EMBEDDED PC MONTHLY SECTION

CIRCUIT CELLAR INK®

THE COMPUTER APPLICATIONS JOURNAL

#88 NOVEMBER 1997

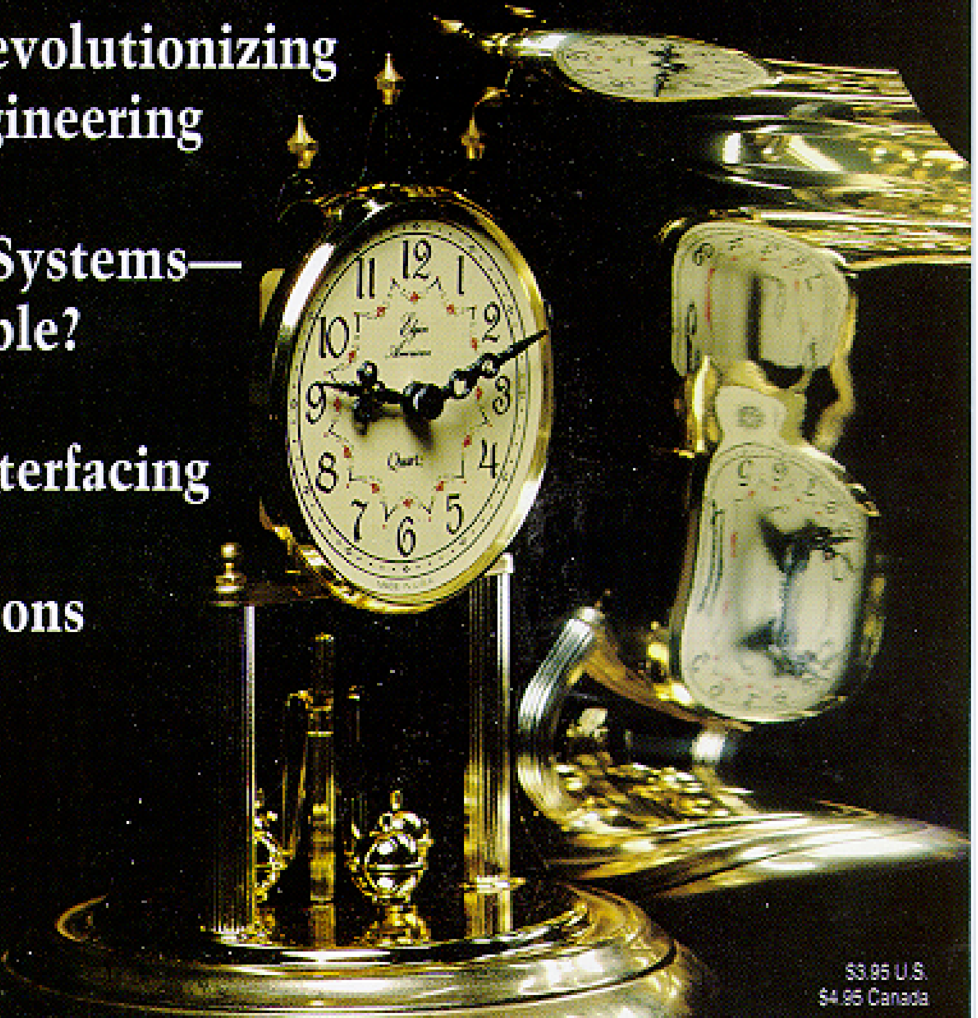
FUZZY LOGIC

Fuzzy Logic—Revolutionizing Automotive Engineering

Self-Correcting Systems—Who's Responsible?

Nonintrusive Interfacing

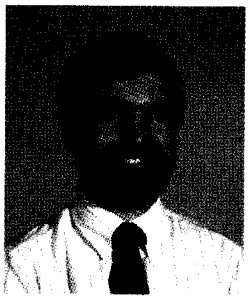
EPC: Pros and Cons of Windows CE



$3.95 U.S.
$4.95 Canada

TASK MANAGER

Stuffed Mentality



I can just picture it. Someone browsing the newsstand comes across this issue of the magazine, sees the theme, and an image of a Beanie Baby shaped like Einstein immediately comes to mind. What else could "Fuzzy Logic" be? (Too bad I didn't mention it in our cover photo meeting.)

How truly unfortunate that such a name was selected for this very serious, deterministic, and useful design philosophy. I've read accounts that the technique is called something very different in Japan, and engineering departments and executives alike embrace the idea as a fine way to improve their products. But, mention fuzzy logic to pointy-haired managers here in the States, and their heads fill with thoughts of toys and liability lawsuits.

While "fuzzy" usually springs forth visions of indecision and no "right" answer, the design technique is based on anything but that. Rather than rely on black-and-white, yes-or-no, and on-or-off events to make decisions, you use ranges, assign weights, and make decisions based on multiple factors. Given identical circumstances, you'll always get identical answers. The usefulness of the answer is far greater, however.

We've covered the basics of fuzzy logic in past issues, so I encourage you to look those up for a ground-up primer. As a step beyond the introductory, though, we start with an article by fuzzy-logic guru Constantin von Altrock on using fuzzy logic in automotive engineering. The Japanese and Europeans have the right idea. Now it's time for the U.S. to take a serious look.

Next, a lot of work has been done on systems that can adapt and modify their behavior based on current conditions and past "learned" behavior. Such systems are often not at all deterministic, making it virtually impossible to test all scenarios in the lab. Who is liable when such a machine causes damage or harm? Rod Taber asks the right questions to start you thinking about it.

We finish our features with a look by Walter Banks at some general techniques for implementing fuzzy logic on decidedly nonfuzzy micro-controllers.

In our columns, Ingo Cyliax finishes his series on developing an affordable MC68030-based teaching platform with a look at the software. Jeff can't keep his hands off the kids' toys as he interfaces an Etch-A-Sketch to a micro. Finally, Tom investigates the latest in USB silicon.

EPC this month focuses on Windows CE and its use in embedded-PC applications. Tom Scott presents the positive side of the argument over whether CE is ready for prime time, while Edward Steinfeld raises some questions and doubts.

In our *EPC* columns, Ingo Cyliax unites PC/104 with the Internet, and Fred replaces the switches-and-lights front panel with a Web browser.

[illegible]

editor@circuitcellar.com

CIRCUIT CELLAR®

THE COMPUTER APPLICATIONS JOURNAL

EDITORIAL DIRECTOR/PUBLISHER
Steve Ciarcia

EDITOR-IN-CHIEF
Ken Davidson

MANAGING EDITOR
Janice Hughes

TECHNICAL EDITOR
Elizabeth Laurençot

ENGINEERING STAFF
Jeff Bachiochi

WEST COAST EDITOR
Tom Cantrell

CONTRIBUTING EDITORS
Rick Lehrbaum
Fred Eady

NEW PRODUCTS EDITOR
Harv Weiner

ART DIRECTOR
KC Zienka

PRODUCTION STAFF
John Gorsky
James Soussounis

ASSOCIATE PUBLISHER
Sue Hodge

CIRCULATION MANAGER
Rose Mansella

CIRCULATION CONSULTANT
John S. Treworgy

BUSINESS MANAGER
Jeannette Walters

ADVERTISING COORDINATOR
Valerie Luster

CIRCUIT CELLAR INK®, THE COMPUTER APPLICATIONS JOURNAL (ISSN 0896-8985) is published monthly by Circuit Cellar Incorporated, 4 Park Street, Suite 20, Vernon, CT 06066 (860) 875-2751. Periodical rates paid at Vernon, CT and additional offices. One-year (12 issues) subscription rate U.S.A. and possessions $21.95, Canada/Mexico $31.95, all other countries $49.95. All subscription orders payable in U.S. funds only, via international postal money order or check drawn on U.S. bank.
VISIT OUR WEB SITE FOR SUBSCRIPTION INFORMATION AT www.circuitcellar.com
Direct subscription orders and subscription related questions to Circuit Cellar INK Subscriptions, P.O. Box 698, Holmes, PA 19043-9613 or call (800) 269-6301.
POSTMASTER: Please send address changes to Circuit Cellar INK, Circulation Dept., P.O. Box 698, Holmes, PA 19043-9613.

Cover photograph Ron Meadows – Meadows Marketing
PRINTED IN THE UNITED STATES

For information on authorized reprints of articles, contact Jeannette Walters (860) 875-2199.

HAJAR ASSOCIATES NATIONAL ADVERTISING REPRESENTATIVES

NORTHEAST & MID-ATLANTIC	MIDWEST & SOUTHEAST	WEST COAST
Barbara (Best) Curley	**Christa Collins**	**Barbara Jones & Shelley Rainey**
(561) 694-2044	(954) 966-3939	(714) 540-3554
Fax: (561) 694-2051	Fax: (954) 985-8457	Fax: (714) 540-7103
b.curley-Hajar@worldnet.att.net	HajarChrista@worldnet.att.net	shelley.hajar@worldnet.att.net

Circuit Cellar BBS—24 Hrs., 2400/9600/14.4k bps, 8 bits, no parity, 1 stop bit, (860) 871-1988. For information, mail to **info@circuitcellar.com**. World Wide Web: **www.circuitcellar.com**

INSIDE ISSUE 88

EMBEDDED PC

www.circuitcellar.com

READER I/O

HDTV AND MONITOR LIMITATIONS

I enjoyed Do-While Jones' "HDTV-The New Digital Direction" (*INK* 86). He took on a large subject!

The section on monitor limitations, however, contains a significant technical error. The author confused the vertical phosphor stripes and slotted shadow mask of Trinitron-style CRTs with the rare and technically challenging Beam Index CRT.

In a Trinitron CRT, there are still three electron beams-one each for R, G, and B. A slotted shadow mask [or aperture grill) is located behind the phosphors. Since the slots extend from the top to bottom on the CRT, Sony uses stabilizer wires to keep the metal strips from vibrating, which causes the horizontal-line artifacts seen on Trinitron monitors. As is typical of all shadow masks, its location behind the screen allows only the "correct" electron beam to strike its associated phosphor stripe.

Maybe you've read about "One Gun" Trinitrons. There's some truth to it, but it's mostly marketing. After the three electron beams are formed by three electron guns, the beams are focused and shaped by a single set of electronic lenses. The larger electrostatic "lenses" provide a better spot quality as the beams are deflected (see www.sonycp.com/_E/Products/Monitors/Technology/Trinitron7.html and ftp.unina.it/pub/electronics/REPAIR/F_mon_repair4.html).

In all shadow-mask CRTs, the vast majority of the electron-beam energy is lost in the shadow mask. The Beam Index CRT achieves high brightness by eliminating the shadow mask and allowing all electron-beam energy to excite the phosphors. As Do-While described, a single beam is time-division multiplexed between R, G, and B. Another stripe synchronizes the multiplexer.

It would be impractical to create a horizontal deflection system with adequate linearity to maintain the RGB multiplexing sync open loop. Some sync implementations use high-conductivity aluminum stripes between the B and R phosphor stripes, so a pulse in anode current indicates the start of a new RGB sequence. Other implementations use a thin stripe of UV-emitting phosphor, with a UV detector incorporated in the back of the CRT bottle to generate the sync signal. Due to their cost and complexity, these systems have only found application in avionic cockpit displays that must be sunlight readable.

Recently, this market has been taken over by AM-LCDs, so Beam Index CRTs may be nothing more than a historical curiosity. There isn't much on the Web about Beam Index CRTs, but I found an abstract titled "Technical challenges of high-voltage power supplies for a modern beam index CRT electronic display" at the <www.optics.org> technical library.

As for ball-park displays, what about the famous Sony Jumbotron (www.sel.sony.com/SEL/bpg) or Mitsubishi's Diamond Vision system (www.amasis.com/diamondvision/technical.html).

Overall, I don't think HDTV requires a breakthrough in monitor technology. I've seen HDTV demos at tradeshows for several years. My desktop monitor produces a 1600 x 1200 pixel image. If the CRT was made in the 16:9 aspect ratio, it could easily display 1920 x 1080.

Consider the price of 17" monitors. They used to be about $1200, but now you can buy a decent one for under $600. Once volume production kicks in, I think prices will be comparable to today's high-end consumer-TV costs.

Tim Godfrey
tgodfreyQdigocean.com

HDTV FREQUENCY FACTS

I enjoyed Do-While Jones' recent article ("HDTV—The New Digital Direction," *INK 86).* But, it left me with some questions. What frequencies and how many channels have been allocated for HDTV broadcasting, and how will the current frequencies be used after the are no longer used for commercial broadcast?

Thanks for an excellent article.

Joe Kihm
jkihm@bellatlantic.net

Existing UHF TV frequencies have been allocated. I don't have the exact number, but I've seen lists of frequencies assigned to various existing broadcast stations. I'm sure there are enough for all existing TV stations.

Everybody's wondering how the current frequencies will be used! The TV stations want to go back to the VHF band because UHF is short range and doesn't penetrate trees very well. The FCC, and perhaps cable and satellite operators, want to keep the broadcast stations in the UHF band to limit broadcast coverage.

The FCC prefers many little stations to just a few superstations because bandwidth is limited, and they want to give it to as many people as possible. Cable and satellite operators, of course, would like to sell signals to people who can't receive them directly. The fewer people who can receive them directly, the bigger the customer base. There could be a bitter political fight about this.

Do- While Jones
do_while@ridgecrest.ca.us

NEW PRODUCT NEWS

Edited by Harv Weiner

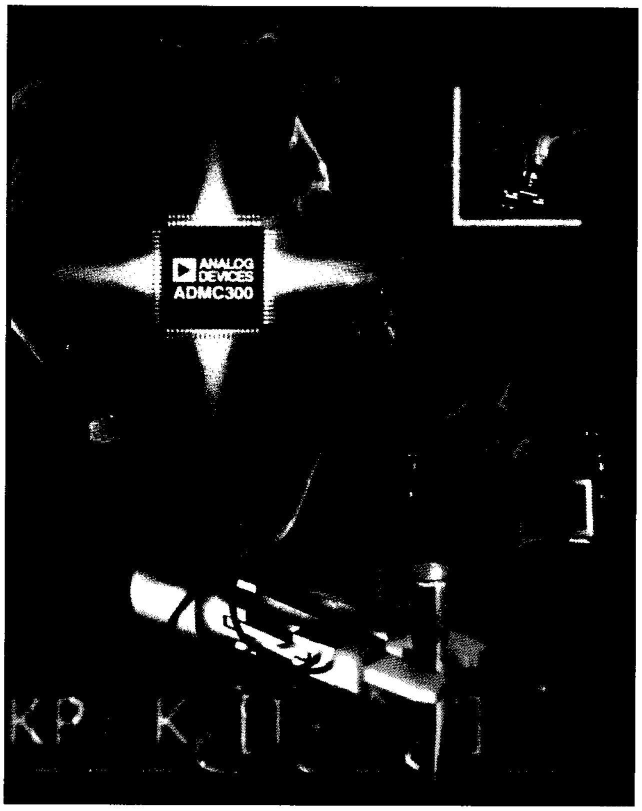
SERVO MOTION CONTROL

The **ADMC300** integrates a 25-MHz fixed-point DSP core and a complete set of peripherals optimized for high-performance motor control. These functions comprise five dedicated analog input channels (including five ADCs), a three-phase 12-bit PWM generator, encoder interface, and an event timer block. The device also features two 8-bit PWM auxiliary timers, expansion capability via two serial ports, and a 12-bit digital I/O port.

Additional features include an internal 4K x 24-bit word program RAM and 1K x 16-bit word data RAM, which can be loaded from an external ROM via the serial port. The ADMC300 also features a 2K x 24-bit word program ROM, including a monitor program tha adds software debugging features through the serial port. The program ROM offers commonly used fixed motor-control algorithm functions and several options for serially loading the device.

The ADMC300 provides smooth torque control over the entire speed range. Control functions required by high-performance servo drives include stall, very fast dynamic response, and accurate position and speed control with excellent regulation. By using sigma-delta conversion technology, it's possible to add high-resolution (11- to 16-bit) A/D conversions to the same silicon as a high-speed DSP core, while reducing the anti-aliasing filter requirements of the system.

The ADMC300 sells for $9.50 in quantities of 100,000.



Analog Devices
One Technology Way • Not-wood, MA 02062-9106
(617) 329-4700 • Fax: (617) 821-4273
www.analog.com **#501**

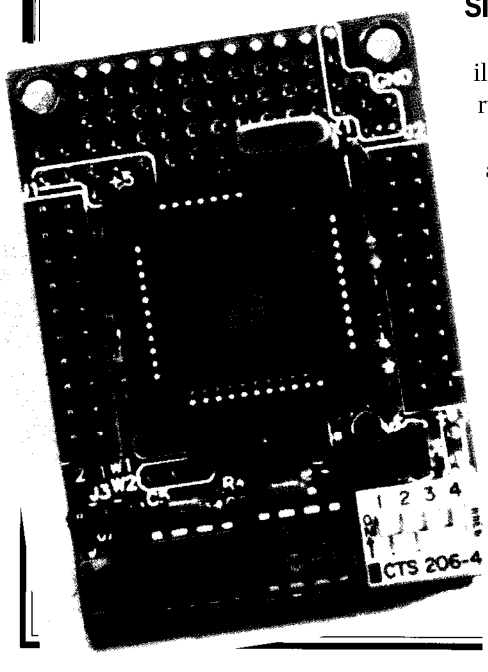
SINGLE-BOARD COMPUTER

The **Tiger Byte-51** is a miniature SBC based on the 8051 microcontroller family. Measuring just 1.65" x 2.5", the board features the 89S8252 microcontroller running at speeds up to 24 MHz.

