest constants, together with the `|=` and `&=` constructs

Accessing fixed memory locations

10. *Embedded systems are often characterized by requiring the programmer to access a specific memory location. On a certain project it is required to set an integer variable at the absolute address 0x67a9 to the value 0xaa55. The compiler is a pure ANSI compiler. Write code to accomplish this task.*

This problem tests whether you know that it is legal to typecast an integer to a pointer in order to access an absolute location. The exact syntax varies depending upon one's style. However, I would typically be looking for something like this:

```
int *ptr;
```

```
ptr = (int *)0x67a9;
*ptr = 0xaa55;
```

A more obscure approach is:

```
*(int * const)(0x67a9) = 0xaa55;
```

Even if your taste runs more to the second solution, I suggest the first solution when you are in an interview situation.

Interrupts

11. *Interrupts are an important part of embedded systems. Consequently, many compiler vendors offer an extension to standard C to support interrupts. Typically, this new keyword is `__interrupt`. The following code uses `__interrupt` to define an interrupt service routine (ISR). Comment on the code.*

```
__interrupt double compute_area
(double radius)
{
    double area = PI * radius *
```

Realize your Vision!

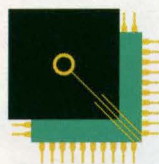
CAN
CANopen
DeviceNet
CAN Analyzers

ICE
RTOS
TCP/IP
Software Tools

Embedded Systems
Academy offers:

- Training courses
- An extensive product line
- Consulting services

ESA will be conducting
classes in 24 cities
around the U. S.
Check our web site for
a city near you.



EMBEDDED
SYSTEMS
ACADEMY

The best endorsement
is what Geoff Lees,
marketing director for
Philips
Semiconductors'
Microcontroller
Business Line says:
"The ESAcademy
hands-on training
classes significantly
shorten the
learning-curve for
new embedded
designs"

Call and register today!

www.esacademy.com

Embedded Systems Academy
Sales: 877 812 6393 Fax: 972 818 0996


```

    radius;
    printf("\nArea = %f", area);
    return area;
}

```

This function has so much wrong with it, it's hard to know where to start:

- ISRs cannot return a value. If you don't understand this, you aren't hired
- ISRs cannot be passed parameters. See the first item for your employment prospects if you missed this
- On many processors/compiler, floating-point operations are not necessarily re-entrant. In some cases one needs to stack additional registers. In other cases, one simply cannot do floating point in an ISR. Furthermore, given that a general rule of thumb is that ISRs should be short and sweet, one wonders about the wisdom of doing floating-point math here
- In a vein similar to the third point,

`printf()` often has problems with reentrancy and performance. If you missed points three and four, I wouldn't be too hard on you. Needless to say, if you got these last two points, your employment prospects are looking better and better

Code examples

12. What does the following code output and why?

```

void foo(void)
{
    unsigned int a = 6;
    int b = -20;
    (a+b > 6) ? puts("> 6") :
               puts("<= 6");
}

```

This question tests whether you understand the integer promotion rules in C—an area that I find is very poorly understood by many developers. Anyway, the answer is that this outputs "> 6." The reason for this is that expressions involv-

ing signed and unsigned types have all operands promoted to unsigned types. Thus -20 becomes a very large positive integer and the expression evaluates to greater than 6. This is a very important point in embedded systems where unsigned data types should be used frequently (see Reference 2). If you get this one wrong, you are perilously close to not getting the job.

13. Comment on the following code fragment.

```

unsigned int zero = 0;
unsigned int compzero = 0xFFFF;
/*1's complement of zero */

```

On machines where an `int` is not 16 bits, this will be incorrect. It should be coded:

```

unsigned int compzero = ~0;

```

This question really gets to whether the candidate understands the impor-



Blackhawk

Ready for a DSP Operating System?

Get Set for a better way to incorporate DSP into your product design.

Go with Blackhawk™ POSIX Operating Systems for DSP Processors.

Blackhawk™ POSIX Operating Systems include:

- Source Code License
- Multiprocessing
- Real Time Extensions
- Multithreading
- Open Source Libraries
- Host File System Utilities
- IEEE Standards Based
- UNIX like

Available standalone or integrated with code generation tools, debugger, simulator, JTAG emulator and reference boards.

Currently supporting TMS320Cxxx™ processors. Phone: 1-877-983-4514

TMS320Cxxx is a trademark of Texas Instruments.

For more information please visit: www.blackhawk-dsp.com

Copyright 2000 EWA

tance of word length on a computer. In my experience, good embedded programmers are critically aware of the underlying hardware and its limitations, whereas computer programmers tend to dismiss the hardware as a necessary annoyance.

By this stage, candidates are either completely demoralized—or they're on a roll and having a good time. If it's obvious that the candidate isn't very good, then the test is terminated at this point. However, if the candidate is doing well, then I throw in these supplemental questions. These questions are hard, and I expect that only the very best candidates will do well on them. In posing these questions, I'm looking more at the way the candidate tackles the problems, rather than the answers. Anyway, have fun...

Dynamic memory allocation

14. Although not as common as in non-embedded computers, embedded systems do

still dynamically allocate memory from the heap. What are the problems with dynamic memory allocation in embedded systems?

Here, I expect the user to mention memory fragmentation, problems with garbage collection, variable execution time, and so on. This topic has been covered extensively in *ESP*, mainly by P.J. Plauger. His explanations are far more insightful than anything I could offer here, so go and read those back issues! Having lulled the candidate into a sense of false security, I then offer up this tidbit:

What does the following code fragment output and why?

```
char *ptr;
if ((ptr = (char *)malloc(0)) ==
    NULL)
else
    puts("Got a null pointer");
    puts("Got a valid pointer");
```

This is a fun question. I stumbled across this only recently when a colleague of mine inadvertently passed a value of 0 to `malloc` and got back a valid pointer! That is, the above code will output "Got a valid pointer." I use this to start a discussion on whether the interviewee thinks this is the correct thing for the library routine to do. Getting the right answer here is not nearly as important as the way you approach the problem and the rationale for your decision.

Typedef

15. *Typedef is frequently used in C to declare synonyms for pre-existing data types. It is also possible to use the preprocessor to do something similar. For instance, consider the following code fragment:*

```
#define dPS struct s *
typedef struct s * tPS;
```

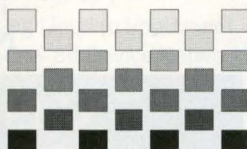
The intent in both cases is to define dPS and tPS to be pointers to structure s. Which

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

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 | **800 428-9340**

Outside U.S. Call 978 448 9340

email: sales@ertfs.com



method, if any, is preferred and why?

This is a very subtle question, and anyone who gets it right (for the right reason) is to be congratulated or condemned ("get a life" springs to mind). The answer is the `typedef` is preferred. Consider the declarations:

```
dPS p1,p2;
tPS p3,p4;
```

The first expands to:

```
struct s * p1, p2;
```

which defines `p1` to be a pointer to the structure and `p2` to be an actual structure, which is probably not what you wanted. The second example correctly defines `p3` and `p4` to be pointers.

Obscure syntax

16. *C allows some appalling constructs. Is this construct legal, and if so what does this code do?*

```
int a = 5, b = 7, c;
c = a+++b;
```

This question is intended to be a light-hearted end to the quiz, as, believe it or not, this is perfectly legal syntax. The question is how does the compiler treat it? Those poor compiler writers actually debated this issue, and came up with the "maximum munch" rule, which stipulates that the compiler should bite off as big (and legal) a chunk as it can. Hence, this code is treated as:

```
c = a++ + b;
```

Thus, after this code is executed, `a = 6`, `b = 7`, and `c = 12`.

If you knew the answer, or guessed correctly, well done. If you didn't know the answer then I wouldn't consider this to be a problem. I find the greatest benefit of this question is that it is good for stimulating questions on

coding styles, the value of code reviews, and the benefits of using lint.

Well folks, there you have it. That was my version of the C test. I hope you had as much fun taking it as I had writing it. If you think the test is a good test, then by all means use it in your recruitment. Who knows, I may get lucky in a year or two and end up being on the receiving end of my own work. **esp**

Nigel Jones is a consultant living in Maryland. When not underwater, he can be found slaving away on a diverse range of embedded projects. He enjoys hearing from readers and can be reached at NAJones@compuserve.com.

References

- Jones, Nigel, "In Praise of the `#error` directive," *Embedded Systems Programming*, September 1999, p. 114.
- Jones, Nigel, "Efficient C Code for Eight-bit MCUs," *Embedded Systems Programming*, November 1998, p. 66.

Free Onboard TCP/IP on this Feature-rich Embedded PC



Call or visit our
website for the Netsock/100
data sheet or a FREE
284-page Handbook

MICRO/SYS
Glendale, CA
(818) 244-4600 • Fax: (818) 244-4246
www.embeddedsys.com



Source Code
Single User License
Immediate Delivery From Web Site

New Lower Prices

C++ Libraries

***Dinkum C++ for VC++**
Dinkum Abridged for VC++
Single User License for \$90

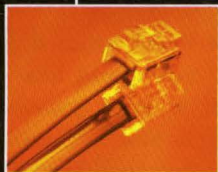
Dinkum Abridged for WIN/CE
includes Dinkum C Library
New Single User License

C Library

Dinkum C Library for VC++
Single User License for \$100

Dinkumware, Ltd.
Genuine Software
www.dinkumware.com

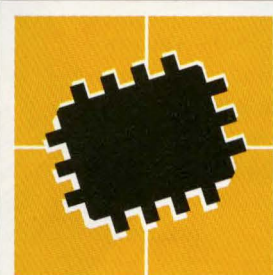
*Dinkumware & Dinkum are Registered Trademarks of Dinkumware, Ltd.
Windows is a Registered Trademark of Microsoft Corporation.



"The emphasis on the classes and tutorials makes this a great conference for continuing education. The 3-day format worked nicely for me."

Brian Hammill, Powerware

Connect to tomorrow's complete embedded solutions.



**EMBEDDED
SYSTEMS
CONFERENCE
SUMMER**

July 10-12, 2000 • Sheraton Boston Hotel • Boston, MA

Get connected to 60 technical classes, 32 expert speakers, and an expanded show floor. Don't miss this opportunity to stay on the leading edge of embedded technologies.