The 89S8252 has 8 KB of program flash memory for on-chip program storage. A major advantage is the chip's mechanism for in-system programming. An SPI synchronous serial port lets user software be downloaded into flash memory. Any PC with a parallel port can be used to program the chip by interfacing with the board's IO-pin connector.

The board features a 2-KB data EEPROM, RS-232 or RS-485 serial port buffering, a DS 1233 EconoReset device, onboard voltage regulator, prototyping area, and access to all CPU lines. Power requirements are an unregulated source of 9 VDC.

The Tiger Byte-5 1 is priced at $59. A **kit** that includes the Tiger Byte-51, a PC cable, and PC Programmer Software sells for $79.



Allen Systems
2346 Brandon Rd. • Columbus, OH 43221
(614) 488-7122 **#502**

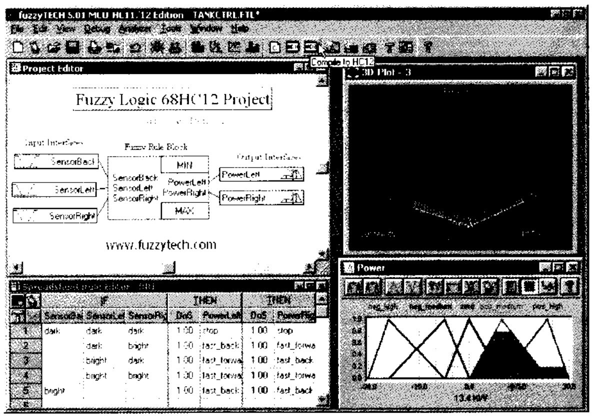
NEW PRODUCT NEWS

FUZZY-LOGIC TOOLS

Motorola and Inform Software have announced new fuzzy-logic development tools for the 8-bit 68HC11 and 16-bit 68HC12 microcontroller families. The ***fuzzy*TECH MCU-HC11/12 Edition** simplifies programming, reduces code size, and enables faster code execution.

The 68HC 12 is the first general-purpose microcontroller family with dedicated fuzzy-logic instructions. Its price/performance ratio enables innovative solutions in diverse applications (e.g., hard-disk design, automotive ABS and ignition systems, as well as routing and cell switching).

Inform provides complete fuzzy-logic design software for all 68HC 11 and 68HC12 derivatives. At the push of a button, *fuzzy*TECH generates the fuzzy-logic system as assembly code for the target microcontroller. On the 68HC 11, *fuzzy*TECH emulates the entire fuzzy-logic algorithm in software, while code generated for the 'HC 12 uses special fuzzy-logic instructions.



	IF			THEN		THEN	
	SensorBa	SensorLef	SensorRig	DoS	PowerLeft	DoS	PowerRig
1	dark	dark	dark	1.00	stop	1.00	stop
2		dark	bright	1.00	fast_back	1.00	fast_forwa
3		bright	dark	1.00	fast_forwa	1.00	fast_back
4		bright	bright	1.00	fast_forwa	1.00	fast_forwa
5	bright			1.00	fast_back	1.00	fast_back

The user can monitor and modify the fuzzy-logic system after implementation in real time on the running microcontroller, requiring only a serial connection between *fuzzy*TECH and the microcontroller. *fuzzy*TECH also supports the 68HC 12's special Background Debug Mode interface for on-the-fly optimization.

*fuzzy*TECH generates faster and more compact fuzzy-logic systems than manual coding can. Complex systems compute in less than 1 ms and require less than 1 KB of ROM on the 68HC11. Due to the 68HC12's special fuzzy-logic functions, implemented fuzzy-logic systems run -15 times faster and are six times more compact than on the 68HC 11.

The lower-cost ***fuzzy*TECH HC11/12 Explorer** offers the same functionality as the MCU-HC11/12 Edition. It has only two inputs and one output and is best suited for basic programming for small fuzzy-logic systems.

*fuzzy*TECH MCU-HC11/12 Edition is available from Inform for $2290, and Motorola sells *fuzzy*TECH HC11/12 Explorer for $199. *fuzzy*TECH HC11/12 Explorer will also be bundled with Motorola's **68HC912B32 Evaluation Board,** available for $99 for a limited time.

Motorola MCU Information
P.O. Box 13026
Austin, TX 78711
(512) 328-2268, x985
www.mcu.motsps.com

#503

Inform Software Corp.
2001 Midwest Rd.
Oak Brook, IL 60521
(630) 268-7550
Fax: (630) 268-7554
fuzzy@informusa.com
www.fuzzytech.com

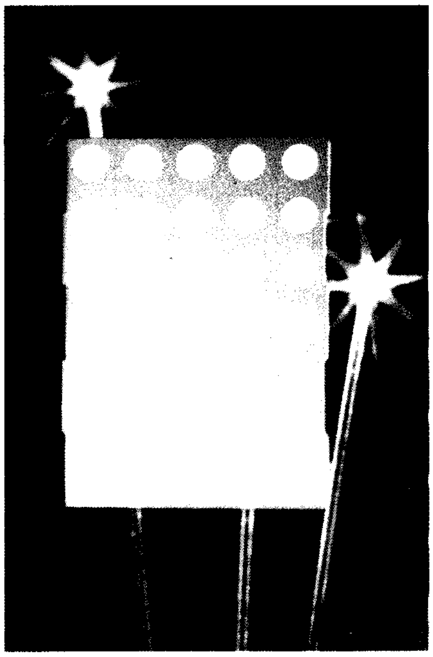
RGB DOT-MATRIX LED

A **5 x 7 dot-matrix display** with full-color RGB output is available from Lumex. The 2.09" high display brings full animation and color capabilities to a wide range of messaging and display applications.

Three LED chips per dot (red, green, and blue) make the display easy to read under a wide range of ambient lighting conditions. Minimum axial light intensity is 300 mcd (blue, 10 mA) with a viewing angle of 100". The LEDs offer extremely quick response times, enabling fast-moving animation. Each color in each dot can be addressed individually or collectively.

Each of the 35 dots is 0.20" in diameter and mounted on 0.30" centers. On the back of the display, two rows of 11 pins are configured on standard 0.10" centers to enable easy integration. Pricing depends on custom configuration.

Lumex, Inc.
290 E. Hellen Rd. • Palatine, IL 60067
(847) 359-2790 • Fax: (847) 359-8904
www.lumex.com #504



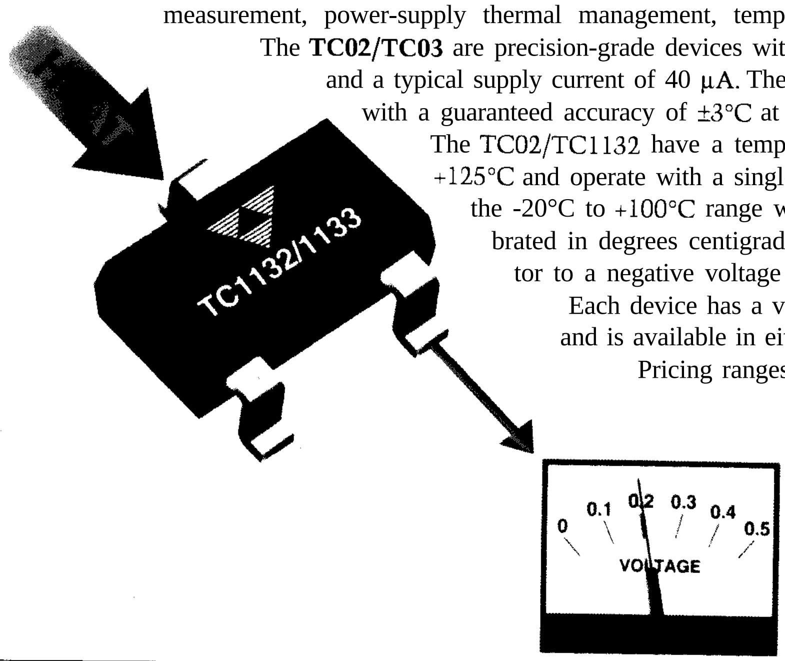
NEW PRODUCT NEWS

TEMPERATURE-TO-VOLTAGE CONVERTERS

TelCom has introduced a series of low-cost solid-state temperature sensors that provide a linearized output voltage directly proportional to measured temperature. Applications include general-purpose temperature measurement, power-supply thermal management, temperature monitors, and controls.

The **TC02/TC03** are precision-grade devices with a guaranteed accuracy of ±2°C at 25°C and a typical supply current of 40 µA. The **TC1132/1133** are consumer-grade devices with a guaranteed accuracy of ±3°C at 25°C and a typical supply current of 60 µA. The TC02/TC1132 have a temperature measurement range of -20°C to +125°C and operate with a single supply. The TC03/TC1133 operate over the -20°C to +100°C range with an output voltage that is directly calibrated in degrees centigrade but requires an external pull-down resistor to a negative voltage source for measurements below 0°C.

Each device has a voltage slope with temperature of 10 mV/°C and is available in either a small TO-92 or SOT23-3 package. Pricing ranges from $0.32 to $0.41 in quantity.

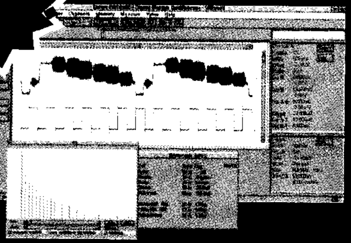


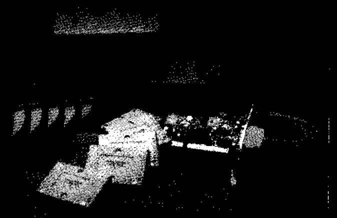
TelCom Semiconductor, Inc.
1300 Terra Bella Ave.
Mountain View, CA 94039-7267
(650) 968-9241
Fax: (650) 967-I 590

#505

Digital Oscilloscopes

Logic Analyzers





- 2 Channel Digital Oscilloscope
- 100 MSa/s max single shot rate
- 32K samples per channel
- Advanced Triggering
- Only 9 oz and 6.3" x 3.75" x 1.25"
- Small, Lightweight, and Portable
- Parallel Port interface to Laptop or PC
- Advanced Math, TV Line Trigger and FFT Spectrum Analyzer options

$499

For $499 you get the model DSO-2102 Scope, Probes, Interface Cable, AC Adapter, and Windows and DOS Software.

- 200 MSa/s max single shot rate
- 2 Oscilloscope channels
- 8 Logic Analyzer channels
- 10 Channels simultaneously
- 125 MHz Single Shot Bandwidth
- up to 128K Samples/Channel
- FFT and Spectrum Analyzer

DSO-28264 (10Ch, 200MS, 64k)	$1999
DSO-28464 (20Ch, 200MS, 64k)	$3799

All prices include Probes and Software

- 40 to 160 channels
- up to 500 MHz
- Variable Threshold
- 8 External Clocks
- 16 Level Triggering
- up to 512K samples/ch

LA4240-32K (200MHz, 40CH)	$1350
LA4280-32K (200MHz, 80CH)	$2000
LA4540-128K (500MHz, 40CH)	$1900
LA4580-128K (500MHz, 80CH)	$2800
LA45160-128K (500MHz, 160CH)	$7000

All prices include Pods and Software
Call for information on our
100 MSa/s Pattern Generator option

Link Instruments (973) 808-8990

369 Passaic Ave • Suite 100 • Fairfield, NJ 07004 • Fax (973) 808-8786

Web: http://www.LinkInstruments.com/cci11 · Email: Sales@LinkInstruments.com

#104

NEW PRODUCT NEWS

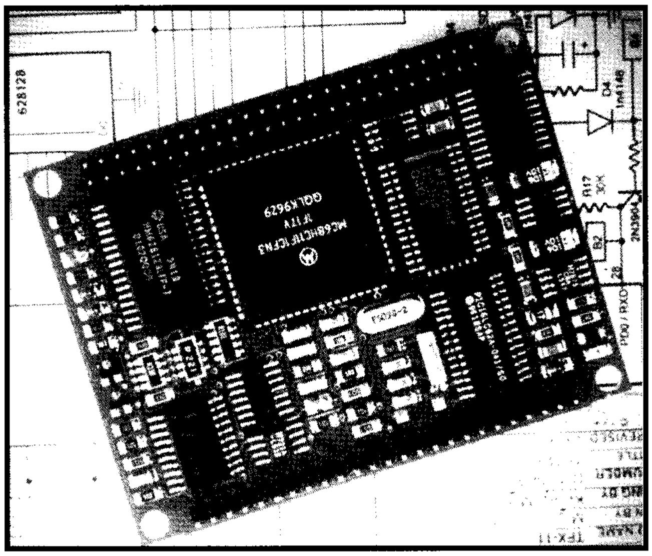
DATA LOGGER/CONTROLLER

The Tattletale Flash express data logger and controllcr engine has been introduced by Onset Computer Corp. The **TFX-11** integrates data-logging and control hardware with TFBASIC software for data collection and control in one product. Its small size, low power requirements, and nonvolatile flash EEPROM make it ideal for embedded portable or remote data-logging and control applications.

The TFX- 11 includes dual processors (Motorola 'HC 11 and Microchip PIC16C62), 512 KB of nonvolatile flash memory, a hardware battery-backed real-time clock, 11 analog input channels with 12-bit resolution and 3200-Hz maximum sampling ratc, eight analog input channels with S-bit resolution, 6400-Hz maximum sampling rate, and **16** digital I/O lincs. It also features 128-KB battery-backed RAM, configured as 64-KB program and 64-KB data storage, and an RS-232 hardware UART.

The 2.4" x 3.2" x 0.5" TFX- 11 accepts a widc range of power-supply voltages from 6 to 30 VDC. Data can be offloaded from flash memory in under 30 s via a standard PC parallel port.

The TFX- 11 sclls for $295.



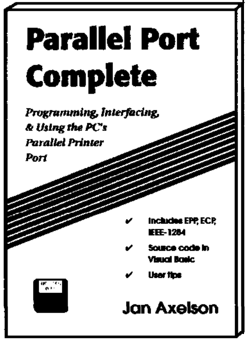
Onset Computer Corp.
P.O. Box 3450 • Pocasset, MA 02559-3450
(508) 563-9000 • Fax: (508) 563-9477
sales@onsetcomp.com • www.onsetcomp.com **#506**

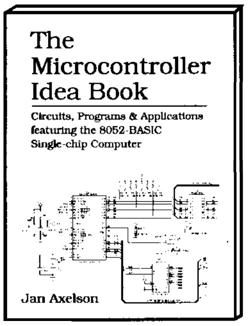
JUMP START YOUR DESIGNS
with these practical guides

NEW! "A focused book that delivers what it promises: detailed technical information on the parallel port." - *Windows Developer's Journal*

"*It's* been a while since I've seen a book as practical as this one." -Nuts *& Volts*

ISBN o-96508 19- I-5
$39.95
343 **pages**
Includes disk


Parallel Port Complete
Programming, Interfacing, & Using the PC's Parallel Printer Port
✓ Includes EPP, ECP, IEEE-1284
✓ Source code in Visual Basic
✓ User tips
Jan Axelson


The Microcontroller Idea Book
Circuits, Programs & Applications featuring the 8052-BASIC Single-chip Computer
Jan Axelson

MICROCONTROLLERS!
"An ideal introduction to low-end embedded design." *EDN*
"The writing is a model of clarity and conciseness."- *PC Magazine*
ISBN 0-9650819-O-7
273 pages. $31.95

Order line: l-800-247-6553

For more information or international orders:
Lakeview Research Phone 608-241-5824
2209 Winnebago St. Fax 608-241-5848
Madison, WI 53704 Email jaxelson@lvr.com

Shipping: $5.00 (U.S. & Canada)

Read sample chapters on the web:
http://www.lvr.com

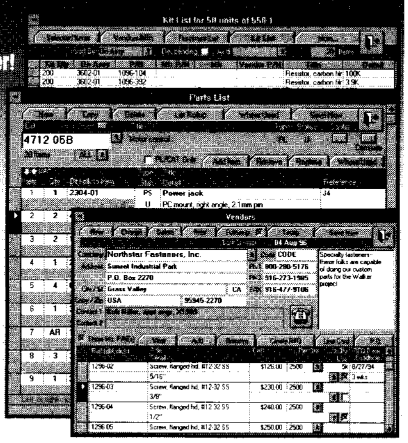
#105

Parts List Manager

For Engineers, Product Designers, Prototypers

Windows-based software helps you stay organized ...And makes the job easier!

Easily create and manage multi-assembly parts lists for products in development ...And after!



Keep track of:
- **Part Specs & Dwgs**
- **Suppliers & Mfrs**
- **Boms and Kit Lists**
- **Product Costs**
- **Engineering Stock**

Parts & Vendors™ Version 2.0
...for independent and departmental projects. Use alone or in a workgroup

Version SE: $99 + s/h EXtended: $299 + s/h

Call 800-280-5176

Trilogy Design
916-273-1985
fax 916-477-9106
P.O. Box 2270, Grass Valley, CA 95945
http://www.trilogydesign.com

Requires 486, 10 meg min. ram. Win 3.x or Win95

FEATURES

FEATURE ARTICLE

Constantin von Altrock

Fuzzy Logic in Automotive Engineering

Automotive engineering is one area where fuzzy logic has made significant inroads. Constantin brings us up to date with the latest developments. He examines fuzzy logic's impact on ABS braking, engine control systems, transmissions, and antiskid steering.

Fuzzy logic is a powerful way to put engineering expertise into products in a short amount of time. It's highly beneficial in automotive engineering, where many system designs involve the experience of design engineers as well as test drivers.

Over the past years, fuzzy logic has become is a common design technology in Japan, Korea, Germany, Sweden, and France. The reasons are manifold.

First, control systems in cars are complex and involve multiple parameters.

Second, the optimization of most systems is based on engineering expertise rather than mathematical models. "Good handling," "Fahrvergniigen," and "riding comfort" are optimization goals that can't be defined mathematically.

Third, automotive engineering is competitive on an international scale. A technology that proves a competitive advantage is soon commonly used.

In this article, I point to case studies in antilock braking systems (ABS), engine control, and automatic gearbox control. I show how superior performance is achieved via fuzzy-logic and neural-fuzzy design techniques. I also discuss development methodologies, tools, and code speed / size requirements.

ABS WITH FUZZY LOGIC

In 1947, Boeing developed the first ABS for airplanes as a mechanical system. Today, ABS is standard equipment on most cars. A microcontroller and electronic sensors measure the speed of every wheel and control the fluid pressure for the brake cylinders.

Mathematical models for a car's braking system exist, but the interaction of the braking system with the road is far too complex to model adequately. Hence, today's ABS contains the engineering experience of years of testing in different roads and climates.

PRODUCING FUZZY ABS

Because fuzzy logic is an efficient way to put engineering knowledge into a technical solution, it's no surprise that many ABS applications are already on the market. Currently, Nissan and Mitsubishi ship cars with fuzzy ABS. Honda, Mazda, Hyundai, BMW, Bosch, Mercedes-Benz, and Peugeot are working on solutions as well.

ABS also benefits from fuzzy logic's high computational efficiency. During a control loop time of 2-S ms, the controllers must fetch all sensor data, preprocess it, compute the ABS algorithm, drive the bypass valves, and conduct the test routines. Any additional function thus has to be computationally efficient.

Most ABS systems use 16-bit controllers, which can compute a medium size fuzzy-logic system in about 0.5 ms, using only about 2-KB ROM space [1]. You can check out a comparison of computing times of fuzzy-logic systems on different microcontrollers [2].

BRAKING BASICS

There are different ways in which fuzzy logic is used in ABS design. The implementation of Nippondenso [3] that I present exhibits an intelligent combination of conventional techniques with fuzzy logic.

Let's first discuss some basics of the braking process. If a wheel rotates exactly as fast as it corresponds to the speed of the car, the wheel has no braking effect at all. If the wheel doesn't rotate at all, it is blocked.

The blocking situation has two disadvantages. First, a car with blocking wheels is hard to steer. Second, the brake effect is not optimal. The point of optimum brake effect is between these two extremes.

The speed difference between the car and the wheel during braking is called "slack." Its definition is:

$$s = \frac{V_{car} - V_{wheel}}{V_{car}}$$

where s is slack (between 0 [no braking] and 1 [blocking]), V_{car} is the car's velocity, and V_{wheel} is the wheel's velocity.

Figure 1 plots the relation between brake effect and slack for different road surfaces. For s = 0, the wheel's speed equals the car's. In the case of s = 1, the wheel blocks completely.

The curves show that the optimum brake effect lies between these two extremes. However, the point of maximum brake effect depends on the type of road. Table 1 lists typical values.

ROAD SURFACE

Conventional ABS controls the bypass valves of the brake fluid so the slack equals a set value. Most manufacturers program this set value to a slack of 0.1, which is a good compromise for all road conditions.

But, as Figure 1 and Table 1 show, this set value is not optimal for every road type. By knowing the road type, the braking effect can be enhanced further.

So, how do you determine what the road type is? Asking the driver to push a button on the dashboard before an emergency brake is not feasible.

Road Condition	Optimum Slack (s)
Dry	0.2
Slippery or Wet	0.12
Ice or Snow	0.05

Table l--The slack value *for* **maximum** *brake effect* **depends** *on the road* **condition.**

Sensors provide one logical alternative. Many companies have evaluated different types of sensors and concluded that sensors which deliver good road-surface identification are too expensive or not sufficiently robust.

However, consider sitting in a 'car equipped with a standard ABS. After driving at a known speed, you could jam on the brake so the ABS starts to work.

Even if you didn't know what the road surface was like, you could make a good guess from the car's reaction. If a driver can estimate the road surface from the car's reaction, fuzzy logic can implement the same ideas into the ABS.

Nippondenso did exactly this. When the ABS first detects the wheel blocking, it starts to control the brake-fluid valves so each wheel rotates with a slack of 0.1.

The fuzzy-logic system then evaluates the reaction of the car to the braking and estimates current road surface. Considering this estimate, the ABS corrects the set value for the slack to achieve the best braking effect in the interval from s = 0.05 to s = 0.2.

The fuzzy-logic system only uses input data stemming from the existing sensors of the ABS. Such input variables are deceleration and speed of the car, deceleration and speed of the wheels, and hydraulic pressure of the brake fluid. These variables indirectly indicate the current operation point of the braking and its behavior over time.

Experiments show that a first prototype with just six fuzzy-logic rules improves performance significantly. On a test track alternating from snowy to wet roads, the fuzzy ABS detected the road-surface changes even during braking.

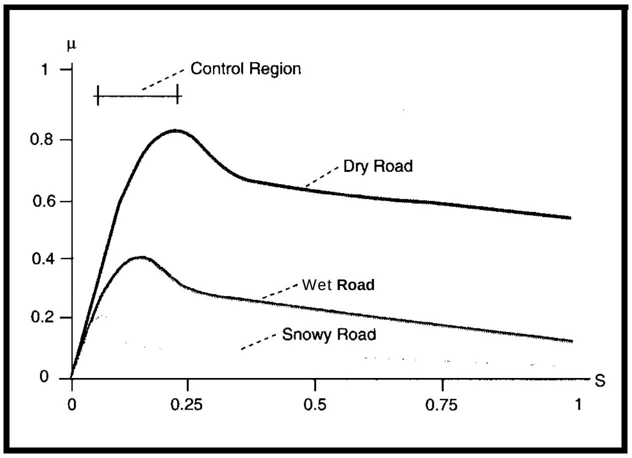


Figure l--This *plot illustrates brake effect over the wheel slacks for dry, wet, and snowy road surfaces. (μ is the friction coefficient or measure of brake effect)*

A FUZZY BRAKE?

Due to the high competition in this area, most manufactur-

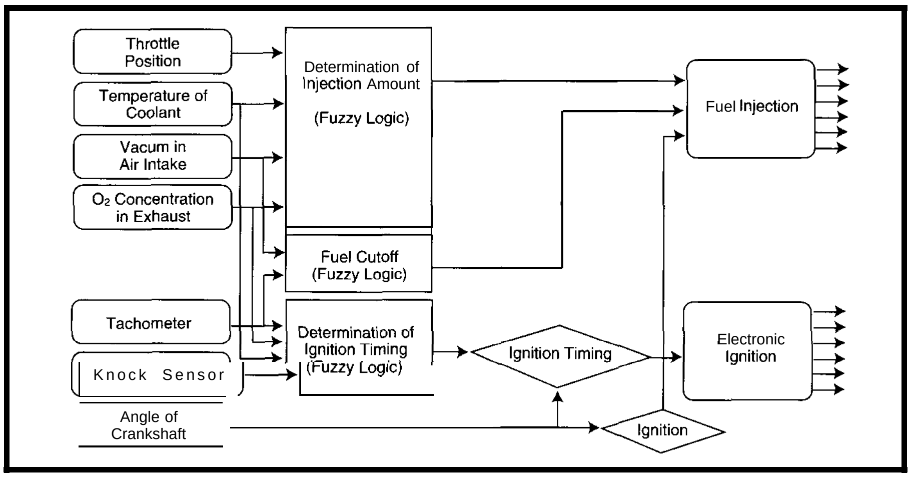


Figure 2—As you can see, the engine controller of NOK Corporation contains three fuzzy-logic modules.

ers are reluctant to publish any details about the technologies they use. The cited application only shows results from an experimental fuzzy-logic system. The details about the final product aren't published.

Also, some car makers (especially in the U.S.) worry about the negative connotation of the word "fuzzy." Since it implies imprecision and inexactness, manufacturers are afraid that drivers may think a fuzzy ABS is inferior. Others are threatened by the possibility of a suit in which a clever lawyer suggests to a layman's jury that a fuzzy-logic ABS is something hazardous.

In Japan, where an appreciation for ambiguity lies in the culture, "fuzzy" doesn't have a negative connotation. By contrast, it's an advantage, as it enables intelligent systems. Hence, companies are proud of its use and promote it in their advertising.

In Germany, on the other hand, the concepts of fuzziness and engineering masterpiece do not fit together well in the public perception. Hence, most manufacturers using fuzzy logic in ABS hide the fact. After all, a fuzzy-logic system is only a segment of assembly code in a microcontroller. Once implemented, who can tell that this code contains fuzzy logic!

VERIFICATION AND STABILITY

When many publications about fuzzy logic appeared for the first time about five years ago, even reputed scientists and professors in the U.S. stated that fuzzy logic shouldn't be used for critical applications. They claimed that it produces inherently instable systems.

This attitude is truly shameful. They demonstrated only that they did not understand what fuzzy logic is about.

A fuzzy-logic system is a time invariant, deterministic, and nonlinear system-nothing fuzzy about that. Such systems are already known and applied in control engineering, and conventional stability theory covers them well [4].

In the case of a fuzzy ABS, stability isn't even an issue. Conventional ABS was considered stable for any slack set value in the interval from 0.05 to 0.2. Hence, a fuzzy-logic road-surface estimator that tunes this value to the optimum cannot make the ABS instable.

Let's move on to see how fuzzy logic easily implements human experience in an embedded engine-control system.

ENGINE CONTROL

The control of car and truck engines is becoming increasingly more complex with more stringent emission standards and constant effort to gain higher fuel efficiency. Twenty years ago, control systems were mechanical (i.e., carburetor, distributor, and breaker contact). Now, microcontroller-based systems control fuel injection and ignition.

Since the control strategy for an engine depends strongly on the current operating point (e.g., revolutions, momentum, etc.), linear control models, (e.g., PID) are not suitable.

On the other hand, no mathematical model describing the complete behavior of an engine exists. Most engine controllers use a look-up table to represent the control strategy. The table is generated from the results of extensive testing and engineer's experience.

The generation of such a look-up table, however, is only suitable for three dimensions (two inputs and one output). Also, the generation and interpretation of such tables is difficult and considered a black art.

Although fuzzy logic can replace these look-up-tables, most manufacturers will not publish any details on a fuzzy-logic engine-control solution.

This secretiveness is due to the fact that the rules of the fuzzy-logic system make the entire engine-control knowl-

Listing 1—The current operation point of the engine is classified by the linguistic variable Situation. Each linguistic term denotes a typical operation point. Because each term is represented as a fuzzy-logic membership function, the linguistic variable can classify all other operation points, too.

```
linguistic variable Situation {

Term 1: Start
Control strategy is that the cold engine runs smooth. Ignition is
timed early, and the mix is fat:

Term 2: Idle
Control ignition timing and fuel injection depending on engine
temperature to ensure that the engine runs smooth;

Term 3: Normal drive, low or medium load
Maximize fuel efficiency by meager mix, watch knocking;

Term 4: Normal drive, high load
Fat mix and early ignition to maximize performance. The only
constraint is the permitted emission maximum:

Term 5: Coasting
Fuel cut-off, depending on situation;

Term 6: Acceleration
Depending on load, fattening of the mix }
```

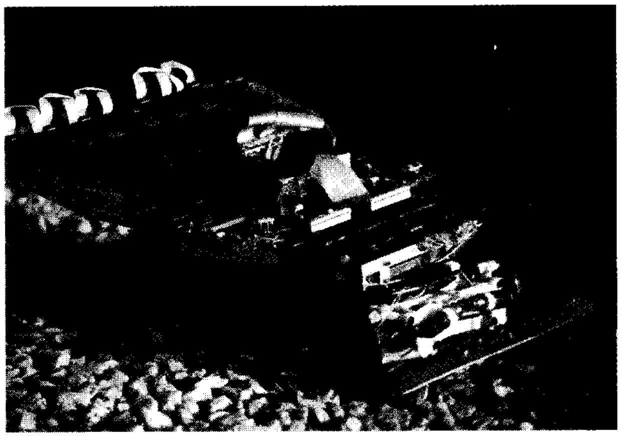


Photo 1 ---This model car *is used in high-speed driving experiments.*

edge of the company completely transparent. They are afraid competitors will learn too much about the solution by disassembling the fuzzy-logic rules.

IDENTIFY DRIVING CONDITION

Nok and Nissan's case study [5] gives the benefits of fuzzy logic in engine control. Figure 2 depicts the components of this engine controller, which contains three fuzzy-logic modules.

The system first notes the engine's operational condition by the linguistic variable Situation. This variable has the linguistic terms in Listing 1.

The determination of Situation is a state estimation of the operation point. Because Situation is a linguistic variable, more than one term can be valid at the same time, so combinations of the operational points can be expressed as defined by the terms.

A possible value of Situation could be (0.8; 0; 1; 0; 0; 0.3). Linguistically, this value represents the driving condition "engine started a short while ago, normal drive condition at medium or low load, slightly accelerating." From this operation-point identification, the individual fuzzy-logic modules control injection, fuel cutoff, and ignition.

Like ABS, engine control needs a very short loop time. Some systems are as fast as 1 ms for an entire control loop. Some manufacturers design the system using fuzzy logic but then translate it into a look-up table for faster processing.

Although a look-up table computes faster, memory requirements may prohibit its use. A look-up table with two inputs and one output, all S-bit resolution, already requires 64 KB of ROM.

Restricting the resolution of the input variables to 6 bits each, the look-up-table still requires 4 KB. A table with three inputs and one output, all inputs 6-bit resolution, requires ¼ MB.

Some engineers implemented a look-up table with a limited resolution and used an interpolation algorithm. However, the interpolation needs about as much computing time as the fuzzy-logic system itself [2].

Another published application of fuzzy logic in engine control is an idle control unit by Ford Motor Corp. [6].

Next, let's check out automatic-transmission control to show how fuzzy-logic systems can adapt their control strategy to drivers.

ADAPTIVE AUTOMATICS

When the first three-speed automatic transmissions appeared on the market about 30 years ago, the engine power of most cars was just sufficient to keep the car in pace with traffic. The necessity of getting maximum momentum from the engine determined the shift points for the gears.

Now, when most car engines can deliver much more power than necessary to keep the car in pace with traffic, automatic transmission systems have up to five speeds, and fuel efficiency has become an important issue, controlling shift points is much more complex.

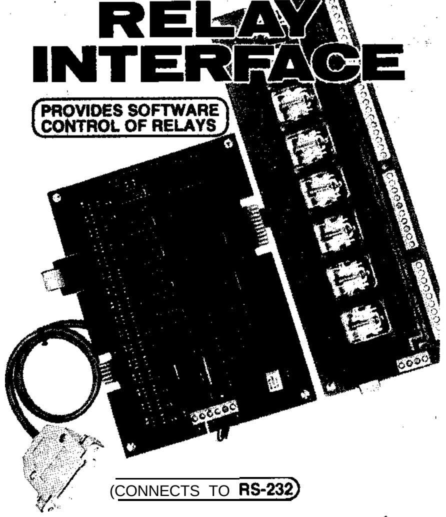
Five speeds and higher engine power give the automatic-transmission system a much higher degree of freedom. Driving at 35 MPH, a three-speed automatic transmission has to select second gear. A five-speed transmission with a powerful engine can select second gear for maximum acceleration, third gear for normal driving condition, and fourth gear for minimal acceleration.

ACCELERATE OR SAVE FUEL

Unfortunately, the goal for the control strategy is in a dilemma. For maximum fuel efficiency, you want to select the next higher gear as early as possible.