Hundreds of products:

- Embedded Internet Tools
- Development Tools
- Emulators and Simulators
- Real-Time Operating Systems
- Networking Tools
- Systems Design Tools
- Debuggers

Course topics include:

- Working with WAP
- Embed with GNU
- Network Communications for Embedded Systems
- Using Linux to Implement 8- and 16-Bit Device Networking Solutions
- Overview of GPS Protocols

Co-Sponsored by:

EmbeddedSystems EETIMES

SEND ME THE DETAILS! (FAX FORM TO: 972-906-6890)

Name

Company

Title

Address

Suite or Mailstop

City

State/Province

Zip/Postal Code

Phone

Fax

E-mail

esp

www.embedded.com/summer • 800-789-2223 • esc@cmp.com

CMP

Come for the news.



Stay for the whole story.

Week in, week out, industry events occur that can make or break your product, your company, even your career. No one covers those stories—and gets to the *why* behind them—better than EE Times and Electronic Buyers News. That's why these two publications are the trusted news sources for more industry professionals than any others in the business. And why their web sites are key members of the EDTN Network.

But at EDTN, news is just the beginning of the story. Because we know you

need many different kinds of information and levels of detail to make informed vendor, device and tools choices.

That's why we also provide in-depth coverage from the industry's leading design feature publications including Embedded Systems Programming, Electronic Design, Communications System Design, Integrated System Design, Printed Circuit Design and more.

So come to edtn.com for the news. And stay with us for the whole story.

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



It's all right here.



Don Morgan

Music and Noise

Noise, it turns out, is interesting, useful, and provocative. Since I started discussing noise in this column, I have received several pieces of e-mail offering opinions and even philosophy on the subject. Some people try to define noise, most insisting that noise is something with a flat spectrum, like the one we discussed last month with the random number generator. This month, we will look at effects resulting from what might be called noise.

I also received a reference to an article appearing in the *New Scientist* called "Random Reality" by Marcus Chown (available online at www.newscientist.com/features/features.jsp?id=ns22273). The author asserts that reality is really nothing but noise. Of course, I can't disagree with that.

When it comes to music, the purists insist that unless every effort is made to reproduce the original sound of the instrument or voice, we have noise. Thus we have 192kHz sample rates, CDs and DVDs, 1/3 octave and parametric EQ, and expensive audio equipment with 120dB noise floors. Anything less is noise.

There are those who love music who believe that it needs to be treated to be listenable. This month, we will see how noise is used to create music. Actually, what we will be talking about is the use of external phenomena to change the character of the original sound, as opposed to presenting the highest quality reproduction of the original audio possible.

In this issue, we will look at the use of random values to produce a musical effect, white noise to imitate the sound

of a guitar string being plucked, and distortion.

White noise does not seem to be what most people would like to have in their music or used to drive their music. But the nice thing about white noise is that it's a composite: everything exists in white noise. By proper filtering, we can form it, shape it, extract the information we need. You need only a vision of what you want,

formance—the list goes on forever. We could call these small differences noise. And with the use of white noise, for instance, we can recreate effects that are very similar.

Chorusing

If you take two or more vocalists or musicians and have them play or sing the same material in unison, you will

Nothing like a little noise to improve the quality of your music. Don Morgan continues his discussion of the uses of noise.

much like a sculptor sees a figurine in a block of stone. Most people can probably see how that might apply to the creation of artificial voices, but what about the sound of a flute. The tone produced by the flute is really the result of the application of a very specialized filter (the shape and construction of the flute itself) to the noise produced by the air blown across the hole in the mouthpiece. Another use for white noise lies in its very nature: a source of zero-mean random values.

If we remove all those things that we agree correlate from existence, we are left with all the tiny differences that we could say represent the individuality or personality of things. Probably, we can all agree on what a vehicle is. If we then subtract that essence from all those vehicle-things around us, we are left with tiny differences in color, height, magnitude, per-

find that you can still tell that two or more singers or musicians are involved. Even in barber shop, where the goal is to sing with one voice, we are aware that more than one voice is involved.

That is because small differences exist in timing, amplitude, and pitch—again, the list is endless. So, in order to create the effect of a chorus, one cannot simply overlay multiple copies of a single artist's performance. Instead, white noise in the form of slowly formed random numbers is used to generate slight variations in pitch, amplitude, and timing. One can become quite elaborate in this by allowing the variations to affect frequency shifting, as well.

Typically, a chorus is implemented as a sum of copies of the original with the original that are made to vary slowly and randomly with the use of a zero-

mean low-frequency random signal. This is the same type of random signal we created with the random number generators in a previous column.

The delays involved in chorusing are usually kept between 10ms and about 30ms, with some delays longer and others shorter to mask any echo

effect. Remember, each voice or instrument in the chorus must be separately controlled in all the various features you choose to genuinely get the sound of a chorus. Stereo chorusing splits the voices or instruments among the channels with the variations in timing between the voices

adding to the illusion of separation of sound.

An interesting aside to this technique and any that involve delaying copies of a single audio track before summing them is that we are really creating a comb filter with these sums. When we simply add identical copies of audio to one another without delay, magnitude and phase reinforce one another. If we introduce delays—small or large—the instantaneous phase of any portion of the audio may actually sum or subtract. These subtractions result in zeros in the spectrum and the comb filter. Comb filters are often used to create or enhance a sense of spaciousness in the sound.

Flangeing

We have a technique called *flangeing* that is similar to chorusing in many ways. Flangeing introduces a spacious and ethereal quality to sound; some people have described it as a whooshing sound. Jimi Hendrix liked to use it on guitar tracks because it seemed to make the sound surround and envelope the listener.

Originally, flangeing, so it is said, was created by recording the same material on two machines, then playing the recordings back with one slightly delayed—the engineers supposedly pressed his thumb on the flange of the tape payout reel.

Today, flangeing is done electronically. Typically, the two tracks are summed with one delayed by periodically varying the delay between 0ms and 10ms with a low frequency such as 1Hz. These delays can be accomplished with a circular buffer and the delay affecting the offset of the pointer within the buffer.

The flangeing processor is quite simple to define. The output signal is a sum of the original signal and a delayed copy:

$$y(n) = x(n) + ax(n - d(n))$$

We will choose a to be a gain function

smx[®] MODULAR RTOS

SYSTEM SOFTWARE OUTFITTERS



SMX/PEG Evaluation Kit

Includes **smx**, **smx++**, **PEGmt** multi-tasking GUI, and **WinBase** to compile, run, and debug in Microsoft Developer Studio.



Multitasking for x86, PowerPC, and ColdFire



smx. Full-featured, fast, small, preemptive kernel. Supports all x86, PowerPC, and ColdFire processors. x86 support for real mode (w or w/o DOS), 16-bit seg. prot. mode, and 32-bit flat prot. mode. 32-bit works with **pmEasy32**, Win95, 98 & NT. Dynamic load module support.

x86 Protected Mode



pmEasy. 16- or 32-bit protected mode entry, DPMI services, application loaders, Soft-Scope[®] and VisualProbe[®] debugger support.

DOS-Compatible File System



smxFile. Full-featured file manager. IDE, floppy, flash, RAMdisk, PCMCIA, LS120, and SCSI drivers available. **smxCD**. ATAPI CDROM file system.

User Interface



PEG. Portable Embedded GUI. **PEGmt** adds multitasking. Support for **SciTech MGL**, **MetaWINDOW**, and **Zinc** graphics. **smxWindows**. Text windowing library.

Networking



smxNet. TCP/IP. No copy. ICMP, ARP, BOOTP, Ethernet, SLIP, and PPP drivers. FTP, NFS, Telnet, SNMP, DHCP, Web server, HTML v3.2 Graphical Browser, SMTP/POP3 email.

Debugging



smxProbe. Provides tracing and task debugging. **smxAware** & **DEBUG/smx** add **smx**-awareness to debuggers. **WinBase** permits IDE development & debugging at low cost.

C++ Classes



smx++. Class library built upon **smx**. Provides complete support for embedded C++.

DOS & Windows Emulation



X-DOS. Full v5 DOS. **unDOS**. DOS, BIOS, and Windows emulator. **EXE Bootloader** boots from BIOS.

Tool & Board Support from A-Z



AMD	MetaGraphics	SciTech
Beacon	MetaWare	SDS
Borland	MetroWerks	Transvirtual
Diab	Microsoft	VersaLogic
EBS	National	Watcom
Intel	Paradigm	Zinc

NO ROYALTIES • 6 MO FREE SUPPORT • 30-DAY FREE TRIAL

MICRO DIGITAL 1-800-366-2491 www.smxinfo.com/rses.htm

and $d(n)$ to be the delay component. This delay may be attained in any manner you choose, including slow random values, but one good suggestion comes from Orfanidis in his book *Introduction to Signal Processing*. He creates a sinusoidally varying delay that moves between the limits $0 \leq d(n) \leq D$. He defines $d(n)$ as:

$$d(n) = \frac{D}{2}(1 - \cos(2\pi F_d n))$$

Here F_d is a low frequency represented as *cycles/sample*. The result is the frequency response of the time-varying comb filter with the peaks at multiples of f_s/d and the notches at odd multiples of $f_s/2d$. The flanged signal is ultimately computed as:

$$y(n) = \frac{1}{2}[x(n) + x(n - d(n))]$$

Phasing

Another technique that can result in something similar, though not the same, is phasing. Phasing is one of the effects (like clipping) popular among musicians. Here we apply a distorting process to audio to create still another effect. You can create this effect by passing the audio through a very narrow notch filter and combining a proportion of the filter's output with the original signal. You may vary the notch cornering frequency using a variable oscillator or by means of a foot pedal.

Since a high Q filter will have very strong phase shifts around the corner frequency, a high Q notch filter will also manifest these phase shifts on each corner. When mixed with the original audio, these phase shifts produce phase cancellations or enhancements that sweep up and down the spectrum.

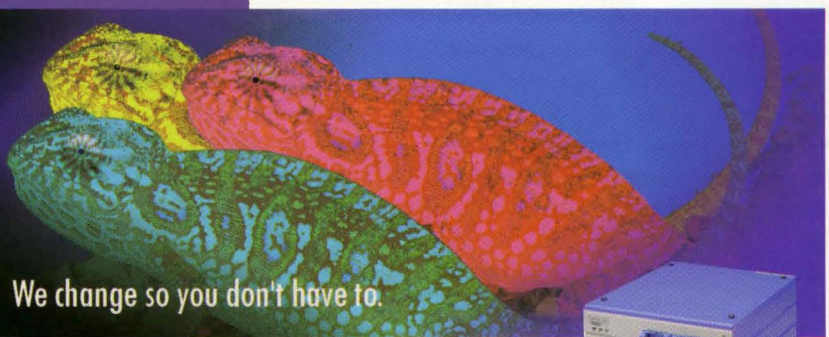
Very often, these notches are created as a system with each filter individually controlled and placed on the frequency axis. In this case, we create a notch polynomial $N(z)$ —place the zeros at the desired notch frequencies

and the poles behind them inside the unit circle. Controlling the length of the radius of ρ (distance from the center of the unit circle to the pole) allows you to vary the notch from very narrow ($\rho = 1$) to wide ($\rho = 0.1$).

There are various techniques available for developing such filters. One

might use second order all-pass filters with a phase response similar to a notch filter, so that when the output of these filters is added to the input, notches are indeed formed. It is also possible to use the state-variable filter discussed in a previous issue ("Simple and Effective," January 2000, p. 87) to

Adaptable In-Circuit Emulators



We change so you don't have to.

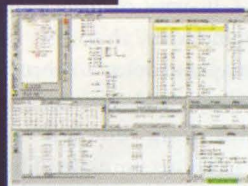
iC1000
8 bit
Emulator



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.

iC2000
8,16,32 bit
Emulator



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.

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

iSYSTEM
www.isystem.com

create the notches; here the amplitude and Q are easily manipulated, or you may use the bilinear transformation. In doing so, we create these filters just as we would any other filter. In brief, the expression for a notch filter is:

$$H(z) = b \frac{1 - 2\cos\omega_0 z^{-1} + z^{-2}}{1 - 2b\cos\omega_0 z^{-1} + (2b-1)z^{-2}}$$

We set ω_0 to $2\omega f$ and b to:

$$\frac{1}{1 + \tan(\Delta\omega / 2)}$$

Q , which is generally accepted to be the ratio of center frequency (ω_0) to bandwidth (ω) is:

$$Q = \frac{\omega_0}{\Delta\omega}$$

As you can see, the factors b and Q are related.

Again, in practice, we can make the features of the notch filter (frequency, Q magnitude) vary with time by using the same sort of mechanism we used in the first two effects. These phasers may be cascaded and even more interesting effects can be generated through independent control over each parameter.

Plucking

Another interesting use of broadband noise, or white noise, to produce music is found in the plucked string algorithm. The sound from a musical instrument changes over time. Here is an algorithm that demonstrates how you can accomplish this effect. It is usually found in music synthesis to produce the evolving sound necessary to imitate musical instruments. Although we're talking about plucking a guitar string here, this algorithm may be applied to other instruments as well.

Real instruments don't have the same sound each time they are

played. Using random numbers means that you can start with random values and adjust them to create an evolving sound. In addition, the random values may be high or low pass filtered to change the character of their sound.

The whole process is very easy:

- Create a short buffer of zero-mean random numbers
- Repeat (re-play) this buffer at a frequency equal to the tone you wish to produce
- Filter or process the values in this buffer with each pass or at regular intervals to get the sound you want

Suppose you want a 1kHz tone. Fill a buffer with 48 random values developed from one of the techniques we have presented and replay that buffer at 1kHz—this will produce your tone. If you wish this to decay as a plucked string might, you could average the values in the buffer with each pass.

A simple moving average at each pass will work. However, it will work differently for tones of different frequencies. Obviously, if you average a buffer 100 times a second, it will decay more slowly than one you average 1,000 times a second. You may use other means, as well, including low and high pass filtering, band passing, boxcar averaging, or FIR filters with long transition bands to shape the evolution of the tone.

If you choose simple averaging, remember that the averaged values may not decay to zero, even with zero mean random values, if the buffer is not long enough and arithmetic errors accumulate. This offset can be kept under control, however, by subtracting a portion of the average of the current values of the buffer from each sample.

Distortion

These popular devices use pre-filtering and post-filtering along with some non-linear component to produce

harmonics, as well as sum and difference frequencies with the original audio.

The filtering is intended to remove the very high end and very low harmonics, as well as the very high and very low sum and difference artifacts, as these are rarely as pleasing as those in the mid-range. The non-linear effect might be clipping or an algorithm to simulate vacuum tube overload.

Depending on the complexity of the device, you may wish to perform band splitting and process each individually before adding them all back together again. This can produce some very interesting results.

Additionally, if you want to increase the number of even harmonics (for a more acceptable and perhaps desirable musical result), you can manipulate the gain for positive and negative half cycles of the input.

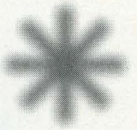
Next month

Among its widening uses, DSP is also popular in motion control. Next month, we'll look at what engineers find so desirable about DSP for this application. **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.

Bibliography

- Orfanidis, Sophocles J. *Introduction to Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1996.
- Kientzle, Tim. *A Programmer's Guide to Sound*. Menlo Park, CA: Addison-Wesley, 1998.



Tools for Embedded Developers

Software

Graphical event viewer

The Graphical Event Viewer is a component of the Illuminator toolsuite. The Graphical Event Viewer allows programmers to view events graphically in the standard message sequence chart format. The Viewer also loads saved charts from file and presents those charts to other charts of a connected system. The Graphical Event Viewer is part of the OSE Evact Handler, a plug-in to Illuminator. The Handler debugs and evaluates a system by allowing a designer to combine a specific event with a specific action. The Evact Handler and the Graphical Event Viewer can function as blocks of processes within an application. The Graphical Event Viewer is scheduled to ship in the second quarter of 2000 along with the Illuminator suite of tools.

Enea OSE Systems

Dallas, TX
(214) 346-9339
www.enea.com

Source-level simulator

The 68000 simulator from Crossware Products creates a virtual 68000 microprocessor on the user's PC and provides source-level debugging facilities without any 68000 hardware. The simulator provides views of memory, registers, and disassembled instructions. The simulator also features multiple watch windows showing the values of local and global C variables; context-colored text windows that allow debugging at the source code level; multiple cycle counters indicating the number of instruction cycles that have occurred during various

parts of the execution process; the state capture facility; source code profiling; and code and data coverage. The creation of custom register windows is also supported. The simulator runs on Windows 95/98 and Windows NT 4.0 and is available now for approximately \$640; the price is about \$350 when purchased as an add-on module.

Crossware Products

Herts, UK
+44 (0) 1763 853500
www.crossware.com

New version of modeling suite

Real-time Studio v. 3.1 adds the ability to import various model elements from Rational Rose, and includes modifications to the Java code generator and its enterprise scalability. Version 3.1 includes a Design View tab to refresh the Synchronizer View. Real-time Studio also features an updated synchronizer which allows changes in a model to be incorporated more quickly than in the previous version. The new version has added grid lines and color to its diagrams. Version 3.1 is available now for a single-user copy of Real-time Studio Professional for \$4,995 and runs on Windows 95, 98, and NT.

ARTiSAN Software Tools

Portland, OR
(503) 245-6200
www.artisansw.com

IDE technology alternative

The pSOS+ Evacuation Kit is targeted at former pSOS+ customers and offers basic pSOS services based on ThreadX

primitives. Former pSOS+ users can migrate their existing application to the ThreadX/MULTI 2000 solution. The pSOS+ Evacuation Kit uses less than 4K of memory for minimal usage. MULTI 2000 and ThreadX are available for most major 32- and 64-bit RISC and CISC target CPUs and for most DSPs. The pSOS+ Evacuation Kit will be offered free of charge for licensed ThreadX customers. It's scheduled to ship at the beginning of second-quarter 2000.

Express Logic

San Diego, CA
(888) 847-3239
www.expresslogic.com

Green Hills Software

Santa Barbara, CA
(805) 965-6044
www.ghs.com

Hardware

Logic analyzer support for ARM9 and ARM9E

The TLA logic analyzer family now supports the ARM9 and ARM9E processors and supports trace port capabilities offered by the ARM Embedded Trace Macrocell. The logic analyzer debugs software code after the design is committed to final silicon. The logic analyzer serves as a raw data collection device by capturing trace port signals. This configuration allows for a direct connection to the ARM9E Trace Port. The TLA logic analyzer can be used as a stand-alone product or as part of an integrated debug environment.

Tektronix Inc.

Beaverton, OR

(800) 833-9200
www.tektronix.com

Emulator supports Intel's Pentium III processor

The newest CodeTAP JTAG emulator was created for the Intel Pentium III Processor-Low Power and offers an integrated development environment with crash-proof run control. The CodeTAP JTAG emulator also offers support of the processor with auto-voltage detection, allows Ethernet communications between target and host debugger without using target resources, provides run control of code execution, delivers fast code downloads, and includes hardware and software breakpoints. The emulator will be available for both PC and Solaris platforms in second quarter of 2000 for \$9,495. The emulator packaged with a Solaris debugger is available now for \$12,495.

Applied Microsystems Corp.
Redmond, WA
(800) 426-3925
www.amc.com

Chips

Flash memory chip with flash protection

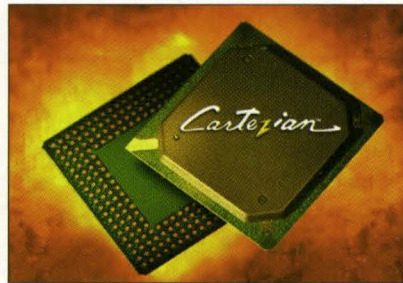
The ST10F168 microcontroller chip includes features that protect the embedded flash memory from code read-out and systems errors. The ST10F168 is comprised of the ST10 16-bit microcontroller core, specially selected peripherals, 256K of internal flash memory, 2K of dual port RAM, and 6K of regular RAM. The ST10F168's peripheral blocks include a CAN interface, 32-channel capture and compare unit, five 16-bit timers, 16 channels of 10-bit analog-digital converters, a four-channel pulse-width modulation unit, an asynchronous communications channel, and 56 interrupt sources. The blocks also include eight channel

peripheral event-controller interrupts which transfer data within one CPU cycle, an on-chip bootstrap loader, an idle mode, and two power-down modes. Samples of the chip are available now for \$25 each in quantities of 1,000.