RELAY INTERFACE

PROVIDES SOFTWARE CONTROL OF RELAYS

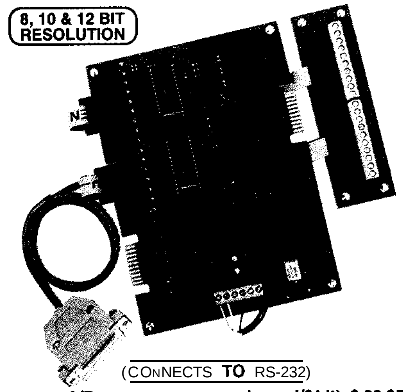


(CONNECTS TO **RS-232**)

AR-16 RELAY INTERFACE (16 channel) . . . . **$** 69.96
Two 8 channel (TTL level) outputs are provided for connection to relay cards or other devices (expandable to 128 **relays** using EX-16 expansion cards). A variety of relays cards and relays are stocked. Call for more info.
AR-2 RELAY INTERFACE (2 relays, 10 **amp)....$ 44.95**
RD-6 REED RELAY CARD (6 relays, 10 **VA)......$** 49.95
RH-6 RELAY CARD (10 amp **SPDT,** 277 **VAC)...$ 69.95**

ANALOG **TO** DIGITAL

8, 10 & 12 BIT RESOLUTION



(CONNECTS **TO** RS-232)

ADC-16 **A/D** CONVERTER' (16 **channel/8 bit)..$ 99.95**
ADC-8G AD CONVERTER* (6 channel/IO **bit).$124.90**
Input voltage, amperage, pressure. energy usage, light, joysticks and a wide variety of other types of analog signals. RS-422/RS-485 available (lengths to 4,000'). Call for info on other A/D configurations and 12 bit converters (terminal block and cable sold separately). Includes Data Acquisition software for Windows 95 or 3.1
ADC-8E TEMPERATURE INTERFACE' (6 **ch)..$ 139.95**
includes term. block & 8 temp. sensors (-40' to 146' F).
STA-6 DIGITAL INTERFACE' (6 **channel).........$** 99.95
Input on/off status of relays, **switches,** HVAC equipment. security devices, keypads, and other devices.
PS-4 **PORT** SELECTOR (4 channels **RS-422)....$ 79.95**
Converts an RS-232 port into 4 selectable RS-422 ports.
CO-422 (RS-232 to RS-422 converter) . . . . $ 39.96

● EXPANDABLE...expand your interface to control and monitor up to 512 relays, up to 576 digital inputs, up to 126 analog inputs or up to 128 temperature inputs using the PS-4, EX-16, ST-32 & AD-16 expansion cards.

- FULL TECHNICAL **SUPPORT...provided** over the telephone by our staff. Technical reference & disk including test software & programming examples in QuickBasic, GW Basic, Visual Basic. Visual C++, Turbo C, Assembly and others are provided.
- HIGH **RELIABILITY...engineered** for continuous 24 hour industrial applications with 10 years of proven performance in the energy management field.
- CONNECTS TO RS-232, RS-422 **or RS-485...use** with IBM and compatibles, Mac and most computers. All standard baud rates and protocols (50 to 19,200 baud).

FREE INFORMATION **PACKET...use** our 800 number. Fax or E-mail to order, or visit our Internet on-line catalog.
URL: **http://www.eeci.com**
Technical Support (614) 464-4470

24 HOUR ORDER LINE (800) 842-7714
Visa-Mastercard-American Express-COD

Internet E-mail: **eeci1@ibm.net**
International & Domestic FAX: (614) 464-9666
Use for information, technical support 8 orders.

ELECTRONIC ENERGY CONTROL, **INC.**
360 South Fifth Street, Suite 604
Columbus. Ohio 432155491

#107

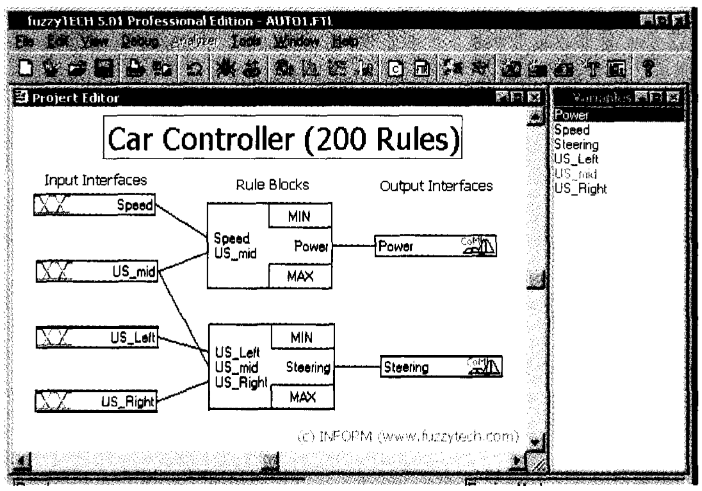


Photo 2—The first version of the fuzzy-logic controller has 200 rules in two rule blocks. The four left boxes indicate input interfaces for sensors, the two right boxes indicate output interfaces for the actuators, and the two large boxes in the middle represent fuzzy-logic rule blocks.

- understand whether the driver wants economical or sporty performance
- avoid unnecessary overdrive, if switching to the next lower gear does not deliver more acceleration

But for maximum performance, you switch to the next higher gear later.

If you have a standard shift, you choose your strategy depending on the traffic condition. An automatic gearbox has no understanding of the traffic condition or the driver's wishes.

However, intelligent control techniques can enhance automatic transmissions as it is based on experience and engineering knowledge rather than mathematical models. Fuzzy logic therefore proves to efficiently implement the technology.

In 199 1, Nissan introduced fuzzy-logic-controlled automatic five-speed transmission systems [7, 8]. Honda followed in 1992 [9], and GM/Saturn in 1993.

The job for the fuzzy-logic system in these applications is similar:

- avoid "nervous" shifting back and forth on winding or hilly roads

Figure 3 shows a typical situation on a fast, winding road. With a standard shift, you'd leave it in fourth gear, but a five-speed automatic transmission switches between the fourth and fifth gears depending on the speed of the car.

The fuzzy-logic transmission controller evaluates more than just the current speed of the car. It also analyzes how the driver accelerates and brakes.

To detect a winding road, the fuzzy-logic controller looks at the number of accelerator pedal changes within a period. Figure 4 shows the definition of the linguistic variable Accelerator pedal changes. The variance of the accelerator pedal changes is input to the fuzzy-logic controller.

Some of the rules estimating the road and driving conditions from these input variables are:

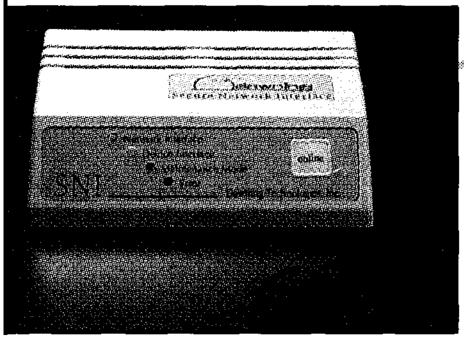
- many pedal changes within a period indicate a fast and winding road
- few pedal changes within a period indicate a freeway
- many pedal changes within a period and a high variance of pedal changes indicate a slow and winding road
- medium variance of pedal changes indicates a fast and winding road
- low variance of the pedal changes indicates a freeway

The interesting part of this application is that the fuzzy-logic controller uses the driver as the sensor. It interprets the driver's reaction to the road and driving conditions and adapts the car's performance accordingly.

This behavior could be used to define an intelligent control system. The technical system tries to understand whether the human is satisfied with its performance and adapts itself to suit the needs of the human using it.

"INTELLIGENT" TRANSMISSIONS

Another example of an automatic transmission system currently under development in Germany illustrates this possibility even better.



Plug Into Tomorrow's Networks, Today.

Why limit the intelligence of serial device connections to your Ethernet network? The new Secure Network Interface from Dawning can provide capabilities far beyond straight-in connections or terminal servers at comparable prices. With the SNI you can have the features of a mini-Web server at each device:

http://www.dawning.com/snipage.htm For more Info Visit Our Website

- TCP/IP stack supports FTP, SMTP, Telnet, and HTTP protocols.
- Acts as a mini-Web server with a discrete IP address.
- Definable access and delivery rules control interaction with multiple users and hosts.
- Allows remote access using standard Web browser software.
- Encryption and authentication options provide security (pending).
- Custom programming available.
- User-programmable for any serial device.
- Price: $595. (10Base2 or 10BaseT)

dawning®
Smart Connections

Call us at: l-800-332-0499 • **716•223-6006**
Fax: 716-223-8615 • **http://www.dawning.com**
e-mail: sales@dawning.com

#108

If drivers want to accelerate, but aren't satisfied with their cars' responce, they unconsciously push the pedal down even more wintin 1-1.5s. This scenario represents the subconscious reaction of most drivers to unsatisfactory acceleration.

Most drivers don't even realize that they like the car to accelerate faster. If an automatic transmission system is capable of detecting this, it can move the shift points higher to achieve more acceleration.

The opposite case is similar. If the automatic transmission detects that the driver accelerates carefully and takes the foot off the accelerator long before red lights, chances are that the driver wants high fuel efficiency.

WHY FUZZY LOGIC?

The question remains, why do you need fuzzy logic to implement these intelligent functions? My answer: while you can use other techniques to implement these control strategies, fuzzy logic is likely to be the most efficient.

Intelligent control strategies are built on experience and experiments rather than from mathematical models. Hence, a linguistic formulation is more efficient.

These strategies mostly involve a large number of inputs. Most of the inputs are only relevant for some specific condition. Using fuzzy logic, these inputs are only considered in the relevant rules, keeping even complex control-system designs transparent.

Another consideration is that intelligent control strategies implemented in mass-market products have to be implemented cost efficiently. In comparison to conventional solutions, fuzzy logic is often much more computational and code-space efficient.

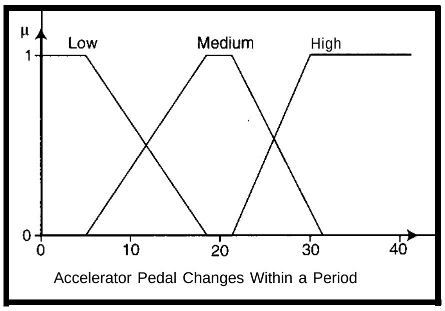


Figure 4—*Here, driving condition is classified using a linguistic variable. The variable linguistically interprets the amplitude of accelerator pedal changes within a certain period.*

Let's look now at how fuzzy logic enables the design of new functionality for automatic steering control.

ANTISKID STEERING

Active stability control systems in cars have a long history. First, ABS improved braking performance by reducing the amount of brake force applied by the driver to what the road surface can take. This system avoids skidding and sliding, resulting in shorter braking distances.

Second, traction-control systems, which do essentially the same thing as ABS, improve acceleration. By reducing engine power applied to the wheels to what the road can take, a traction-control system maximizes acceleration and minimizes tire wear.

After skid-controlled braking and acceleration, the next logical step is skid-controlled steering. An antiskid steering system (ASS) reduces the steering angle applied by the driver through the steering wheel to the amount the road can take. It optimizes the steering action and avoids sliding since a sliding car is very difficult to restabilize, especially for drivers not accustomed to such situations.

Though an ASS makes a lot of sense from a technical point of view, such a system is harder to market. For an ABS, you can prove that it never performs worse than a traditional braking system. For an ASS, this is hard to prove.

Also, it may be difficult to sell cars that "take over the steering" in emergency situations. Even ABS faced a long period of rejection by customers because they felt uneasy about a system "inhibiting" their brake action.

For these reasons, it may take a long time before ASS will be implemented in a production car. All results shown in this section stem from the research of a German car manufacturer [10]. Because this system is one of the most complex fuzzy-logic embedded systems ever developed, it effectively demonstrates the potential of the technology.

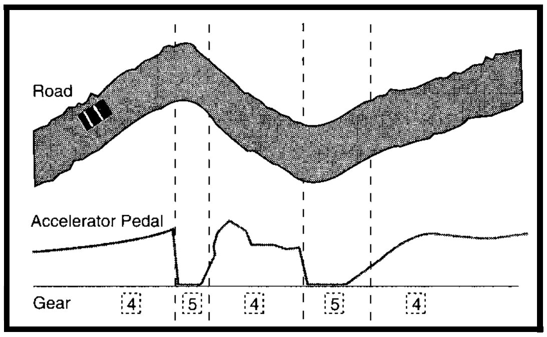


Figure 3-A *five-speed automatic transmission with fixed shift points always switches between fourth and fifth gear on a winding road. A driver with a shift gearbox would leave it* **in fourth** *gear.*

THE TEST VEHICLE

Real experiments were made on a modified Audi sedan and the 20" model car shown in Photo 1. In the following discussion, I only present the results derived from the model-car experiments.

A midmounted I-hp electric motor powers the car, rendering the power-to-weight ratio of a race car. This setup enables the researchers to perform skidding and sliding experiments in extreme situations at high speeds.

On dry surface, the car reaches a velocity of 20 MPH in 3.5 s, with top speeds up to 50 MPH. The speed for most experiments ranges from 20 to 30 MPH. Each wheel features individual suspension and has a separate shock absorber. The car has disk brakes and a lockable differential [10].

The car's controller uses the motherboard of a notebook PC connected to an interface board driving the actuators and sensors. Actuators are power steering servo, disk brake servo, and pulse-width modulated motor control.

Sensors are three ultrasound (US) distance sensors for tracking guidance (see Figure 5) and infrared (IR) reflex sensors in each wheel for speed. The control loop time-from reading in sensor signals to setting the values for the actuators-is 10 ms.

To measure the dynamic state of the car (e.g., skidding and sliding), IR sensors measure the individual speed of all four wheels. Evaluating wheel-speed differences, the fuzzy-logic system interprets the current situation.

Three fixed-mounted US sensors measure the distance to the next obstacle to the front, left, and right. This setup permits autonomous operation of the car. Low-cost sensors were intentionally used in this study-rather than CCD cameras and image-recognition techniques-to show that expensive sensors can be replaced by a fuzzy-logic control strategy.

Figure 6 shows a sample experiment involving the model car. The obstacle is placed right after the curve, so the US sensors of the car detect the obstacle too late.

To not hit the obstacle, the car has to decide for a very rapid turn. To optimize the steering effect, the anti-skid controller must reduce the desired steering angle to the maximum the road can take, avoiding both sliding and hitting the obstacle.

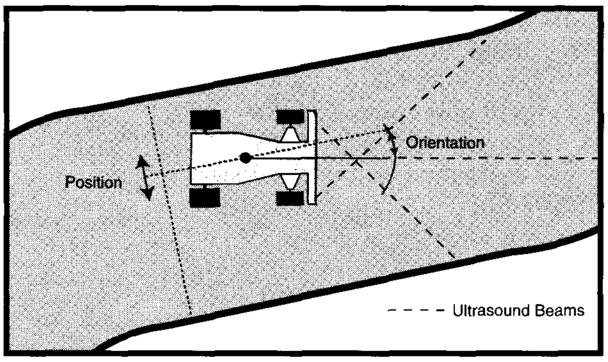


Figure 5--*Three ultrasound sensors guide the car in the track.*

MODEL BASED VS. FUZZY LOGIC

In theory, you can build a mechanical model for a car and derive a mathematical model with differential equations to implement a model-based controller. In reality, the complexity of this approach is overwhelming, and the resulting controller would be difficult to tune.

Here is the point for fuzzy logic: race-car drivers can control a car in extreme situations very well without solving differential equations. Hence, there must be an alternative way for anti-skid steering control.

This alternative way is to represent the driving strategy in engineering heuristics. Although there are multiple ways of expressing engineering heuristics, fuzzy logic has proven very effective for the following reasons.

You can often formulate engineering heuristics in if-then causalities. In contrast to other methods of expressing if-then causalities (e.g., expert systems), the computation in a fuzzy-logic system is quantitative rather than symbolic.

In a fuzzy-logic system, you use a few rules to express general situations, and then the fuzzy-logic algorithm deduces decisions for the real situations that occur. A conventional expert system needs a rule for each possible situation.

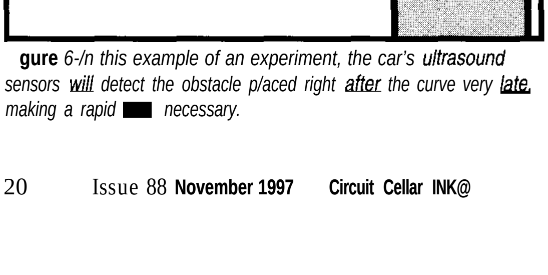
In a fuzzy-logic system, every element is self-explanatory. Linguistic variables are close to the human representation of continuous concepts. Fuzzy if-then rules combine these concepts much the same way humans do.

Fuzzy logic is nonlinear and multiparametric by nature. So, it can better cope with complex control problems that are also nonlinear and involve multiple parameters.

And finally, fuzzy logic can be efficiently implemented in embedded control applications. Even on a standard microcontroller, a fuzzy-logic system can outperform a comparable conventional solution both by code size and computing speed.

DESIGN AND IMPLEMENTATION

Photo 2 shows the first version of a fuzzy-logic controller for the car. The objective for this controller was autonomous guidance of the car in the track at slow speed, where no skidding and sliding yet occurs.


gure *6-/n this example of an experiment, the car's ultrasound sensors will detect the obstacle p/aced right after the curve very late, making a rapid* ■ *necessary.*

In Photo 2, the lower rule block uses the distances measured by the three US sensors to determine the steering angle. The upper rule block implements a simple speed control by using the distance to the next obstacle measured by the front US sensor and the speed of one front wheel only.

Due to the slow speeds, no skidding or sliding occurs. All wheel speeds are the same.

This first version of the fuzzy-logic controller contained about 200 rules and took only a few hours to implement.

The second version of the fuzzy-logic controller implements a more complex fuzzy system for dynamic stability control. It includes antilock braking as well as traction and antiskid steering control **(see** Photo 3).