STMicroelectronics
Lexington, MA
(781) 861-2650
www.st.com

Embedded communications microprocessors

The Cartezian Communications Engine



ziLOG's eZ80 Cartezian Communications Engine

integrates RISC performance and DSP programming. The chip features a synthesizable 32-bit RISC/DSP processor and RISC/DSP edge processors, communications peripherals, and software stacks. It is targeted for the telecommunications and data communications industries. The Cartezian Communications Engine includes several communications peripherals including a 10/100 Ethernet MAC, an ATM SAR with UTOPIA to reserialize the packets, and PCI to provide access to system interfaces. The software stack includes memory-mapped peripherals with API access, codecs, a TCP/IP stack and complementary stacks to enable VoIP and Web management capabilities. The Cartezian Communications Engine Development Platform is scheduled to ship in the fourth quarter of 2000 and the Cartezian Communications Engine is scheduled to ship in the first quarter of 2001.

ziLOG Inc.

Campbell, CA
(408) 558-8500
www.zilog.com

OEM

Plug-in computer

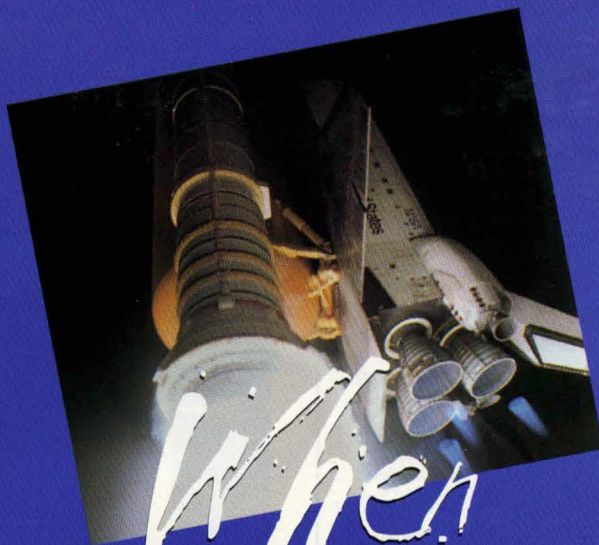
The CPC-2245 plug-in computer is approximately 2.4 in.-by-4 in. It is made up of a 486-level processor, 8KB cache, 32MB of RAM, a 10/100 Base-T Ethernet controller, a PCI slot, a VGA interface with a 64-bit accelerator, two serial ports, a parallel or floppy disk port, and support for up to two hard drives. A CompactFlash socket is also offered for a solid-state disk, and Windows CE can be pre-installed. The CPC-2245 has a DIMM connector which allows it to connect to larger application-specific boards. The device is available now for about \$400.

Advantech Technologies Inc.
Irvine, CA
(949) 789-7178
www.advantech.com

Low-speed CAN interfaces

The Low-Speed CAN interface boards are used with desktop, industrial, and notebook PCs for applications that require connections to low-speed CAN buses. The boards are available in three formats: the PCMCIA-CAN/LS, a PCMCIA Type II card which is PC Card standard-compliant; the PCI-CAN/LS, a short PCI board; and the PXI-8460, a 3U module which is compatible with the PXI/CompactPCI instrumentation standard. The maximum baud rate for the boards is 125Kbits/s and all ship with NI-CAN software. All boards are available starting at \$895.

National Instruments
Austin, TX
(800) 258-7022
www.ni.com



When
people ask
what you do,
tell them
to turn on
the six o'clock
news.

www.boeing.com/employment



EMBEDDED SYSTEMS ENGINEER

MITEQ Inc., a leader in satellite communications systems, is seeking innovative engineering skills with expertise in Embedded Systems Development. We have an immediate opportunity for a dedicated individual with hardware design experience and software engineering talent.

Design embedded digital hardware. Experience with 80386EX, and 8051 microcontrollers, Altera PLDs and Xilinx FPGAs. DSP experience is a plus.

Firmware design and development for embedded systems using C and Assembly language.

Selected candidate must have a BSEE and at least 5 years related experience. Special consideration will be given to those also familiar with microwave communication systems. If you want to join a winning team send us your resume today.



www.miteq.com

100 Davids Drive, Hauppauge NY 11788

fax:(631) 439-9216

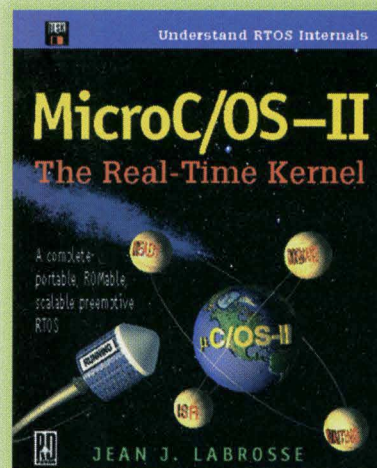
Email: phyland@miteq.com

EEO

The Grass Is Always Greener!

www.rdbooks.com

RD
BOOKS



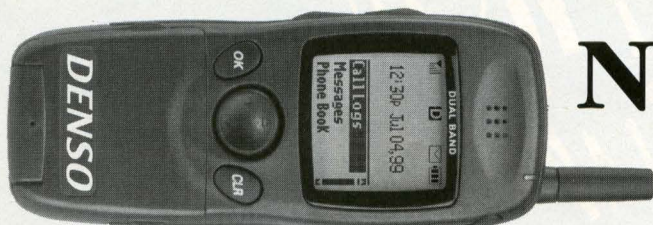
Learn the inner workings of an RTOS! This latest release of *MicroC/OS* has been completely revised and includes many new features. It is a completely portable, ROMable, preemptive real-time kernel. Complete code is included for use in your own applications.

ISBN 0-87930-543-6 • \$69.95

CALL: 800-500-6875 | FAX: 408-848-5784

E-MAIL: rd@rushorder.com

ESPF99



Not just talk.

Cool wireless opportunities in coastal North County San Diego.

Live and breathe technology. With the wireless phones we're making, you'll be connecting with CDMA technology that integrates voice, data/fax, GPS, and the Internet. Our engineers get involved in concept-to-completion projects that create major impact. If you want an environment that's not just great to look at, but where your ideas can get attention, look no further. Industry Week named us one of the "World's Top 100 Best Managed Companies." And our phone with Internet access has just been named one of 1999's Best Products by Business Week magazine. The opportunities are here. What will you make of them?

Join our highly talented team in a small company environment with all the resources you need, from start to finish. We offer generous compensation, a fully-paid benefits program, and a state-of-the-art engineering design facility overlooking the Pacific Ocean in beautiful Carlsbad, California.

EMBEDDED SOFTWARE ENGINEERS

We have positions available at all levels (including recent grads) in the areas of call processing, low level device drivers, and user interface software for current generation and 3G product development. Positions require BSEE/BSCS and 2+ years C and Assembly programming in a real-time, embedded, multi-tasking environment.

DENSO

Wireless Professional Staffing

jobs@densolabs.com (ASCII text only)

Fax (760) 929-3317

5770 Armada Drive

Carlsbad, CA 92008

EOE

DENSO

Real products.
Real opportunities.

densowireless.com

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.

ODETICS

EMBEDDED SOFTWARE ENGINEER

Odetics, recognized as one of the "100 Best Companies to Work for in America", is on the leading edge in developing products deploying advanced information, software and sensor technologies in the areas of surface transportation, broadband communications and precision timing. Currently, we have several positions from mid to senior-level to join our software development teams, where you will be designing and developing real-time embedded systems using C++.

To qualify for these positions you should have experience with the design and programming of real-time systems in such areas as image processing, automatic target recognition applications, GPS systems or telecommunications software.

For more information and specific job descriptions, visit our website at: www.odetics.com SEND RESUMES TO: Dick Fearn, Odetics, 1515 South Manchester, Anaheim, CA 92802. Ph: 714-780-7818; Fax: 714-780-7999; E-mail: dsf@odetics.com (MS Word/text format only) WE ARE AN EQUAL OPPORTUNITY EMPLOYER M/F/H/V

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, C/D/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!

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

EDS OPPORTUNITIES

More than a job, a career path.

At EDS, we can offer you an exciting career in

Embedded Systems Development.

More than 130,000 professionals are enjoying the benefits of an EDS career. We offer excellent training, competitive salaries, and above all, room to grow. When you join EDS, you become part of a dedicated, diverse team helping to make Embedded Systems businesses better. We provide cutting-edge opportunities and insights to leading embedded systems 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

All positions require relocation to Southeast Michigan, as well as strong interpersonal and communication skills. Please mail, fax or email your resume, indicating position of interest to: EDS Staffing, Dept. 72-9350, Attn: Deborah Polvi, 700 Tower Drive, 5th Floor, Troy, MI 48098; Fax: (248) 265-4501; Email: deborah.polvi@eds.com.

EDS

EDS and the EDS logo are registered marks of Electronic Data Systems Corporation. EDS is an equal opportunity employer and values the diversity of our people. Copyright ©1999 Electronic Data Systems Corporation. All rights reserved.

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

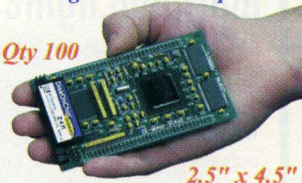
Visual Basic or Visual C/C++

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

C-PROGRAMMABLE SINGLE-BOARD COMPUTERS

FOR REAL-TIME
CONTROL

Z-World's SBCs are ideal for machine control, data acquisition and embedded applications. Complete software development system includes editor, compiler and debugger, saving you time and cost.



OP7100
Touchscreen
from \$549 Qty. 1

JACKRABBIT SBC
from \$49 Qty. 1

PK2200 SBC
from \$295 Qty. 1

Z-World SBCs include:

- Digital I/O to 288
- A/D & D/A converters
- RS-232/RS-485 ports
- Ethernet
- Time/Date clock
- Free tech support

Order online @

www.zworld.com



or call toll free 888.362.3387
2900 Spafford Street, Davis, CA 95616 USA
Tel 530.757.3737 • Fax 530.753.5141

C166 In-Circuit Emulator EMUL166-PC Real-Time Microprocessor Development Tools

Call (408) 866-1820 for a product brochure and a FREE Demo Disk.

Information is also available via Fax, call our 24-hour Fax Center at (408) 378-2912.

Visit our web site- <http://www.nohau.com>

See EEM '97- pages D 1274-1282

NOHAU
CORPORATION

51 E. Campbell Avenue
Campbell, CA 95008-2053
Email: sales@nohau.com

In-Circuit Emulators

SIMULATOR

CHIPVIEW[®]

8051 and derivatives
Intel 80196 / 80296

This remarkable tool utilizes ActiveX technology to provide fast, cycle-accurate simulation of CPU, interrupts, and on-chip peripherals.

Add your own ActiveX controls to simulate off-chip peripherals, ASICs, and even entire systems! Attach to HTML ActiveX controls to mock-up and test your interface!

Interface to your HDL simulator for hardware/software co-verification!

Download a
30-Day Trial Today!
www.chiptools.com



CHIPTOOLS[®]

905-274-6244 FAX: 905-891-2715

DEBUGGERS

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.

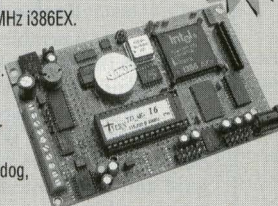
4630 Beloit Drive, #20, Sacramento, CA 95838
FAX (916) 924-8065 • BBS (916) 924-8094
(Mon. - Fri. 8 am - 5 pm, PST)
<http://www.needhams.com/>



i386-Engine-M[™] 8 ch. 16-bit ADC 4 ch. 12-bit DAC

Prices Start at \$199 Qty 1 • \$89 OEM

- C/C++ Programmable.
- 4.5"x2.7", 33MHz i386EX.
- 8 ch. 16-bit 100K HZ ADC.
- 4 ch. 12-bit 200K HZ DAC.
- RS232/485, timers, watchdog, TTL I/O.



C/C++ Development Kit for 40+ Low Cost Controllers with ADC, DAC, solenoid drivers, relays, PC-104, PCMCIA, LCD, DSP motion control, 10 UARTs, 300 I/Os. Custom board design. Save time and money!

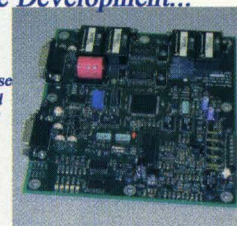
TERN
INC.

1724 Picasso Ave., Suite A
Davis, CA 95616 USA
Tel: 530-758-0180 • Fax: 530-758-0181
<http://www.tern.com>
sales@tern.com



Control Systems Development...
Embedded Systems Engineering...
Real Time Systems Expertise...
Hardware Development...

68332
Multipurpose
I/O Board
w/RS485



Scientific MicroSystems, Inc.

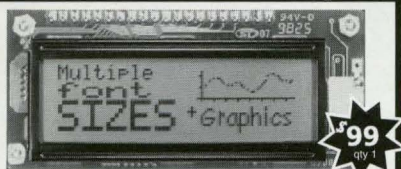
1431 Graham Dr. Suite 218
Tomball, TX 77375

281-351-0202, 1-888-790-1902

e-mail: sales@scimisisys.com

Web Site: www.scimisisys.com

Serial in, graphics out.
Almost too easy.



G12032 Serial Graphics Display
120x32-pixel LCD w/backlight

Jump-start your design project with our easy-to-use serial graphics LCDs. Works like a tiny terminal at 2400 or 9600 bps. Stores custom fonts, bitmap screens in EEPROM. Order today, show a product with a graphics display tomorrow!

www.seetron.com

Scott Edwards Electronics, Inc.
phone: 520-459-4802 fax: 520-459-0623



\$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

UniSTAC™ for Strong ARM designs (SA-1110, SA-1100)



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

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.

Sophia Interface with target

- Sophia original connector-without removing target CPU
- Adapters also available for BGA256.

Advanced GUI source level debugger Watchpoint

Sophia's powerful high-level language debugger hosted on Windows® 95/98, WindowsNT®

Also Supported:

Power PC, ARM7/9186/386/486, Pentium®, 680x0, SuperH, V800/850E, R3000/4000, M16C, M32R

Sophia systems™

408-467-9911

www.sophia.com



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
Siemens • Sharp • SST • Temic • Winbond
• Windows & Mouse User Interface • Real-time/Non-intrusive
• Source-level debug • Rentals available

MetaLink®
Corporation

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-1SD

- 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

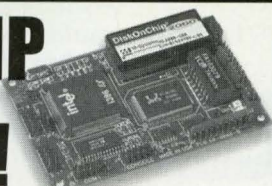
Probes for **16F87x** and **16C77x** by Jan 2000
With data break support

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
TOOL

TechTools™ ROM Tools

FREE CD!



Save Development Time by using a TechTools Emulator in your memory Socket. Modify, download and test code in a matter of seconds. Become even more productive with the Advanced features only available from TechTools.

Easy to use, Economical EPROM Emulator.

High-speed, Reliable, EPROM Emulation at a reasonable price.

- Fast Download (i.e. 512Kbit file in 1.5 seconds or less).
- Fast 90ns & 45ns access times.
- Read-back, Verify and Self-test Functions.
- Full-screen editor, QuickLoader, batchable loader & utilities included.
- All sizes and speeds can be Daisy-chained together and individually addressed from one LPT port.
- Memory retention for true power-up emulation.
- Low voltage Target Adapter with Ext. Power supply available.
- 8 bit & 16 bit Models available up to 4 Mbit.



EconoROM™ II
From \$149.00

Ultra FAST FLASH and EPROM Emulator with powerful special features:
EPROM and FLASH Emulator with Address Snap-Shot & Trigger Circuit.



FlexROM™ II
From \$349.00

- Target Write-back.
- Arbitration support.
- Fast 90ns & 45ns access times.
- Ultra Fast Download (up to 2.5 Mbit per second).
- Daisy-chain port for multi-unit operation.
- Full-screen editor, QuickLoader, batchable loader & utilities included.
- C Library & DLL.
- On-The-Fly editing.
- Snap-Shot circuitry captures the target's most recent access.
- Trigger circuitry generates a trigger each time the target accesses a user-specified address.
- Low voltage Target Adapter with Ext. Power supply available.
- 8 bit & 16 bit Models available up to 8 Mbit.

Advanced Memory Emulator with No-Impact, 'LIVE' Access.



"LIVE" Editor Included

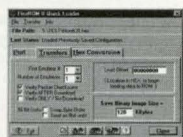
UniROM not only emulates EPROM, Flash and SRAM, but also adds hardware assisted debugging, Live editing, Live Watches and a robust library for custom applications. UniROM is the only Memory Emulator that allows real-time 'Live' editing and monitoring with zero impact on the target system.

From \$595.00



UniROM™

QuickLoader™ Software



Win95/NT QuickLoader Software provides a quick "1 Click" method of converting and downloading files to the emulator from the Windows Environment.

(Included with all FlexROM II and EconoROM II Emulators.)

TechTools

Email Request:

sales@tech-tools.com

Telephone: 972-272-9392

Fax: 972-494-5814

www.tech-tools.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

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

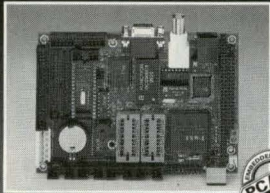
Copyright © 1998, TechTools, P.O. Box 462101, Garland, Texas 75046-2101 • EconoROM, FlexROM, UniROM, QuickLoader, the "Wizard" symbol and TechTools are trademarks of TechTools, P.O. Box 462101 Garland, Texas 75046.

• All other trademarks are trademarks or registered trademarks of their respective company.

Embedded Systems Development Tools

TechTools™

386SX40 PC104 Embedded Controller



ALi M6117D Intel compatible 386SX40Mhz

- 4MB System DRAM on board
- HMC HM86508 VGA Chip w/1MB DRAM
- Support DiskOnChip up to 144MB
- REALTEK 8019AS Ethernet on board
- 4SI/PIIDE/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 PC104 relative products available:
486SX, AC/DC, Digital I/O, Ethernet Link, 4 Port RS232

NUCLEUS Electronic Corp. 800-683-7335
Tel: (909) 468-5700 <http://www.nucleus1.com>
Fax: (909) 468-5704 Email: info@nucleus1.com



Real-Time Debugging

CPU32 ColdFire 68HC12
68302 68020 68306

call toll free

info@noral.com

888-88-DEBUG

www.noral.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.intral.com
email: sales@intral.com
U.S. and Canada: 800.327.7171
Other: 414.273.6100
FAX: 414.273.6106

GALEP-III Pocket Multiprogrammer

**This size
fits all!**

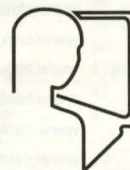


- Programs 8-bit and 16-bit EPROMs, EEPROMs, Zero Power RAMs, Flash, serial EEPROMs • GAL, PALCE, ATF • 87xxx, 89xxx, PIC12/16/17Cxx • All DIL devices **without adaptor** • **Lightning fast** parallel data transfer (e.g. 27C512 read/compare 2 sec) • Power supply independent due to **rechargeable battery** • 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 • '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.contec.com!

GALEP-III Set with cable, battery, recharger... **\$333.00**

PLCC Adaptor for 8-bit EPROMs / 16-bit EPROMs / GALs each **\$149.00**

CONITEC 1951 4th Ave, Ste 301 • San Diego, CA 92101
Tel: (619) 702-4420 • <http://www.contec.com>



Blunk Microsystems

\$5K RTOS

\$5K LAPB

\$5K TCP/IP

NEW! \$5K Flash File System

☒ Royalty-free

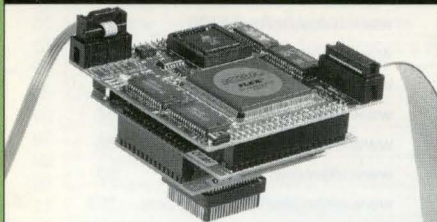
☒ Source code included

☒ **CodeWarrior™** CPU32
and MPC860 Integration

(408) 323-1758

www.blunkmicro.com

8051, 80C196 PIC®, AVR®



Integrated Development Systems: Compilers, Simulators, Programmers, In-Circuit Emulators

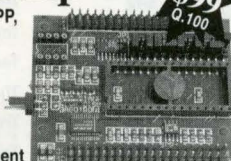
Non-intrusive, with trace feature, hardware
unconditional & complex breakpoints, triggers,
programmable clock, memory mapping & banking,
full project & source level support for C compilers...

www.phyton.com

(718) 259-3191 **Phyt©n**

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



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.