This 600-rule fuzzy-logic controller has two stages of the fuzzy inference. The first stage, represented by the three left rule blocks, estimates the state variables of the car's dynamic situation from sensor data. The two lower rule blocks estimate skidding and sliding states from speed sensor signals, while the upper rule block estimates the car's position and orientation in the test track.

Note that the output of the left three rule blocks-the state variable estimation-is linguistic rather than numerical. An estimated state of the car can therefore be "the position is rather left, while the orientation is strongly to the right, and the car skids over the left front wheel."

The second stage, represented by the three right rule blocks, uses these estimations as inputs to determine the best control action for that driving situation. The upper rule block determines the steering angle, the middle one the engine power to be applied, and the lower one the brake force.

Such a two-stage control strategy is similar to the human behavior. It first analyzes the situation and then determines the action. It also allows for efficient optimization, since the total of 600 rule structures in six rule blocks can be designed and optimized independently.

The first version of the controller was only able to guide the car on autonomous cruise (see Photo 2). The second version also succeeded to dynamically stabilize the car's cruise via ABS, traction control, and ASS **(see** Photo 3).

However, this version required a much longer design time before the results were completely satisfactory. The second version also uses advanced fuzzy-logic technologies such as FAM inference [11] and the Gamma aggregational operator [12].

ONLINE DEVELOPMENT

The development of the fuzzy-logic system used the *fuzzy*TECH software product [11]. Given the graphical definition of the system structure (cf. Photos 2 and 3), the linguistic variables, and the rule bases, *fuzzy*TECH's compiler generates the system as C or assembly code.

This code was implemented on a PC board mounted on the car. Figure 7 shows how the running fuzzy-logic system was modified on-the-fly for optimization.

The fuzzy-logic code is separated into two segments. One contains all static parts-code that doesn't need to be modified for system alterations. The other segment contains all dynamic parts-the code containing membership functions of the linguistic variables, the inference structure, and the rules.

The dynamic segment is doubled, with only one of the segments active at the same time. In this situation, the parser, linked to the development PC via a communications manager, can modify the inactive code segment.

This technique enables modifications on the running system without halting or compiling. At the same time, the entire inference flow inside the fuzzy-logic controller is graphically visualized on the PC, since the communications manager also transfers all real-time data.

The ASS example demonstrates the applicability of fuzzy-logic technologies for a complex control problem found in the automotive industry. The system was a straightforward design based on experimental experience without a mathematical model of the process.

During optimization, the control strategy was easy to optimize due to the linguistic representation inherent in the fuzzy-logic system. Tests and verification were expedited due to the controller's transparency. And, the poor computational performance of early fuzzy-logic software solutions was overcome via a new generation of software-implementation tools [2,13].

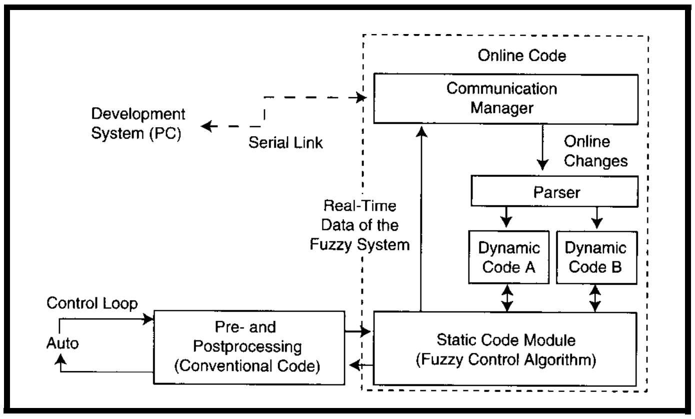


Figure 7—*The fuzzyTECH* **Online Edition** *features both visualization of running system and modifications on-the-fly.*

In the remainder of this article, I provide an overview of other automotive applications where fuzzy logic has been used successfully.

HVAC IN CARS

Fuzzy-logic design technologies are well-established in heating and air conditioning of residences and offices. Hence, it's no surprise that many car manufacturers also use fuzzy logic in their HVAC-system designs.

While most car manufacturers work on these systems, very few publish their efforts. The control approach in general and hence the use of fuzzy logic in the design differ significantly for each manufacturer. In this section, I use an example that Ford Motor Company developed in the U.S. [14].

The fundamental goal of HVAC in cars is to make vehicle occupants comfortable. Human comfort, however, is a complex reaction, involving physical, biological, and psychological responses to the given conditions. The performance criterion-comfort-is not some well-defined mathematical formula but a sometimes inconsistent and empirically determined goal.

Typical HVAC-system sensors measure cabin temperature, ambient temperature, sun heating load, humidity, and other factors. Typical actuators are variable speed blowers, means for varying air temperature, ducting, and doors to control the direction of airflow, and the ratio of fresh to recirculated air. This multiple-input, multiple-output control problem doesn't fall into any convenient category of traditional control theory.

Photo 4 shows the control surface of a part of a HVAC system-the blower-speed control. Blower speed depends on two input variables-the temperature error (i.e., the in-car temperatur'e minus the set-point temperature) and the engine-coolant temperature.

Photo 5 shows the rule base. If the temperature error is zero, low blower speed is desired. If its too hot inside (i.e., positive temperature error), high blower speed is needed to cool the cabin.

If the error is negative, indicating that it's too cold inside, and the engine is cold, little blower speed is needed for defrost. If the error is negative but the engine is warm, high blower speed is needed to heat up the cabin.

OTHER APPLICATIONS

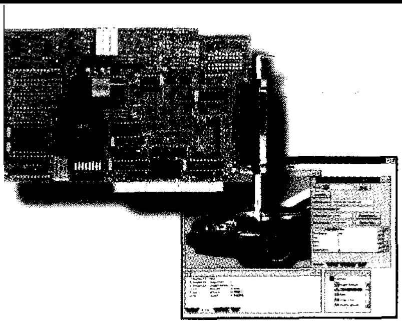
This section briefly introduces some other examples of fuzzy-logic control in automotive engineering. For details, refer to the papers cited.

Peugeot Citroen of France developed a fuzzy-logic system for an intelligent cruise control [15]. The system combines multiple functions for autonomous intelligent cruise control (i.e., following another vehicle, stop and go procedures, and emergency stop]. The system uses three fuzzy-logic blocks with four inputs, one output, and 30 rules each.

Optimization and verification of the rule base used a Citroen XM sedan with automatic gearbox and ABS as test vehicle. The fuzzy-logic controller runs on an 8-bit microcontroller.

The car uses a speed sensor and a single-beam telemeter for the distance to the next car. The actuators command brake pressure and accelerator. Tests show the fuzzy-logic controller can handle the cruise under all the tested conditions.

Future regulations in the European Community (EC) require a speed con-



Lowest Cost

Data Acquisition

ADAC's **new Value-line has uncompromising design features and high quality components at prices below the low cost guys!**

Just check out the specs:

Lowest Cost
5500MF **$195**
8 channels 12-bit A/D,
16 digital I/O, Counter/Timer

High Speed
5508LC **$245**
8 channels 12-bit A/b,
100KHz, DMA

Multi-Function DMA
5516DMA **$295**
16 channels 12-bit A/D,
DMA, 16 digital I/O

High Resolution
5500HR **$595**
16 channels 16-bit A/D,
DMA, 8 digital I/O

learn more:

voice	*800-648-6589*
fax	**617-938-6553**
web	**www.adac.com**
email	**info@adac.com**

ADAC
American Data Acquisition Corporation
70 Tower Office Park, **Woburn,** MA 01801 **USA**

#110

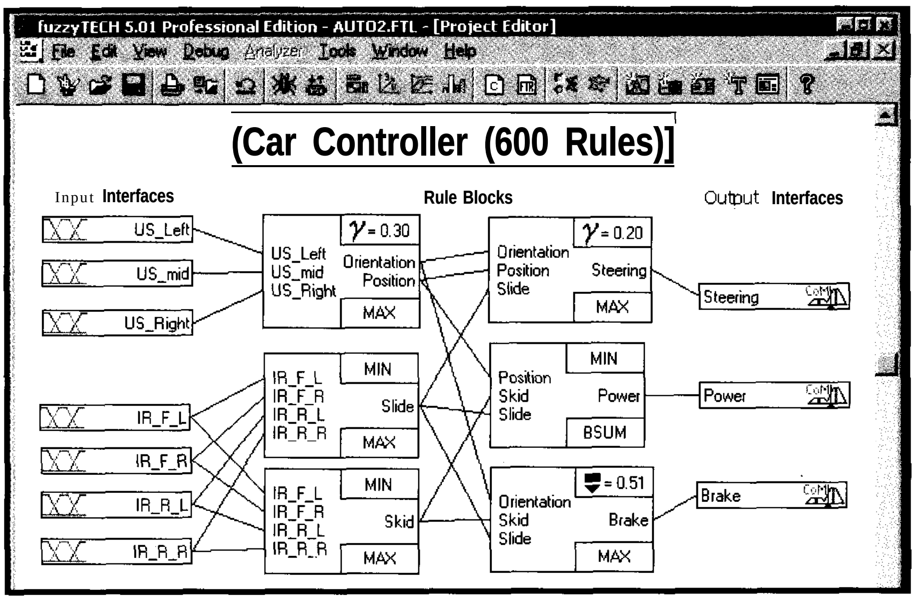


Photo 3—*Here's the second version of the fuzzy-logic controller. The controller uses advanced fuzzy-logic design technologies and contains a total of 600 rules.*

trol for limiting truck speeds on roads in Europe. Today's speed limiters use adaptive ND-type controller. However, the resulting truck behavior is unsatisfactory, compared to an experienced driver.

Therefore, a number of recent designs use fuzzy-logic control to achieve robust performance, even under the strong load changes of commercial trucks [16, 17].

A paper from Ford Electronics describes the design of a traction-control system for a radio-controlled model car [18]. The fact that Ford publishes model-car applications is symptomatic of the fear of many automotive manufacturers to admit that they use fuzzy logic as a design technique for "real" cars.

THE FUTURE IS FUZZY

Over the past five years, fuzzy logic has significantly influenced the design of automotive control systems. Since using fuzzy logic involves a paradigm shift in the design of a control system, five years is a short period. The move from analog to digital solutions has taken a much longer time.

The key reason for fuzzy logic's success in automotive engineering lies in the implications of its paradigm shift. Previously, engineers spent much time creating mathematical models of mechanical systems. More time went to real-world road tests that tuned the fudge factors of the control algorithms.

If they succeeded, they ended up with a control algorithm of mathemati-

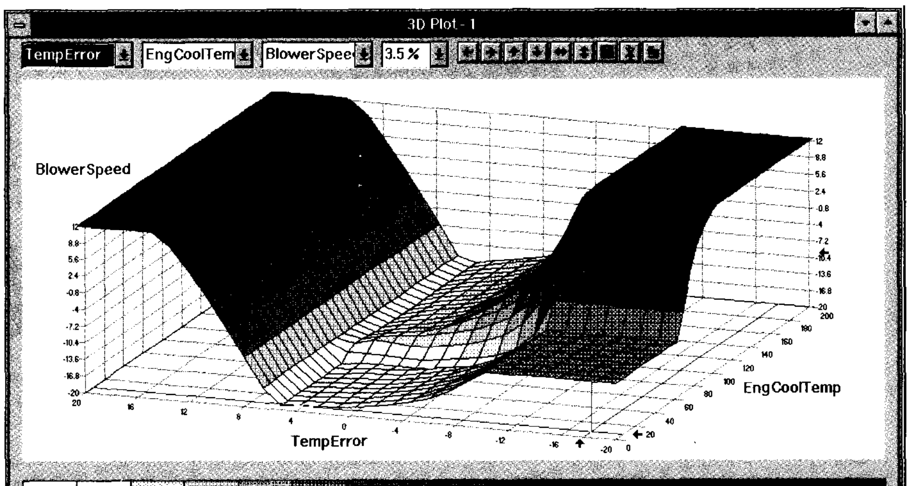
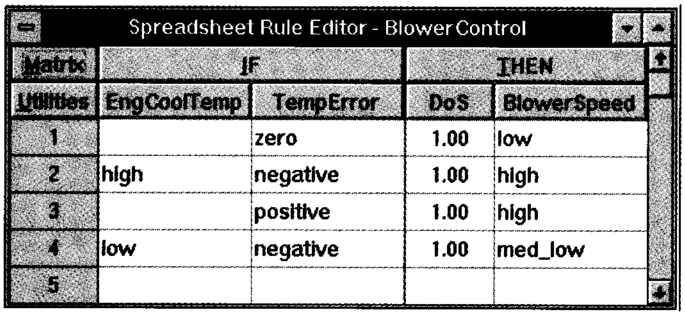


Photo 4—*This graph depicts the control surface of the air conditioner's blower-speed control. Blower speed is determined by temperature error and engine-coolant temperature.*



Spreadsheet Rule Editor - Blower Control				
Matrix	IF		THEN	
Utilities	EngCoolTemp	TempError	DoS	BlowerSpeed
1		zero	1.00	low
2	high	negative	1.00	high
3		positive	1.00	high
4	low	negative	1.00	med_low
5				

Photo 5—*The rule base for* **blower-speed** *control shows how the two* **variables** *of engine* **temperature and** *temperature error affect blower speed.*

cal formulas involving many experimental parameters. Modifying or later optimizing such a solution is very difficult because of its lack of transparency.

Fuzzy logic makes this design process faster, easier, and more transparent. It can implement control strategies using elements of everyday language. Everyone familiar with the control problem can read the fuzzy rules and understand what the system is doing and why.

It also works for control systems with many control parameters. Designers can build innovative control systems that would have been intractable using traditional design techniques.

The future for fuzzy logic in automotive engineering is bright. Semiconductor manufacturers are incorporating fuzzy-logic instruction sets in their controllers. Motorola just introduced the new 68HC12 family of 16-bit micros that integrate a complete instruction set for fuzzy logic at no extra cost.

Another new development is the upcoming IEC **113 l-7** fuzzy-logic standard [19]. This international standard defines consistent fuzzy-logic development and documentation procedures.

With these two developments, designing with fuzzy logic becomes a much simpler task. ❑

Constantin von Altrock began research on fuzzy logic with Hewlett-Packard in 1984. In 1989, he founded and still manages the Fuzzy Technologies Division of Inform Software Corp., a market leader in fuzzy-logic development tools and turn-key applications You may reach Constantin at cva@inform-ac.com.

REFERENCES

[1] Intel, "Fuzzy Anti-Lock Braking System," developer.intel.com/ design/MCS96/DESIGNEX/235 1. htm, 1996.

[2] "Benchmark Suites for Fuzzy Logic," www.fuzzytech.com/ e_dwnld.htm, 1997.

[3] N. Matsumoto et al., "Expert antiskid system," *IEEE IECON'87, 810-816, 1987.*

[4] C. von Altrock, *Fuzzy Logic and NeuroFuzzy Applications Explained,* Prentice Hall, Englewood Cliffs, NJ, 1995.

[5] H. Kawai et al., "Engine control system," *Proc. of the Int'l Conf. on Fuzzy Logic and Neural Networks,* Iizuka, Japan, 929-937, 1990.

Control Anything...

EASILY

The PK2270™ EasyStart Kit

- The fastest, easiest way to develop control systems
- 30 I/0 lines, RS232, RS485, rugged enclosure, LCD, & keypad
- Includes all necessary hardware, simplified software development system, step-by-step documentation and many sample programs.

INNOVATION IN CONTROL TECHNOLOGY

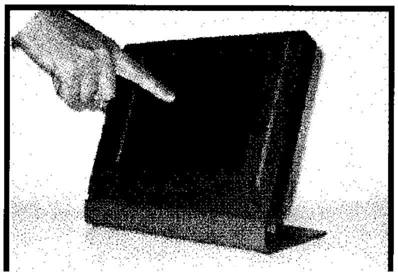
Call 1-888-362-3387 toll-free for your PK2270 EasyStart Kit or to receive a free catalog. We'll ship within 24 hours.

ZWORLD

2900 Spafford Street TEL 916-757-3737
Davis CA 95616 USA FAX 916-753-5141 http://www.zworld.com

#112

Touch The Future



LCD Touch Monitors
LCD Touch Screens
VGA LCD Displays
LCD Controllers
ISA, **PC104,** Analog, Video

EARTH Computer Technologies

Lowest Prices on **Earth!**

1110 Calle Cordillera - San Clemente - CA - 92673
Ph: 714-361-2333 - Fax: 714-361-2121
Email: oemsales@flat-panel.com
FREE CATALOG available at http://www.flat-panel.com

#113

[6] L. Feldkamp and G. Puskorius, "Trainable fuzzy and neural-fuzzy systems for idle-speed control," *2nd* IEEE *Int'l. Conf. on Fuzzy Systems,* 45-51, 1993.