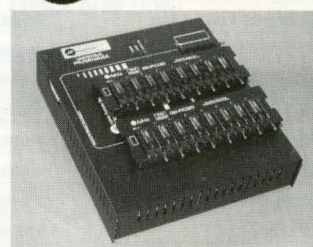
Visit <http://www.bagotronics.com> for info, prices, and FAQs



BAGOTRONIX
2900-1 Crescent Drive
Tallahassee, FL 32308
850-942-7905 phone & fax



Advin



Production and Engineering Programmers Extensive support for

Microchip PICs and others,
Wide variety of package types supported.
Thousands of happy customers worldwide
are living proof of our product quality.

Please contact us and find out why
Advin is your best choice.

- Accept trade-ins of out-dated Data I/O models -

www.advin.com

1-888-GO-ADVİN 408-243-7000

Advertiser	URL	Page	Advertiser	URL	Page
Accelerated Technology	www.atinucleus.com	10	KADAK Products Ltd.	www.kadak.com	55
Advanced Transdata Corp.	www.adv-transdata.com	141	Keil Software Inc.	www.keil.com	87
Advantech Technologies Inc.	www.advantech.com/epc	57	Lauterbach	www.lauterbach.com	49
Advin Systems Inc.	www.advin.com	143	Lynx Real-Time Systems	www.lynx.com	99
Agilent Technologies	www.agilent.com	25	MetaLink Corp.	www.metaice.com	141
Allant Software Corp.	www.allant.com	51	MetaWare, Inc.	www.metaware.com	47
American Arium	www.arium.com	C3	Micro Digital Inc.	www.smxinfo.com	132
ARC Cores Inc.	www.risccores.com	29	Micro/sys Inc.	www.embeddedsys.com	128
Archimedes Software	www.archimedessoftware.com	122	Microsoft Inc.	www.microsoft.com/embedded/	75
ARTISAN Software Tools	www.artisansw.com	45	Microtek International Inc.	www.microtekintl.com	4
Avocet Systems Inc.	www.avocetsystems.com	C4	Microware Systems Corp.	www.microware.com	93
Bagotronix	www.bagotronix.com	143	MITEQ Inc.	www.miteq.com	138
Blackhawk	www.blackhawk-dsp.com	126	Motorola	www.digitaldna.motorola.com	8,9
Blunk Microsystems	www.blunkmicro.com	143	National Engineer Search	www.nesnet.com	139
Boeing Recruitment	www.boeing.com/employment	137	Needham's Electronics	www.needhams.com	140
CAD-UL	www.cadul.com	100	Nohau Corp.	www.nohau.com	C2
Ceibo	www.ceibo.com	23	Nohau Corp.	www.nohau.com	140
ChipTools	www.chiptools.com	140	Noral Micrologics	www.noral.com	143
CMX Systems Inc.	www.cmx.com	50	Nucleus Electronics Corp.	www.nucleusl.com	143
Cogent Computer Systems Inc.	www.cogcomp.com	127	Odetics Telecom	www.odetics.com	139
CONITEC Datasystems Inc.	www.conitec.com	143	Pacific Softworks	www.pacificsw.com	52
COSMIC Software	www.cosmic-software.com	81	Paradigm Systems	www.devtools.com	121
Denso Wireless	www.denso-int.com	138	ParaSoft Corp.	www.parasoft.com	107
Dinkumware, Ltd.	www.dinkumware.com	128	Phar Lap Software Inc.	www.pharlap.com	96
Domain Technologies	www.domaintec.com	142	Premia	www.premia.com	105
EBS	www.etcbin.com	127	Python Inc.	www.python.com	143
EBS	www.etcbin.com	60	QNX	www.qnx.com	41
EDTN	www.edtn.com	123	QNX	www.qnx.com	117
EDTN	www.edtn.com	130	Questlink	www.questlink.com	14
Electronic Data Systems	www.eds.com/careers	139	Rabbit Semiconductor	www.rabbitsemiconductor.com	76
Electronic Engineering Tools	www.eetools.com	141	RadiSys	www.radsys.com/SS7	79
EMAC Inc.	www.emacinc.com	58	RLC Enterprises	www.rlc.com	140
Embedded Power Company	www.embeddedpower.com	95	Scienitific Microsystems Inc.	www.scimisys.com	140
Embedded Systems Academy	www.esacademy.com	125	Scientific Placement Inc.	www.scientific.com	139
emWARE	www.emware.com	71	Scott Edwards Electronics Inc.	www.seetron.com	141
Enea OSE Systems	www.enea.com	97	Signum Systems	www.signum.com	140
esmertec, Inc.	www.esmertec.com	82	Sophia Systems	www.sophia.com	141
EST Corp	www.estc.com	1	Swell Software	www.swellsoftware.com	146
Express Logic	www.ghs.com/armsolutions	21	TASKING	www.tasking.com	43
Express Logic	www.expresslogic.com	101	TechTools	www.tech-tools.com	142
General Software	www.gensw.com	80	Tektronix	www.tektronix.com/	31
Grammar Engine Inc.	www.gei.com	142	Tern Inc.	www.tern.com	140
Green Hills Software Inc.	www.ghs.com	6	Texas Instruments	www.ti.com/sc/tps56300	37
Green Hills Software Inc.	www.ghs.com/armsolutions	21	Texas Instruments	www.ti.com/sc/pci	26,27
HI-TECH Software	www.htsoft.com/pic	142	The MathWorks	www.mathworks.com	77
HI-TECH Software	www.htsoft.com	111	Treck Inc.	www.treck.com	67
Hitex Development Tools	www.hitex.com	141	Trillium Digital Systems	www.trillium.com	88
Hitex Development Tools	www.hitex.com	113	U.S. Software Corp.	www.ussw.com	63
Hiware	www.hiware.com	124	Vertical Net	www.embeddedtechnology.com	103
IAR Systems	www.iar.com	39	Vesta Technology	www.sbc2000.com	141
i-LOGIX	www.ilogix.com	19	WebPRN.com	www.webPRN.com	115
Infineon Technologies	www.infineon.com	12,13	Wind River Systems	www.wrs.com	34,35
Insignia Solutions	www.ignisia.com	85	Working Engine Inc.	www.workingEngines.com	139
InterNiche Technologies Inc.	www.iniche.com	68	Waferscale	www.waferscale.com	72
Introl Corp.	www.introl.com	143	XILINX, Inc.	www.xilinx.com	73
iSYSTEM USA	www.isystem.com	133	ZF Embedded	www.zfmicro.com	16,17
JK Microsystems	www.jkmicro.com/ufish	142	ZWorld	www.zworld.com	140
JK Microsystems	www.jkmicro.com	142			

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
- ☐ 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

Design Tools/Test Equipment

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

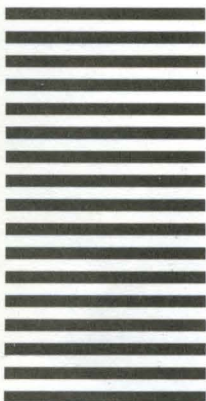
Embedded Systems



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.



Jack G. Ganssle

ESC Chicago

The most recent Embedded Systems Conference Spring, in Chicago, again demonstrated the size and depth of our industry. Two hundred vendors filled the exhibit hall at McCormick Place, showing their niftiest goodies. Many thousands of attendees crowded the show floor, picking up the latest embedded buzz as well as the numerous freebies always available at the booths.

The Embedded show is always much more than just an exhibition; at its core it's an educational program. A decade ago, the West Coast conference originated as numerous seminars augmented by a handful of tabletop vendor displays. That philosophy still reigns. In Chicago this year, nearly 100 classes covered all aspects of embedded development, including real-time UML, using an RTOS, creating predictable real-time designs, code inspections, debugging ISRs, and so on. Once again the TCP/IP seminars were very popular, as developers struggle to incorporate connectivity into their designs.

A couple of themes seemed to pervade the conference. First, not surprisingly, was Internet connectivity. Though I remain somewhat skeptical about the utility of an Internet-aware toaster, the picture painted by many vendors was one of total appliance awareness.

emWare (www.emware.com)—which sells an interesting, though non-standard, connectivity solution for even small 8051-style processors—showed dozens of small single-board computers connected to the 'Net. One corner of the booth was a very cool complete

kitchen—microwave, fridge, counter, and sink. I hoped in vain for some tasty morsels prepared by a bevy of microprocessor droids, but instead got more of a brain-than-stomach fill from their engineering VP.

The refrigerator's door sports a flat panel touch sensitive display linked via emWare's software to the 'Net. Pull a package of frozen chicken out of the freezer and the appliance's bar code

ucts will indeed be foisted off on us as "must-have" gadgets. I remain somewhat skeptical of the utility of many of these ideas.

Other 'Net-connected products—not at the show but clearly possible—do intrigue me. How about glasses (for those of us optically challenged) that project encyclopedic amounts of data to us? Or why not integrate a calculator with our cell phones, so we

The Embedded Systems Conference offered Jack a chance to observe industry trends. Here's what he learned in the Windy City.

scanner identifies the item and sends a request to the 'Net for information about the product. In seconds, dire warnings appear: "This food item has high cholesterol and an excess of calories." As you place the food in the microwave, it, too, looks for information online, this time coming up with cooking recommendations which automatically set the microwave's timer to a suggested value.

Useful? I dunno. Cool? You bet. More importantly, the appliances showed a glimmer of the future touted by this and numerous other vendors. Soon, the thinking goes, everything will be smart and 'Net-connected. We'll be surrounded by incessant information processing, our every action mediated by a microprocessor, almost every thought transmitted instantly to colleagues and friends.

I don't doubt that all sorts of prod-

have one less thing to carry?

As our capacity for technology increases, wise people will assess the cost/benefit ratio of each new capability. Gimmick or useful tool? How will a 'Net-connected toaster improve our lives? Contrary to common belief, the Amish—famous rejecters of anything modern—are not so much Luddites as just very careful consumers of technology. According to a fascinating story in *Wired* (www.wired.com/wired/archive/7.01/amish.html), Amish leaders look for the impact of new products on their social and religious lives. Phones, for instance, encourage interruptions that tear at the fabric of their close family lives. Isn't it amazing how quickly we abandon our family and friends at the first ring of the phone? When I was little, my dad never allowed us to answer the phone dur-

The WRS/ISI deal puzzled many commentators since VxWorks and pSOS traditionally compete head-to-head. Why buy a direct competitor?

ing dinner. At the time I thought he was odd. Now I see the wisdom in not allowing this particular bit of technology to be our master.

Wired, the source of this article on the Amish, is itself an example of misplaced capabilities. All of those bright colors and words buried in pictures hurt my middle-aged eyes. I just can't read it. Ironically the online version is more graphically traditional and much easier to read.

The Wind River show

I lost count of the number of Wind River booths. If you haven't followed this company for the last few months then you probably only know WRS as

the VxWorks people. With the stock market mania driving share prices through the roof and perhaps with a bit of paranoia about the possibility of Microsoft one day getting serious about the embedded space, WRS intends to grow to \$1 billion in sales through acquisitions and increased business. The acquisitions have started, with a vengeance.

Since summer, WRS has acquired Dr. Design, a West Coast design house; TakeFive Software, authors of the SNiFF+ source code analyzer; Diab (compilers); Software Development Systems (another compiler/debugger vendor); and Integrated Systems Inc. (who brought us pSOS).

Prior to the acquisition, WRS and ISI were the two largest embedded tool companies. Now, with some \$300 million in combined revenue they dwarf all others in sales, profits, and show floor space.

The WRS/ISI deal puzzled many commentators since VxWorks and pSOS traditionally compete head-to-head. Why buy a direct competitor? Some wags suggested that eliminating pSOS frees VxWorks to become the dominant OS, creating a powerful barrier to Microsoft's CE or even to embedded Linux. In fact, their latest ad's tag line states, "Roll up your Windows." Others figure it's a way to get a large base of smart employees in a tough labor market.

Regardless, WRS now owns most of the RTOS market, many of the more popular 32-bit compilers, and SingleStep, which might be the most common debugger of all for 32-bit Motorola processors.

ISI customers may be concerned about the future of pSOS. Happily Wind River briefed the press about their product roadmap for both pSOS and VxWorks (the press release is available at www.windriver.com/press/html/roadmap.html). They intend to offer one more release of VxWorks (code named "Cirrus") and one of pSOS ("Stratus"), both in the third quarter of 2000. In 2001 the two products will merge into a new RTOS called "Cumulus" (I see clouds on the horizon), which will maintain compatibility with the pSOS API.

At the show, Wind River announced the acquisition of Embedded Support Tools (www.estc.com), for 6.4 million shares of stock. That's about a third of a billion dollars for a \$28 million dollar outfit, one whose reported profit has never exceeded \$3 million. Not a bad deal—for EST. WRS's current \$2 billion market capitalization, though, means they're flush with buying power.

Why EST? This company, too, specializes primarily in tools for 32-bit

PEG™ does this...

- out of the box
- on your hardware
- with your tools










Portable Embedded GUI

www.swellsoftware.com

Swell Software, Inc. • info@swellsoftware.com • 810-982-5955



New for 2000...

Embedded Systems

P R O G R A M M I N G

CD-ROM Library 1988-1999

The *Embedded Systems Programming* CD-ROM Library Release 5.0 contains all columns, features, news items, editorials, and source code from the 1988 premiere issue through the December 1999 issue. This time-saver contains a powerful text search engine and is a must-have for veteran readers and for those new to *Embedded Systems Programming*, the preeminent source of embedded development for more than 11 years.

Features Include:

- All columns, features, and source code from the premier 1988 issue through the December 1999 issue
- More than 1,200 articles—all columns and features
- A powerful text search engine
- The entire 2000 Buyer's Guide—more than 1,200 products covered in detail
- Code you can copy directly into your designs
- Windows, Unix, Linux and Mac compatibility
- All past and present author biographies
- Links to updated information on www.embedded.com

\$79.95 new \$29.95 upgrade

ways to order:

online
www.embedded.com

e-mail
orders@mfi.com

phone
(800) 444-4881 U.S./Canada
(785) 841-1631 other countries

Order it Online today!
www.embedded.com

The fascinating and perplexing open source movement and its poster child (Linux) was on the minds of developers, seminar leaders, and vendors, and was even the focus of a panel discussion.

Motorola processors. Their BDM and JTAG hardware debuggers, as well as board support packages, are a nice hardware complement to WRS's software offerings. It will be interesting to see how Wind River deals with EST's software debugger, which competes directly with the SingleStep debugger WRS got in buying ISI/SDS/DIAB.

WRS is still shopping, so more mergers are in the offing.

Big vs. small

The fascinating and perplexing open source movement and its poster child (Linux) was on the minds of developers, seminar leaders, and vendors, and was even the focus of a panel discussion. Traditional embedded RTOS purveyors, who may feel little threat from CE, are now under attack on a very different front.

The panel held a wide ranging debate on the value of open source to the embedded community. Under repeated and focused questions from the audience, each of the panel's vendors made it clear, though, that open source is a vehicle to build companies by charging developers for their products and services. Their very clear message: don't confuse "open source" with "free."

For years we've seen the big vendors retreating from 8- and 16-bit operating systems and tools. The Linux craze, too, is a 32-bit venue, one that's creating opportunities and headaches for these high-end companies. Yet 8- and 16-bit processors and their toolchains continue to thrive. For example, CMX (www.cmx.com) showed their beta MicroNet TCP/IP stack with web server that runs on most processors, burning a meager 12K on an 8051. While not 100%

RFC-compliant, the product handles most embedded networking chores adequately. CMX's primary business is real-time OSes for deeply embedded systems. I was delighted to find them handing out a complete price list with their datasheets, a refreshing change from more common complex pricing models. In fact, in a recent discussion about build vs. buy issues with RTOSes, one developer told me they elected to build their own when salesmen from two competing vendors left the developers baffled about costs.

The 8-bit world grew with the addition of two new Z80 derivatives. Zilog (www.zilog.com) showed their eZ80 (a very clever name) while Rabbit Semiconductor (www.rabbit-semiconductor.com) touted the Rabbit 2000. The eZ80 offers binary compatibility with the Z80, yet extends the address space to 16Mbits and cranks the clock to 80MHz. The on-board MAC offers DSP-like performance for some applications.

The Rabbit is more an extension of the Z180 architecture than of the Z80, though it does not offer binary or even true source compatibility. A few Z80 instructions were dropped to pack most instructions into a single byte, while new C-friendly opcodes bring the CPU into the modern era.

Perhaps the best feature is the addition of interrupt levels, which allows easy implementation of critical handlers inside of ISRs and operating systems. The Z80's primitive interrupt structure required an awful lot of work to create true reentrant code.

Both Zilog and Rabbit offer their own proprietary tools, a move which seems a throwback to the early embedded days. Perhaps their rationale parallels that of the open source

dream: getting cheap tools into developers' hands. At \$99 for Zilog's compiler/debugger and \$139 for Rabbit's development board with the Dynamic C environment it's pretty hard to complain about pricing. Yet I wouldn't be surprised to see these semiconductor providers contracting with third-party tool companies to give potential customers more options.

The Chicago show brought the decline of in-circuit emulators into focus for the first time. I saw this in the vendors who were not there, some of whom failed in the last year (Orion and Pentica). It was apparent in how chip vendors, including Rabbit, showed ICE-less debuggers. I think WRI's valuation of EST is a warning to all hardware tool vendors, as EST primarily sells BDM/JTAG hardware. It appears emulators for high-end CPUs will wither in favor of the much cheaper BDMs. Perhaps low-end ICEs will survive, especially as many 8-bit CPUs just don't support serial debug ports. But even there we're seeing change, as in Triscend's 8031 E5 core, which contains a complete serial debugging environment.

Eight- and 16-bit compiler and debugger sales, though, are stronger than ever. In this 32-bit world it ain't easy getting much respect in eight bits, yet Keil (www.keil.com) told me growth in their 8051 and 167 tool sales is phenomenal. IAR (www.iar.com), which provides compilers and debuggers across a wide spectrum of small and large processors, showed their very cool visualSTATE tool that creates code for state machines that you design graphically, on screen.

Other trends

For many years Bruce Powel Douglass of I-Logix (www.ilogix.com) has been an evangelist of UML, sometimes seeming almost alone in his mission to convert us all to modeling freaks. With about 10 talks targeted at UML, including the special guest lecture,

and numerous booths pushing the same concept, the handwriting on the wall seems clear. Large projects need the discipline and capability of UML to avoid disaster.

Virtual hardware design is nothing new; for a decade or more large programmable logic devices let designers create "hardware" designs using tools that closely parallel our firmware-creating compilers. Each year, however, finds more convergence as hardware design requires ever less soldering and more programming. Last year ARC Cores (www.arccores.com) introduced their synthesizable CPU, encouraging customers to modify the instruction set to suit particular needs. Scary stuff. Then Triscend (www.triscend.com) demonstrated that a CPU core surrounded by an FPGA gives engineers customizable virtual peripherals. This show brought Tensilica (www.tensilica.com) to the forefront with their Xtensa 32-bit synthesizable RISC processor.

Tensilica's only product is the intellectual property behind the core and the supporting tools. They don't make chips. At 24,000 gates the CPU is pretty tiny, eating up only 0.7 square millimeters in .18-micron geometry.

If you'd like to build an ASIC with the processor you'll pay them a six-figure licensing fee plus royalties. Alternatively there's a \$64,000 "binary only" deal in case you'd like to dump the core into a very large FPGA. Though the fees keep these processors out of the reach of low-volume products, it's clearly an indicator of things to come.

Another parallel with software, in this case the open source movement, was not at the show. The OpenRISC 1000, soon to be available at www.opencores.org, will be a freely downloadable 32-bit RISC processor. Created by students in Slovenia, it's awfully hard to see what the ultimate success of this effort will be. But pundits said the same thing about Linux not too many years ago.

Finally, I was struck by the number

of outsourcing companies with booths at this year's show. The shortage of engineers, more complex products, and shorter time-to-market pressures have created a land-office business for these hired guns. The startup WebPRN (www.webprn.com) is so virtual they provide neither products nor product development; instead their mission is to electronically connect companies looking for developers with these outsourcing businesses.