[7] H. Takahashi, K. Ikeura, and T. Yamamori, "5-speed automatic transmission installed fuzzy reasoning," *IFES'91–Fuzzy Engineering toward Human Friendly Systems, 1136-1137, 1991.*

[8] H. Ikeda et al., "An intelligent automatic transmission control using a one-chip fuzzy inference engine, " *Proc.* of the *Int'l.* Fuzzy *Systems and Intelligent Control Conf. in Louisville, 44-50, 1992.*

[9] *P.* Sakaguchi et al., "Application of fuzzy logic to shift scheduling method for automatic transmission," *2nd IEEE Int'l. Conf. on Fuzzy Systems, 52-58, 1993.*

[10] *C.* von Altrock, B. Krause, and H.-J. Zimmermann, "Advanced fuzzy logic control of a model car in extreme situations," *Fuzzy Sets and Systems,* **48:1**, *41-52, 1992.*

[11] INFORM GmbH/Inform Software Corp., fuzzy*TECH and NeuroFuzzy Module 5.0 User's Manual,* Chicago, IL, 1997.

[12] H.-J. Zimmermann and U. Thole, "On the suitability of minimum and product operators for the intersection of fuzzy sets," *Fuzzy Sets and Systems, 2,* 173-186, 1979.

[13] *C.* von Altrock and B. Krause, "On-Line-Development Tools for Fuzzy Knowledge-Base Systems of Higher Order," *2nd Int'l Conf. on Fuzzy Logic and Neural Networks Proceedings,* Iizuka, Japan, 1992.

[14] L.I. Davis et al., "Fuzzy Logic for Vehicle Climate Control," *3rd IEEE Int'l. Conf. on Fuzzy Systems, 530-534, 1994.*

[15] J.-P. Aurrand-Lions, M. des Saint Blancard, and P. Jarri, "Autonomous Intelligent Cruise Control with Fuzzy Logic," *EUFIT'93–1st Eur. Congress on Fuzzy and Intelligent Technologies,* Aachen, 1-7, 1993.

[16] V.M. Thurm, P. Schaefer, and W. Schielen, "Fuzzy Control of a Speed Limiter," ISATA Conf., 1993.

[17] www.fuzzytech.com/e_a_spe.htm.

[18] R. Russ, "Designing a Fuzzy Logic Traction Control System," *Proc. of the Embedded Systems Conf., 2, 183-196, 1994.*

[19] www.fuzzytech.com/e_iec.htm.

SOURCES

***fuzzy*TECH Development System**
Inform Software Corp.
200 1 Midwest Rd.
Oak Brook, IL 60523
(630) 268-7550
Fax: (630) 268-7554
www.fuzzytech.com

Fuzzy-logic 68HC12 microcontroller
Motorola MCU Information
P.O. Box 13026
Austin, TX 78711
(512) 328-2268, x985
www.mcu.motsps.com
www.fuzzytech.com/motorola.htm

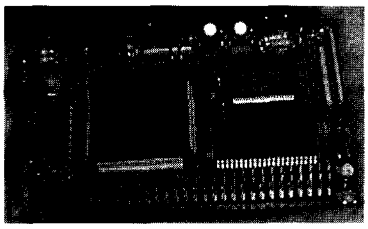
I R S

401 Very Useful
402 Moderately Useful
403 Not Useful

Introducing RI 67-JIF and READS1 66

R167-JIF - High Performance CAN-ABLE™ Industrial Controller

- Industrial Boards with GND & VCC planes
- Built-in CAN interface circuitry
- Operates at 20MHz with zero wait states
- Five 16 bit timers & watchdog timer
- 16 Channel 10-bit A-to-D
- 54 bits of general-purpose I/O
- 512KB RAM, 512KB 5v FLASH
- 4 channels PWM
- Just 2.75" x 3.5"
- Comes with C and Assembly Demos and FLASH programming software



READS166 V3.00, an Integrated Visual Development Environment. Smart Hardware Software Integration

- Assembler, C Compiler, and Monitor
- Native 32-bit Code to run on Windows95 and WindowsNT
- Sophisticated Project Management System
- Drag-and-Drop Code Development Libraries
- Hardware-Aware Dynamic Debugging: Single Step, Breakpoints, Animation
- Supports interrupts, Mixed Mode Programming (C -Asm)

R167-JIF -- $167, READS1 66 Lite -- $300

RIGEL Corporation
PO Box 90040, Gainesville, FL 32607
Tel: 352-373-4629, FAX 352-373-l 786
Download fully functional READS166 Demo (4K), www.rigelcorp.com

#114

AVT inc.
Advanced Vehicle Technologies, Inc.

Products to support the design and testing of vehicle network components

AVT's Services...
- Automotive Network Engineering: J1850 VPW, PWM, & ISO-9141
- Hardware Design: Digital & Analog
- Embedded Software/Firmware Development
- PC Based Software Design & Development
- Custom Software & Hardware Development, Assembly, and Test

AVT's Products...
- AVT-716: Triple Interface J1850 (VPW & PWM) & ISO-9141 (RS-232/422)
- AVT-715: Dual J1850 Interface VPW & PWM (RS-232/422)
- AVT-921: Dual J1850 Interface VPW & PWM (ISA Bus)
- AVT-1850: J1850 VPW Development System

"Vehicle network expertise, products, and resources"

1509 Manor View Road • Davidsonville, MD 21035
(410) 798-4038 voice, (410) 798-4308 fax

e-mail: avt-inc@ari.net • home page: http://www2.ari.net/avt-inc/

#115

Self-Evolving Systems and Liability

FEATURE ARTICLE

Rod Taber

Who's Responsible for a System's Growing Pains?

If software is self-evolving, how can a corporation protect itself?After all, who knows what changes and developments will evolve as the software interacts with its environment? Rod gives guidelines to help you keep out of the courtroom.

At first glance, self-evolving systems shouldn't be in critical applications. Does it make sense to sell a heart pacemaker that places itself beyond testing? Should the FDA approve such a system? Does it deserve a patent? What legal problems arise for the manufacturer if it fails?

Quick answers to such questions are beyond the scope of a short article. Even the legal system yields multiple conflicting opinions as it adjusts to new technology and aligns itself with the ever-changing social milieu. Nevertheless, in this article, I want to highlight some issues for further discussion.

The 1980s brought us small, fast, and cheap computers. We developed new sensors and estimation theory. Neural nets and fuzzy controllers are now in video cameras, car transmissions, elevators, and a host of other smart consumer products. We progressed from no adaptation to offline and online adaptation.

The next step, self-evolving systems, will produce very smart machines—machines that reconfigure themselves depending on incoming data. Self-evolution goes beyond customary adaptation. It entails paradigm shifts.

An example from pattern recognition illustrates the idea. Incoming data forces a machine to shift from a Gaussian classifier to one based on some other alpha stable distribution. In doing so, it must forsake using the arithmetic mean of pixel regions as a measure of central tendency.

The mean and higher order moments of the induced distribution may not exist, as in the Cauchy distribution. The machine may have to invent as it goes along because the characteristic exponent of the distribution curve changes with time.

In a different context, a system may switch from using one credit-reporting agency to another, depending on real-time transaction costs and accuracy. Or, a pacemaker shifts from a constant to a variable rate voltage spike to counter impending arrhythmia.

All of this sounds a bit scary. Do we really want machines with a high degree of nondeterminism in special applications?

Should machines evolve? Evolution has been regarded as a fact of life since Spencer first spoke of the survival of the fittest. While the context was biology over long-time scales, adaptation plays an equivalent role in systems and cybernetics.

Old-line'computer companies fall by the wayside as more agile companies offer new ideas and approaches. We know that smart machines are here to stay and that machine IQ will likely increase. It's a matter of competition for market share and visibility.

Also, our legal system will necessarily twist, resist, and adapt as it has in the past. Witness the sea change in public opinion on the tobacco issue.

Twenty years ago, a reasonable person would accept that in taxing the product, states in effect legalized and legitimized tobacco use. States became partners with the tobacco industry, subsidizing it and helping farmers grow tobacco for a slice of the pie.

Now, the states see a potential source of unearned income. They argue for an even larger slice, based on revisionist history that has something to do with the tobacco companies acting in concert with cartoon camels to deceive the public.

My point is that, right or wrong, perspectives change with time and information. What can we learn or infer from the tobacco problem? We need to watch out for:

- retroactive governmental disapproval for devices
- new liability for old devices
- people who sue when they identify a plausible target for litigation

What threats loom on the horizon for computing as more and more devices are able to evolve? Surely, the usual dangers of personal-injury suits, detrimental-reliance suits, and patent litigation won't disappear.

And, the future may bring strange cultural values and severe financial crunches. The federal pension and social-security crises will almost surely wear down state treasuries and create financial hardships.

Hence, governments and individuals will look for easy litigation targets. In that environment, it's going to be easy to blame computers and smart devices for problems real and imagined.

Is a self-evolving system inherently riskier than a dumb system that refuses to change even in the face of overwhelming evidence? The legal question is whether an increase in untrustworthiness translates into enhanced legal liability and litigation probability.

Conversely, does an increase in trustworthiness decrease liability? Before engaging those questions, let me make a few observations about conventional computing and the current legal system.

CONVENTIONAL COMPUTING

Most historians credit Von Neumann with creating the foundations for digital computing. His ideas ranged from cellular automata to making reliable systems out of unreliable components. Digital computers soon found their way from the War Department to the Census Bureau and on to the general data-processing departments of corporate America.

These so-called Von Neumann computers downsized 20 odd years ago through brand names like IMSAI, ALTAIR, Processor Technology, and Southwest Technical Products. These small computers held their position until the mid 1980s when computers surfaced as PCs and Macintoshes.

Years ago, technicians repaired vacuum-tube computers on a minute-

Patents

Moore's law states that technology improves by a factor of two every 18 months. Chip densities double as do internal computer clock rates, but new algorithms come along only once in a while. Consumer products and their patents shows an interesting fact-virtually all smart machines use a version of the Mamdani controller. Thus, there are several thousand products and patents based on a single variety of model-free estimator.

The same can be said for neural networks but with a little less force. The practical implication for fuzzy and neural engineers is that the projects they're working on may be already patented. Figures i and ii show the significant increase in the number of fuzzy and neural patents over the last decade. But, how many engineers actually read patent summaries?

Small companies can't afford a patent dispute, but they may be forced into defending themselves-and patent cases aren't cheap. It may take a million dollars just to get to first base. Not only do you pay for several law firms working perhaps full time for more than a year, but you have travel expenses, court reporters, expert witnesses, and a whole realm of problems you just don't want to deal with. On top of that, a case may tie up senior engineering staff for more than a year. Your competitors flourish, and you languish. In a real sense, even if you win, you lose. It sounds bleak, and it is.

Large companies have full-time legal counsel and several law firms on retainer. What can a small company do? Well, forewarned is forearmed. The U.S. Patent Office established more than 70 depositories around the country. Large libraries subscribe to CD-ROM editions of patents granted, and the library staff can help answer your "how do I" questions quickly and at no cost.

Printed patents do cost a little, but you can often view them on microfilm for free. Also, patent information is available on the Web. You can sit at your computer and search all patents from dozens of years ago to present day. In the U.S., no one has easy access to patent applications prior to a patent being granted. This situation may change if special interests in Congress have their way. Last but not least, retain a patent search firm or patent attorney. **It's** a lot cheaper than going to court.

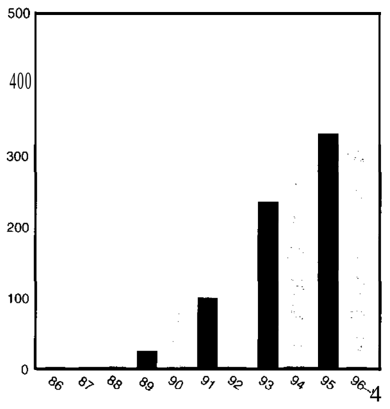


Figure ii-*In this graph, you see that the approximate number of neural patents issued by the U.S. Patent Office for the last decade was nearly double that of the fuzzy patents [1].*

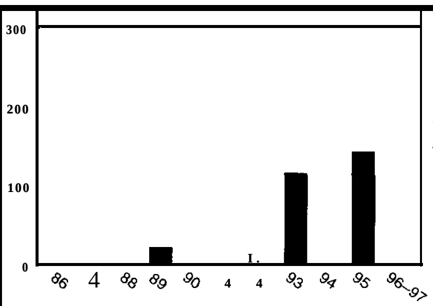


Figure *i--This graph displays the approximate number of fuzzy patents issued per year by the U.S. Patent Office [1].*

to-minute schedule. Tubes replaced in the morning literally burned out by the afternoon.

The situation wasn't much better with early dynamic and static RAMs for the Intel 8080 and Motorola 6800 machines. Every techie from that era remembers spending a huge amount of time writing memory checkers. We wrote programs to tell us the row and column address of bad 21L02 chips on 4-KB boards. We used a lot of chips.

Now, desktop computers are much more complex than the mainframes of yesteryear in most respects, and they're much more reliable. In this case, complexity correlates with reliability.

Software has also changed since the days of the vacuum tube. We wrote self-modifying code. Programmers loved its on-the-fly nature.

But, software engineers labeled it a hazard and a liability. It was difficult to maintain, and the code depended on the sequence of data inputs and a lot of other stuff, none of which was necessarily deterministic. Electrical engineers therefore designed separate data and instruction spaces to prevent it or at least minimize its use.

No one seems to track the prevalence of self-modifying code anymore. It isn't an issue. Theoretical computer scientists still wrestle with the provability of even static code. We have existence theorems on computability, but we can't prove that a particular program works as intended. Every program is correct for some problem, although it may incorrect for the problem at hand.

We can show by example that many "real" programs are not correct for all inputs. This is especially true for artificial intelligence in all its forms.

Pattern classifiers always fail, given enough noise in the signal. We know these devices will fail for some data inputs. Such programs satisfice specifications. In other words, they do not always satisfy those specifications but rather reach a reasonable compromise.

Today, we simulate fuzzy systems, neural networks, and genetic algorithms in ways typically described as nonalgorithmic. The new function approximators and state search techniques are model-free because the mapping from system inputs to system outputs doesn't use a math model of the plant. Sanity checkers may supervise model references, but the computing system doesn't depend on a mathematical model of the process.

Model-free estimators are clever end runs around classical model techniques like the Kalman filter. There are no differential equations to solve. The physical systems of interest either can't be expressed as differential equations or, if such equations exist, they're beyond solution.

In a patent trial, how do you explain Fourier transforms to an uneducated jury!

Fuzzy model-free estimators such as the Mamdani controller calculate the expected value of a parameter given the input data and fuzzy sets. Neural nets of the popular feed-forward variety suffer from the problem of "connectionist glop."

Given inputs, we only know in the context of such a network that the webs of interconnections produce approximately correct answers. Thus, New Age paradigms induce a certain measure of uncertainty in our systems.

Over the past 50 years, we've designed hardware that can fail, written software with few if any provable properties, and built OSs that crash and deadlock without warning.

Yet, these systems work extremely well. We issue patents and approve medical devices containing computers. We have, to a large extent, built reliable systems out of unreliable components. Von Neumann would be proud.

OUR LEGAL SYSTEM

Things were a lot simpler in the old days when the issues were water rights and whether Smith invented a particular kind of wheel before Jones did. Anyone can tell one wheel from another, and the calendar was never a mystery.

How complicated can a wheel be? More importantly, how complicated can the issue of its invention be? Actually, such questions can be very complicated. In fact, these mundane issues can be as complex as a patent case on digital telephony.

So, why do we have everlasting trials costing millions of dollars? Unfortunately, there isn't just one answer.

But, we can make one crucial observation. Back in the past, nearly anyone drawn from the population at random could understand the issues. A wheel spoke was a piece of metal or wood. Jurors could see and touch it and render a reasonable judgement because they understood the issues. There was little need for expert witnesses.

The New Age brings a constellation of technologies. Instead of wheels and spokes, we have bit rates, quadrature modulation, Mersenne number transforms, and virtual memory. Expert witnesses can explain the issues, but it takes an educated jury to understand the abstract arguments.

Vonnegut opined that scientists who can't explain to ten-year-olds what they're doing are charlatans. Unfortunately, scientists can't always explain their opinions in the present forum. And, it's not entirely their fault.

Juries fall asleep during three-hour lectures on the polymerase chain reaction and the virtues of Western Blots. These subjects are not elements of a low-tech/no-tech world.

Some jurors have little more than a high-school education. Some don't even have that. You have to explain graph theory, Fourier transforms, and PID control to them, and they can't ask questions. And, they decide whether your company lives or dies.

What do we have now! Aside from unusual cases with master experts and judges adjudicating, attorneys intentionally select undereducated people for jury duty. They don't want jurors with high powers of critical ability. The lawyers may not understand the technology, and they certainly don't want the jury understanding things they can't understand themselves.

Other lawyers want the best-educated juries they can find. These lawyers may have advanced degrees, so such a jury is not a threat but an asset.

I'm sure you can imagine a court case in which a clever attorney sways a jury by proclaiming that a child's death was due to a device that changed its mode of operation at the whim of its computer. Examination of a witness might go along these lines:

"You made this device knowing full well that you could not test it?"

"We did test it."

"The system you tested was the same as the one that killed the child?"

"The system was the same."

"Then which of your so-called algorithms killed the child?"

"I don't know. It changes."

"Then wasn't the child killed at the whim of a computer?"

Hearing "whim of a computer" can wake a sleeping jury. They may not understand computer technology, but they can sympathize with a child, and so the case becomes child versus computer. When people don't understand the important issues, they focus on what they do understand-regardless of the relevance.

STEPS FORWARD

The foundation of all computing devices is a little shaky. We can't prove that programs or systems are correct.

Programs contain self-modifying code. Memories fail in nondeterministic ways. Neural nets work mysteriously. Our systems tend to satisfice, not satisfy (i.e., compromise), specifications.

On top of this pyramid of uncertainty, we're adding yet another layer—self-evolving devices. Will this cherry on top of the sundae help society or not!

My view is that self-evolving systems need not have Orwellian connotations. They do not necessarily lead to social contractions as the Krell matter-from-mind machine did in the sci-fi movie *The Forbidden Planet.* They can be just as understandable as present systems, provided we engineer them that way.

An example of a good engineering process is how functions move from testbed to system via ASICs and VLSI. In other words, trusted software moves to chips, and new software comes along. Engineering moves forward. There's no reason that the legal system cannot move in parallel with technology.

The system has changed and continually evolves to meet new needs. There's always a time lag. Engineering and the legal system evolve on different time scales. There'll be bad product liability decisions and patent actions gone awry.

But, in the meantime, there are some ways to reduce liability in both arenas:

- exhaustively test systems
- enforce modularization standards on software and hardware components
- use step-wise refinement and hierarchical design
- put experts in the jury
- let the jury pose relevant questions to the experts

These changes may help keep court proceedings from degenerating into a battle of experts with their corresponding pedigrees and side issues. It would also tend to reduce the number of loaded and tricky questions.

When everyone understands the issues stripped of their legalese, I believe the questions of liability and infringements will have much simpler answers.

Object-oriented programming and other good software and hardware engineering practices suggest that there is no a priori reason why devices of arbitrary complexity can't be reliable.

Reliable devices should decrease the threat of device misadventures and decrease resulting liability.

And on the legal side, courts appear to be experimenting with limited measures of reform, and Congress is trying to reform patent and tort laws. ❑

Rod Taber holds a Ph.D. in computer science from Texas A&M University. He writes neural network and fuzzy systems code and provides technical support for patent litigation in those areas. You may reach Rod via his Web site at www.patent-neural-fuzzy.com.

REFERENCES

[1] U.S. Patent Office, www.uspto.gov.

I R S

404 Very Useful
405 Moderately Useful
406 Not Useful

RISC
32-bit controller

- *28 Programmable analog I/O, SE or differential, 4-20 mA*
- *Keypad & display operator interface*
- *24-Plus digital I/O*
- *4 RS-232/485 ports, optically isolated*
- *4 Counter/quadrature encoder, pulse timing*
- *Flash EPROM program storage*
- *C Programmable*
- *Starting from $595*

Call for NEW catalog & information on more embedded controllers!

REMOTE™ PROCESSING
The embedded control company

Incredibly powerful

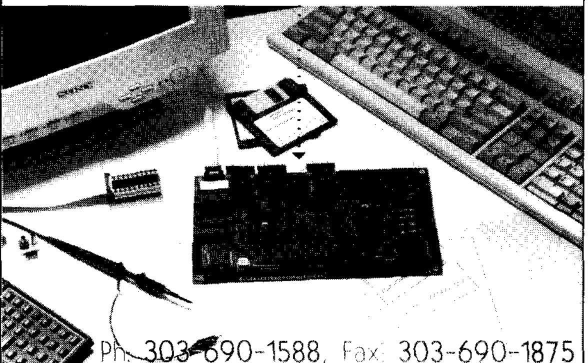
At 50 *nS* instruction time, our WC-400 delivers the power that an 8 *or 16-bit* controller *can't.*

Inherently complete

We've included everything to handle *even the* most demanding industrial I/O requirements.

Instinctively simple

Easy *to program,* run, and get **going!**



Ph. 303-690-1588, Fax 303-690-1875

#116

FEATURE ARTICLE

Walter Banks

Fuzzy Concepts Using C

Walter takes fuzzy logic into the trenches with this down-to-earth implementation of fuzzy logic using C on small micros. He uses examples from real-world applications to illustrate his points.

Much of the power of fuzzy logic comes 'from its ability to focus on specific areas of interest. In many cases, fuzzy-logic–based systems can produce superior results with significantly lower resolution than congenital approaches to problem solving.

A lot has been written about fuzzy logic as another programming paradigm, but fuzzy logic is now regularly mixed with conventional code in many applications. Fuzzy-logic operators are the formal method of manipulating linguistic variables.

This article is primarily aimed at the eight-bit embedded-system developer who is using a high-level language to implement an application. The notation in the fuzzy examples uses the syntax of Fuzz-C, a C preprocessor that translates mixed fuzzy and C into C.

There is no need or reason (other than convenience) to use this preprocessor. The examples show the direct relationship between fuzzy expressions and C. My main point: fuzzy logic doesn't need special hardware or software, and it offers a powerful tool for problem solving.

Central to fuzzy-logic manipulations are linguistic variables. Linguistic variables are nonprecise variables that convey information.

I can say, for example, that I drove to work fast or that I drove at the speed limit. The first description depicts behavior that borders on reckless. The second portrays me driving a profile of time, space, and different speeds on the expressway and residential roads.

Depending on context, driving at the speed limit can have other interpretations as well. If the expressway speed limit is 55 MPH, then it might mean that I'm driving close to 55 MPH—perhaps 52 or 58 with some distribution around 55. My attention to the exact number may be changed by other factors. Known police-radar locations often limit my speed to an exact 55!

Linguistic variables in a computer require there to be a formal way of describing the linguistic variable in crisp terms the computer can deal with. The graph in Figure 1 shows the relationship between measured speed and the linguistic term FAST.

Each of us may differ about what counts as fast. But, at some speed, we all say that it is not fast, and at some other point, we agree that it is fast.

In the space between fast and not fast, the speed is, to some degree, both. The horizontal axis in Figure 1 shows the measured or crisp value of speed. The vertical axis describes the degree to which a linguistic variable fits the crisp measured data.

I can describe temperature in a nongraphical way with the declaration:

```
LINGUISTIC Speed  TYPE  unsigned
  int MIN 0 MAX 100 {MEMBER FAST
  {60, 80, 100, 100}}
```

This declaration describes both the crisp variable Speed as an unsigned int

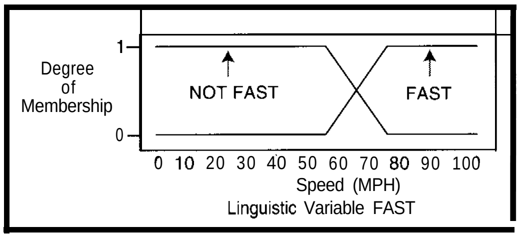


Figure I--Here *the* **linguistic** variable FAST **is compared** *to* **the crisp speed, producing a degree of membership.**

Listing I--This code *fragment* determines *the degree of membership of FAST in the variable Speed.*

```
unsigned int Speed:    /* Crisp value of Speed */
unsigned char Speed_FAST (unsigned int __CRISP){
  if (  CRISP  < 60) return(0);
  else{
    if ( _CRISP <= 80) return((( CRISP  - 60) * 12) + 7);
    else {
      return(255); } } }
```

and a linguistic member FAST as a trapezoid with specific parameters.

If I add the linguistic variable FAST to a computer program running in an embedded controller, I need to be able to translate the graphical representation into code. The C code fragment in Listing 1 is an example of how this might be done.

The function Speed FAST returns a degree of membership scaled between 0 and 255. This type of simple calculation is the first tool required for calculations of fuzzy-logic operations.

The same code can be translated to run on many different embedded micros. The generated code for converting between crisp and fuzzy is fast and easily incorporated into embedded applications (see Listing 2).

Central to the manipulation of fuzzy variables are fuzzy-logic operators that parallel their Boolean-logic counterparts. These operators—f_and, f o r, and f_ no t-can be defined as three macros to most embedded-system C compilers as you see in Listing 3.

FUZZY EXPRESSIONS

Fuzzy expressions describe at a higher level how actions are to be performed. Implementing many of these fuzzy rules in an application results in a behavior.

This statement can read literally and evaluated with considerable precision:

```
IF Speed IS FAST AND Radar IS
  DETECTED
THEN Brake IS ON:
```

Both Speed and Radar can be measured as crisp values. Radar-detector relevance can be measured by the rate of little beeps the detector emits as you pass the industrial park or by the loud tone you get when the cop points the radar gun directly at your car.

.isting *P-Here's an implementation example of degree of membership of FAS Ton a typical 8-bit* microcontroller.

```
0050                             unsigned int Speed: /* Crisp value
                                   of Speed */
                                 unsigned char Speed-FAST (unsigned
                                 int CRISP)
0051                             {
0100  F5  51  MOV    $51,A
0102  D3      SETB  C            if ( CRISP < 60) return(O):
0103  94  3C  SUBB  A,#3C
0105  40  02  JC     $010B
0107  E4      CLR   A
0108  22      RET
                                 else {
010A  E5  51  MOV    A,$51         if (__CRISP <= 80)return(((__CRISP
                                     60) * 12) + 7);
010B  D3      SETB  C
010C  94  50  SUBB  A,#50
010E  40  08  JC     $011A
0110  E5  51  MOV    A,$51
0112  D3      SETB  C
0113  94  3C  SUBB  A,#3C
0115  31  46  ACALL $146
0117  24  07  ADD    A,#07
0119  22      RET
                                   else {
011A  78  FF  MOV    RO,#FF          return(255); } } }
011C  22      RET
```

A simple calculation relates the crisp values to the fuzzy range between fuzzy 0 and fuzzy 1. Fuzzy expressions arc also evaluated between fuzzy 0 and fuzzy 1.

The fuzzy result of the controlling expression determines the degree of braking required. A brake value of fuzzy 0 is no brake pressure, whereas a value of fuzzy 1 is full braking. It's important to note that many of the linguistic conclusions arc a result of the general form of the above equation.

FROM CRISP TO FUZZY

Computations performed in the fuzzy domain must consistently translate logical operations in the crisp domain to degrees of membership in the fuzzy domain. Much of this article is devoted to practical implementations of developing a degree of membership from the crisp data.

The simplest translation is converting basic Boolean 0 and 1 data to fuzzy 0 and 1. This macro enables Boolean-logical comparisons to be made in fuzzy-logic expressions:

```
#define logical_to_fuzzy(a)
   ((a)?F_ONE:F_ZERO)
```

This C macro converts a logical value to a fuzzy 0 or fuzzy 1. It simply tests for logical 1 or 0 and replaces the Boolean-logical value with a fuzzy 1 or 0 (i.e., F ONE, F_ZERO).

First, you replace the crisp equality comparison with a fuzzy comparison that accounts for a particular range of data. Then, you base a comparison on three data values-the comparison point, the range until the comparison has failed (delta), and the current variable value.

Delta is the distance to a value where the current comparison ceases to be important. Consider for a moment the definitions in Listings 4a–f. In each case, the delta value returns a fuzzy 0 or fuzzy 1, and any further deviation from the center point does not change the result.

Most people have a good sense of the delta required by an application. Figure 2 shows a Fuzzy Equal relationship to the crisp domain. As Listing 4a shows, the definition of F u z zy

Listing 3—*These* linguistic variables andoperafors *written in C* define *typical Boolean* operations.

```
#define f_one        0xff
#define f_zero       0x00
#define f_or(a,b)    ((a) > (b) ? (a): (b))
#define f_and(a,b)   ((a) < (b) ? (a): (b))
#define f_not(a)     (f_one+f_zero   a)
```

Eq u a 1 can be easily implemented on most small microcomputers.

This technique can be extended to normal arithmetic comparisons. In the following example, **Fuzzy Not Equal** has the same membership as F_NOT (F_EQ(v)). The definition of F_NOT **is:**

```
#define  F_NOT(a)(F_ONE-(a))
```

All of the normal crisp identifiers can be used in the fuzzy domain. Membership functions for each of the normal crisp comparisons are given in Listings 4b–f.

The implementations for the fuzzy numerical comparisons all use simple linear functions to calculate the degree of membership. The merits of using some form of exponential functions on small eight-bit microcontrollers is attractive.

There hasn't been a universal consensus, but power functions can express emphasis. For instance, we might say a long fly ball is NEARLY a home run and a ball that traveled to the warning track is **VERY-NEARLY** a home run.

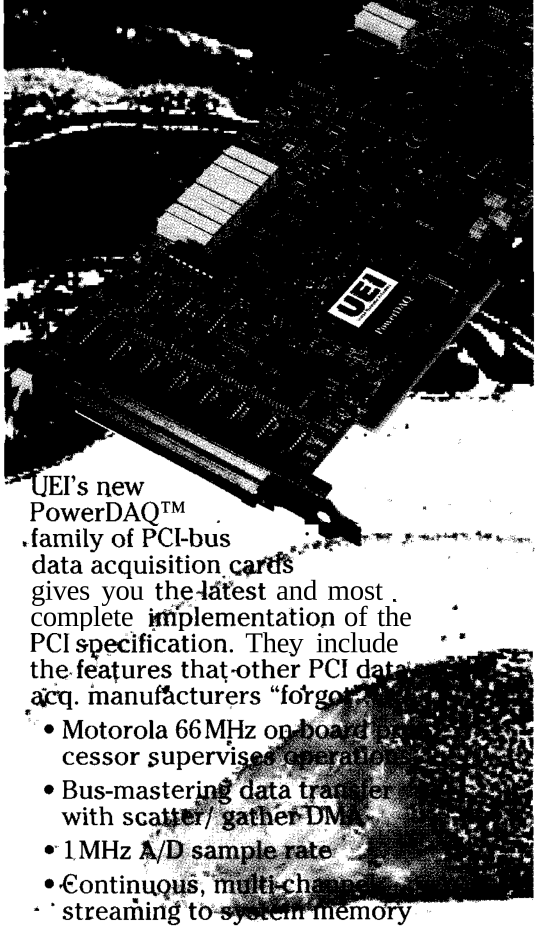
Most exponential functions can be implemented as the sum of a series. For example, a reasonable implementation for **VERY-NEARLY can** be made with the definitions in Listings 5a and 5b.

The linguistic function has two slopes with an intersection halfway from delta to the setpoint. Exponential functions can be implemented by degrees, where a single linear function has a degree of 0 and **VERY-NEARLY** (as just described) has a degree of 1. Such

Listing 4—*All normal arithmetic comparisons can be made in the fuzzy domain. Using fuctions such as these enables the developer to make close comparisons and easily mix linguistic operations with crisp ones. This effectively adds a linguistic data type to a developer's fool box.*

```
  typedef DOMtype unsigned char

a) DOMtype F_EQ(v,cp,delta){
    long m = ABS(cp - v);
    if (m > delta) return(F_ZERO);
      return((m/delta)  *  (F_ONE  F_ZERO));}

b) DOMtype F_NE(v,cp,delta){
    long  m = ABS(cp v);
    if (m > delta ) return(F_ONE);
      return(F_ONE -  ((m/delta)   *  (F_ONE F_ZERO)));}

c) DOMtype  F_LT(v,cp,delta){
    if (v < (cp - delta)) return(F_ONE);
    if (v < cp)
      return(F_ONE - ((cp  v/delta)  *  (F_ONE - F_ZERO)));
    else  return(0);}

d)DOMtype  F_LE(v,cp,delta){
    if (v < cp) return(F_ONE);
    if (v < (cp + delta))
      return(((v cp)/delta)  *  (F_ONE - F_ZERO));
    else  return(0);}

e)DOMtype  F_GT(v,cp,delta){
    if (V < CP) return(F_ONE);
      return(F_ONE  (((v  -  cp)/delta)*  (F_ONE  F_ZERO))); }

f)DOMtype  F_GE(v,cp,delta){
    long  m  =  ABS(cp v);
    if (V > CP) return(F_ONE);
      return(F_ONE  ((m/delta)   *  (F_ONE F_ZERO)));}
```

If it's not PowerDAQ™ it's not PCI



UEI's new PowerDAQ™ family of PCI-bus data acquisition cards gives you the latest and most complete implementation of the PCI specification. They include the features that other PCI data acq. manufacturers "forgot":

- Motorola 66MHz on-board processor supervises operations
- Bus-mastering data transfer with scatter/ gather DMA
- 1 MHz A/D sample rate
- Continuous, multi-channel streaming to system memory
- Simultaneous analog and digital input and output
- Clean 32-bit drivers for Windows 95 &Windows NT
- DLL library supplied with each board (includes source code)

Call today for your free catalog:

(800) 829-4632

UEI

United Electronic Industries

10 Dexter Ave, Watertown, MA 02172

Tel: (617) 924-1155
Fax: (617) 924-1441

internet: www.ueidaq.com
E-mail: sales@ueidaq.com

#118

implementations can be automated with reasonable compilers.

Degrees of membership normalize the structure of the data to be scaled between fuzzy 0 and 1. Exponential functions can be applied in series with degrees of membership to emphasize or deemphasize membership functions in application code.

HEDGE MODIFIERS

The C macro HEDGEmodifier specifies the hedge v e ry as being of order 1 (single break in power function):

```
#define very(a)
   HEDGEmodifier(1,a)
```

Listing 6a defines a hedge of order 1. Note that this function is easily implemented on most small microcomputers.