Stellcom (www.stellcom.com) told me they've started a new division whose charter is simply to help prospective customers convert raw product ideas into complete business plans and specifications. Apparently the dot-com mania spawns many businesses with little more than an idea and capital, with no idea how to build a business around the concept.

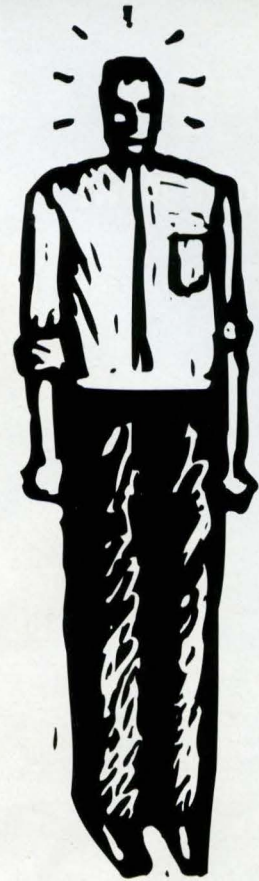
Fun first

As a member of the conference's Advisory Board, I'll admit to an (emotional, not financial) interest in these events. I'm biased. The fact is, though, that the conferences are probably the best educational opportunity the embedded world has to offer. Part of the appeal for me is a chance to meet face to face with others in the industry. But most of all, I go for the fun!

On another note, I recently obtained a Write-Only Memory chip. Signetics created this "product" 25 years ago to celebrate April Fool's Day. Since I don't expect to be creating products that need write once, read never capability, I'm holding a contest to give away the part and its hilarious datasheet. See www.ganssle.com. **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.

DON'T FRET!



Reprints are easy to get!

Full of interesting articles and valuable industry news, **Embedded Systems Programming** has information you'd like to share. Reprints are great for trade shows, marketing kits, and presentations.

Contact Sherry Bloom for details.

tel: 415/905-2701

fax: 415/975-3111

Email:

sbloom@mfi.com

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



P.J. Plauger

Quitting Time

This is my last regular column for *Embedded Systems Programming*. I've written a "State of the Art" column for every regular issue of this magazine since it was started back in the fall of 1988. That's 137 installments, counting this one, plus a handful of articles. I even got the privilege of writing the introductory essay for the premiere issue. It served as a kind of statement of intent that, I am pleased to say, the magazine still pretty much honors. So I feel very much a part of *ESP*, which makes quitting as a regular columnist all the harder.

This is not about "artistic differences" or anything else of that ilk. I thoroughly enjoy working with Lindsey Vereen and Felisa Yang as editors. The former has done a great job of steering the magazine in recent years—it has never been better—and giving me the latitude I need to write what I want. The latter has firmly but gently steered me to the safe side of every deadline, then edited my copy with intelligence and common sense. They even gave me an unsolicited raise recently. A columnist could not ask for a better working environment.

Still, I'm packing it in. I plead exhaustion, the press of other matters, advancing age. Unless you are a fellow writer, it's hard for me to communicate how much effort goes into writing each one of these essays. The time spent at keyboard is but a small number of hours per month. That's a serious enough tax when you're trying to build a small software company, but it's only a part of the total cost. A regular columnist has to come up with a dozen reasonable themes per year.

Those of us who are sufficiently conscientious do some degree of research for each one. In an exploding field like embedded systems programming, personal experience covers an ever smaller part of the whole. Each year I have a bit less energy, yet each year I have to spend a bit more energy making sure that these columns are not complete nonsense by the time you read them.

the late 1970s so that I could write C compilers and other software. Before I knew it, the company had several dozen employees and I was playing president instead of writing software. Or books, or science fiction, or any of the other things I like to write far more than staff memos and employee evaluations.

When the opportunity came along to write regularly for a magazine, I

After 11 years, it's time to get a real job.

Plauger bids farewell to his readers as he hangs up his writing hat.

I knew the day had to come sooner or later—nothing lasts forever. I just didn't know until very recently what it would look like. I have had one publication fold from underneath me. Another changed its focus and name, in a way that I didn't want to follow. But those were decisions made for me, in whole or in part. For what it's worth, I am also quitting as a regular columnist for *The C/C++ Users Journal*, a sister publication to *ESP*. And for much the same reasons, though I will continue as senior editor at *CUJ* for a while longer. Both assignments are coming to an end because I need them to at this point in my life.

You may find it hard to believe, but I feel like I'm quitting my day job. I started writing for magazines a dozen years ago because I wanted to make a career shift. My wife and I started a company called Whitesmiths, Ltd. in

took it. When the opportunity came along soon thereafter to sell Whitesmiths, I took it. When the opportunity came along soon after that to teach for a year in Australia, while I finished a book, I took it. In a few short years, I made the transition from manager to full-time writer, just as I had hoped. Along the way and to this day, writing for various magazines has been a relatively small but steady component of my writing income. It is, in short, my day job.

But a funny thing happened on my way to becoming a full-time writer. The publishing business changed. Books I coauthored in the 1970s are still in print, though their sales have pretty much run down. Books I authored or coauthored in the 1990s are already out of print, though their sales were still nontrivial. The shelf life of books is approaching that of

orange juice, and books are being treated more and more like an equally fungible commodity. I have made less money than I expected as a writer, to be sure, but I've always made enough to live comfortably. What bothers me is the loss of long-term relationships, and commitments, between authors and book publishers. It just ain't as much fun as it used to be.

At the same time, I can't seem to stay out of the software business. Code I wrote just to round out a book or two has ended up earning 10 times what I got from book royalties. Writing HTML manuals for sale over the internet has a happier mix of costs vs. benefits, at least for my temperament. The upshot is that I have drifted inexorably back to writing and licensing software, no longer just to illustrate textbooks. Tana and I started Dinkumware, Ltd. in 1995 as a vehicle for licensing that software to larger customers. We've managed to stay at just a few employees, but the company has steadily grown. I have to accept the reality that Dinkumware has truly become my day job, and that my career as a writer is once again secondary.

Crusader rabbit

There's another reason why I've been reluctant to stop writing these columns. I am a compulsive teacher and crusader. The former I attribute to a mix of genetics and upbringing. My mother taught school, as did both her parents, in the one-room school houses of West Virginia. All three instilled in me from my earliest years a love of learning. I feel an obligation to pass on that love with a passion that sometimes borders on the obsessive.

One of the challenges I take on as

a columnist is explaining some interesting little piece of technology. My target audience is always the practicing programmer, who I imagine cares more about results than detailed mathematical proofs, but who nevertheless wants techniques that have a solid grounding in theory. I've got maybe 3,000 words, in a typical installment, to deliver such a nugget and show how it can be useful. When I succeed, as I have from time to time, the result pays for many months of sweating over deadlines and forcing out the words.

A variant of my obsession for teaching is a strong aversion to misinformation. I care little whether that misinformation stems from simple confusion or from an active effort to deceive. In fact, it can be hard to distinguish the cases. True Believers are often quite zealous because they suffer some degree of confusion about the facts. And zealots often resort to deceptive behavior with the best of intentions. But I believe strongly that accurately informed decisions are the best ones, even when I prefer a different outcome. I champion accurate information both for individuals and groups, whether they're acting as engineers or voters. Muddy thinking is an indulgence that few can truly afford.

Quite a few of the columns I've written over the years have been about such "people" matters. A number of them—perhaps too many—stem from my activities on various programming language standards. That's because developing standards is far more about politics than about technology, as I have learned over the years, sometimes belatedly. And the effect of that political process extends far beyond the little rooms in which the group decisions are hammered out. (See my March 2000 column "Great

Reckoning, Little Room" on p. 141 as a telling example of the power of *not* having a standard.) But the same lessons extend to all sorts of group interactions. You can suffer as much from misinformed decisions if they occur in your technical group design sessions, or in a board meeting at your company.

A programmer of embedded systems thus needs a spectrum of skills, from technical to political. For the past decade or more, I've made it my one-man crusade to teach some of those skills. If I can't articulate how to improve a skill, I at least try to illustrate why it's an important one to have. Sometimes I get feedback from you readers that tells me I've succeeded. Sometimes I get caught out on a factual error or omission. Believe it or not, I welcome both forms of communication. Either is better than silence, which might mean quiet satisfaction but is more easily read (by us authors) as indifference. But I accept the silence as well, knowing how seldom I take time from my busy schedule to communicate with my fellow writers.

I've written around 350 essays over the past dozen years. I intend to write a few more, while I'm still able. I haven't run out of things to say, just the perseverance to write to a tight production schedule. I hope you've enjoyed at least some of my offerings in *Embedded Systems Programming*. And I hope I can please you again as a reader from time to time.

Thanks for reading.

esp

P.J. Plauger is the author of the standard C++ library shipped with Microsoft Visual C++. His latest books are The Draft Standard C++ Library and Programming on Purpose (three volumes), both published by Prentice Hall in Englewood Cliffs, NJ.



It's hard to compete without the right tools.

You can't afford to get left at the gate if your new Intel processor based system is going to finish in the money. But helping you finish in the money is what we're all about.

Ever since 1991, American Arium has been working closely with Intel to create powerful new development tools for each new Intel processor. So whether you've designed an embedded system around a Pentium® processor, or a server using a Merced processor, we can provide you with the most



advanced in-circuit emulators in the business. Also...

- they're available with early silicon
- they're affordable
- we provide in-depth tech support and
- there's an upgrade path

Don't risk finishing back in the pack. Let us help get you to market ahead of schedule and under budget.



14811 Myford Road, 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

www.avocetsystems.com

Every Chip, Every Tool, One Source

QuickTool Finder

Select Chip ▼

PowerPC Families ▲

MPC5xx
MPC8xx
PPC4xx
68K Families
680x0
683xx
Motorola 8/16 Bit
68HC05/08
68HC11
68HC12
68HC16
ColdFire
6801/03
8051 All Derivatives
80C320/520
8xC751/2
C500
DS5000
8080/85
8051XA
80166
80196xx
8086/88
x86 Intel/AMD
80386EX
80186/188xx
Conexant/WDC
6502/65816
C18/19/29
Z180/64180/Z80
Z8/Z8000/Zx80
Microchip PIC
V20/30/40/50
320C20
TLC555

AVOCET
SYSTEMS, INC

(800) 448-8500

sales@avocetsystems.com