Negative hedge modifiers describe such concepts as ROUGH LY. I might say a certain man is about 45 and be referring to someone within a year or two of 45. I can then describe a broader range ofindividualsas ROUGHLY.ABOUT 45.

```
#define  ROUGHLY(a)
   HEDGEmodifier(-1,a)
```

A hedge modifier of order -1 is defined as in Listing 6b. As an alternative, hedge functions can be implemented as simple table look-up functions with at most some linear interpretation.

Hedge functions may be used with any membership function and may be cascaded to strengthen emphasis. Two hedges of order 1 in series yield a hedge of order 3. I haven't needed to implement hedges other than 1, 0, and -1.

Hedge functions permit the application designer to formally add emphasis. Here, hedge functions have a common mathematical basis and an application-specific meaning. And, they smooth out the roughness in descriptions.

The quest for improved accuracy in the last twenty years has mostly been a quest for improved numerical relevance. The fuzzy domain always focuses on the relevant data and limits to fuzzy 1 or 0 when the data ceases to be relevant.

IN THE REAL WORLD

The techniques I've described go a long way toward describing a problem in an effective and natural way. Crisp comparisons are necessary in some contexts. But in the last few years, we've discovered many more applications for fuzzy logic than we had previously imagined.

I am primarily in the code-creation business and recognize that most application software is written on conventional computers with conventional instruction sets. All of the definitions in this article can be implemented on even basic eight-bit micros using acceptably few instructions, typical shifts, adds, and comparisons-things that eight-bit micros do well.

It's practical to build fuzzy-logic control systems in consumer appliances. Against conventional implementation techniques, fuzzy logic competes well. In general, my experience has been that fuzzy logic is less math intensive and more logic intensive than traditional implementations. ❑

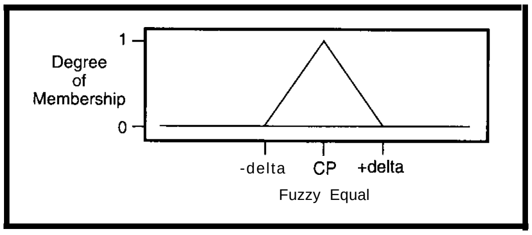


Figure 2—*Fuzzy equality checks the area around the c omp a r e value for close matches. The better the match, the higher the degree of membership.*

Listing *C--Exponential functions can be implemented by degrees. N EA R L Y and V E R Y_ N EA R L Y represent different levels of hedge functions.*

```
a) DOMtype NEARLY (long v,sp,delta){
     long m = sp - v;
     if (m < 0) return (F_ZERO);
       return((d/m) * (F_ONE - _ZERO));}

b) DOMtype VERY_NEARLY(long v,sp,delta){
     return((nearly(v,sp,delta) +  nearly(v,sp,(delta/2)) / 2);}
```

Listing 6—*Here's the general form for hedge functions for order 1 and order -1. Hedge functions can be cascaded to increase the extent of a condition.*

```
a) hedge order 1 translation is
     if (in < 0.5)
       return (in >> 1)
     else
       return (0.25 + (in - 0.5) * 1.5):

b) hedgemodifier  order -1 translation is
     if (in < 0.5)
       return (in * 1.5
     else
       return (0.5 + (in 0.5) * 0.5);
```

Walter Banks is president of Byte Craft Limited, a company specializing in software tools for embedded microprocessors. His interests include highly reliable system design, code-generation technology, programming language development, and formal code-verification tools. You may reach him at walter@bytecraft.com.

SOURCE

Fuzz-C, C 8051 compiler
Byte Craft Limited
421 King St. N
Waterloo, ON
Canada N2J 4E4
(519) 888-6911
Fax: (519) 746-6751
www.bytecraft.com

I R S

407 Very Useful
408 Moderately Useful
409 Not Useful

EMBEDDED PC

NOVEMBER 1997

Photo courtesy of Phar Lap Software

NPC

OPTOISOLATED PC/I 04 I/O MODULE

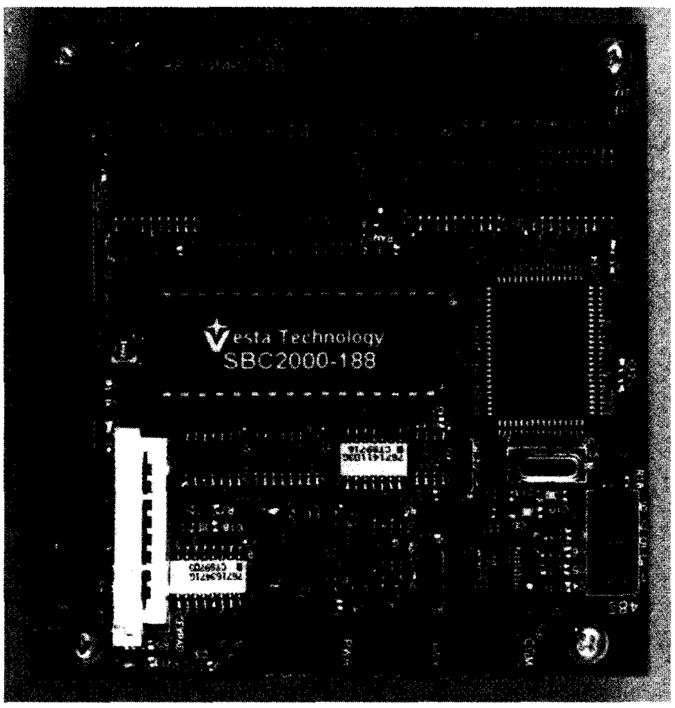
The AIM1 **04-Multi I/O** is a PC/I 04 I/O expansion module that uses an optoisolated serial bus to connect with its analog subsystem. It provides 8 opto-isolated digital inputs, 2 analog outputs, and 16 single-ended or 8 differential analog inputs. This combination gives the user everything required to interface with most sensors and actuators with enough digital capability to monitor switches and controls.

The board features a conversion time of 500 µs/channel, which is equivalent to converting up to 2k samples per second-fast enough for many general-purpose industrial instrumentation require ments. The analog-input section is based on a flexible 12-bit ADC. The two precision analog-output channels are provided by a 12-bit DAC. Each channel can update in 320 µs and output control signals or waveforms over a bipolar range of -5 to +5 V or 0–25-mA current loop sink. Both analog functions offer 1 -kV isolation between the PC/I 04 system and the electrical system under control.

The digital inputs use conventional optoisolation and can switch signals over a 10-30-V range. Each channel includes built-in debounce filtering with a 1 0-ms time constant and can interface digital signals to a maximum frequency of 50 Hz.

The AIM104-Multi I/O board sells for $295.

Arcom Control Systems
13510 S. Oak St. • Kansas City, MO 64145
(816) 941-7025 • Fax: (816) 941-7807
www.arcomcontrols.com **#510**



EMBEDDED CONTROLLER

The **SBC2000-188** embedded controller contains all the system-level resources needed in most embedded applications. Designed around an 'x86-compatible engine, the board features 1 -MB onboard RAM/ROM/flash space, watchdog timer, EEPROM, power-supply monitor, and real-time clock. It has two RS-232 serial ports, **an** alphanumeric LCD port, and a keypad port. Peripheral support is available from more than 20 interface products.

The SBC2000-188 features a wide range of development software. An embedded, multitasking compiled BASIC executes up to five times faster than QuickBASIC. C is supported using Paradigm's remote debugger, extensive iibraries from Vesta, and Borland or Microsoft C compiler. ROM-DOS is also available.

The PC/ 104 form-factor-compatible bus provides access to the most commonly used signals. The VAST (Vesta Addressable Synchronous Transfer) network connector accommodates a variety of low-cost, low-power peripherals. An optional RS-485 adapter converts one of the two RS-232 ports to either a two- or four-wire RS485 network.

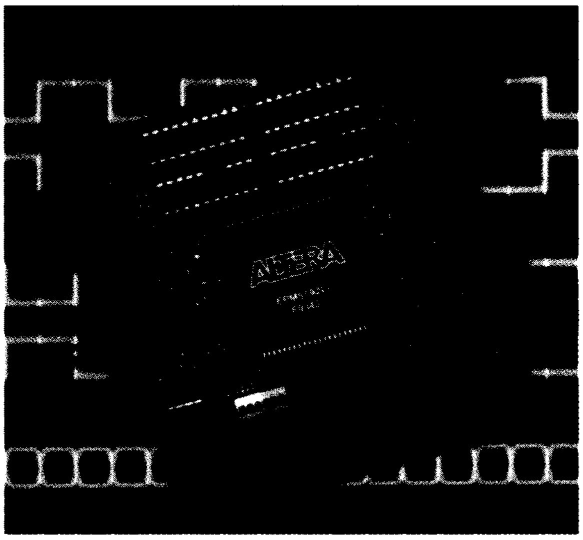
The SBC2000-188 sells for $159 in single quantities or $1 14 in 100-piece quantities.

Vesta Technology, Inc.
11465 W. **I-70**
Frontage Rd. N.
Wheat Ridge, CO 80033
(303) 422-8088
Fax: (303) 422-9800
vesta@vestatech.com

#511

NouveauPC

edited by Harv Weiner



PC/l 04 ENCODER INTERFACE

The Model 5912 **PC/104** Encoder Interface features a 1.3-MHz quadrature input rate with 24-bit presettable counters from up to four incremental quadrature encoders or pulse sources. Inputs may be single-ended or differential and are conditioned by a four-stage digital filter. The 5912 generates interrupts on index pulse, over/underflow, or compare value match, and it decodes Xl, X2, and X4 to provide higher encoder resolution.

Software libraries are compatible with most C, C++, Visual Basic, Visual C++, and Turbo Pascal compilers and include 16-bit drivers for Windows 3.1.

The Model 5912 costs $425 in a single-axis configuration. A development kit is also available for easy implementation in development and prototype stages.

Technology 80, Inc.
658 Mendelssohn Ave. N • Minneapolis, MN 55427
(612) 542-2980 • Fax: (612) 542-9785
www.tech80.com **#513**

EMBEDDED C++ COMPILER

The Embedded C++ Compiler (C++/EC++) is a subset of C++ optimized for resource-constrained embedded applications. Implemented as a user option within the C++ compiler's broader framework, the compiler achieves a 30–90% improvement in code size and run-time efficiency over full-blown C++, particularly for applications containing I/O of any kind.

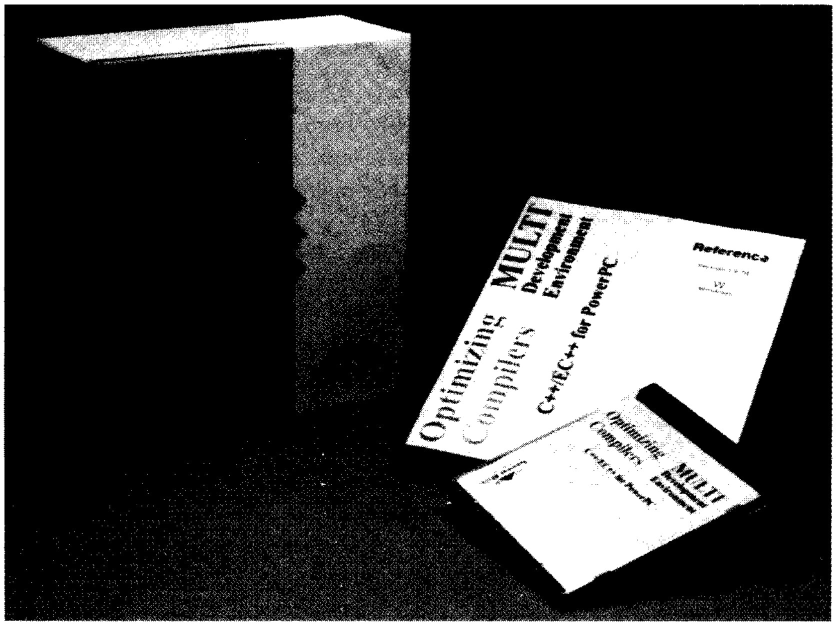
C++/EC++ provides many of C++'s object-oriented facilities, which enhance code reusability and simplify the partitioning and maintenance of complex code. However, it omits a number of C++ features that aren't essential for most embedded applications but significantly increase code size and impair run-timeefficiency. Included are multiple inheritance, virtual base classes, exceptions, run-time type identification, virtual function tables, and mutable specifiers.

C++/EC++ is fully integrated with the MULTI Development Environment, which automates every aspect of embedded software development. Featuring a window-oriented editor and an RTOS-aware source-level debugger, MULTI also includes a run-timeerrorchecker, application profiler, and project/version control.

Thecompilerwill initially beavailable for Motorola 68k and PowerPC family processors, with versions for the Hitachi SH, MIPS, ColdFire, and NEC V800 processors following later in the year. The compiler will also produce native code for workstations running Solaris 2.5. Development host options include PCs and workstations running Windows NT, Windows 95, Solaris 2.5, and HP/UX 10.0.

Windows NT and Windows 95 versions of the C++/EC++ compiler cost $3900. Unix versions are priced at $5400.

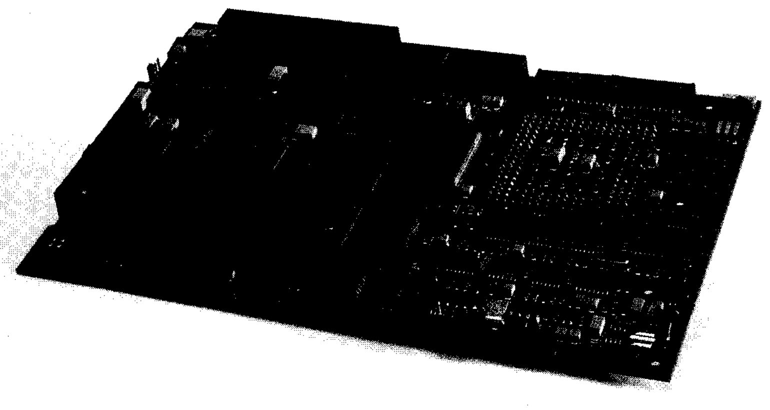
Green Hills Software, Inc.
30 W. **Sola** St.
Santa Barbara, CA 93101
(805) 965-6044
Fax: (805) 965-6343
sales@ghs.com • www.ghs.com **#514**



PC/104+ MPEG DECODER

The Traftech PC/104+ MPEG Decoder Card supports full audio/video synchronization with MPEG-1 system-layer decoding. It demultiplexes the MPEG bitstream, extracts the time stamps for the video and audio bitstreams, buffers both compressed audio and video data, and provides onboard audio decoding while maintaining full synchronization of audio and video signals throughout. It has built-in compensation for any differences in processing delay of the two signals.

Card features include full video CD 2.0 support, special display modes (e.g., pause, freeze, and slow motion), and automatic frame-rate conversion from all common MPEG picture rates to standard PAL or NTSC display frame rates. In addition, high-quality decoded video at a rate of 5 MBps, error detection, and concealment features with a direct interface to 5 12 KB of DRAM are included.



The card uses the Zoran ZR36120 PCI-bus controller, which permits the MPEG decoding to be handled at the maximum bus bandwidth of the 32-bit PCI bus. The card is plug-n-play compatible for seamless insertion into Windows 95 and Windows NT applications.

The MPEG Decoder Card sells for $595

Traftech, Inc.
698 1 Millcreek
Mississauga, ON
Canada **L5N 6B8**
(905) 814-1293
Fax: (905) 8 14- 1292
info@traftech.com
www.traftech.com

#515

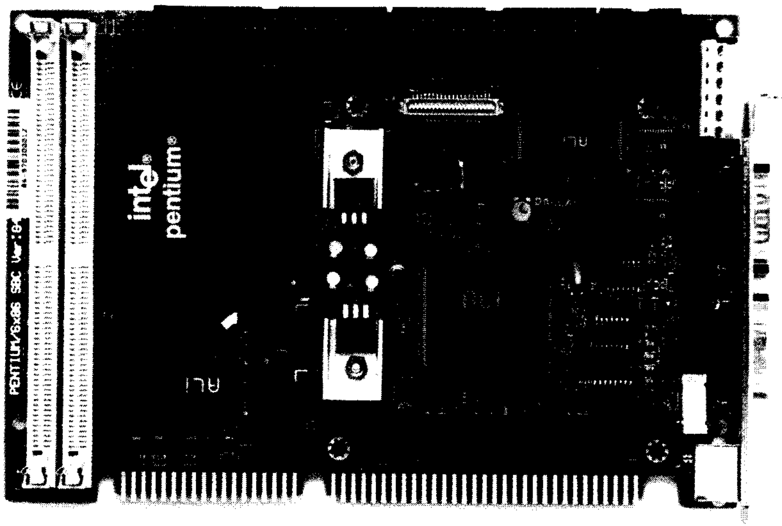
HALF-SIZE SINGLE-BOARD COMPUTER

The PC-560 is a half-sized, Pentium-based SBC that contains all the basic elements found in a standard IBM PC/AT-compatible desktop computer system plus some unique features making it ideally suited for industrial applications. It can be configured to operate without a display and keyboard, so it's a perfect solution for a wide variety of industrial applications.

This computer features a high-performance I/O controller, watchdog timer, PCI SVGA/flat-panel controller, and single 5-V power-supply operation. It also includes a PC/104-compatible interface port, so it can operate in a stand-alone configuration.

The PC-560 supports 75–200-MHz Pentium, AMD 5k86, and Cyrix 6x86 processors. Its 1 MB of video memory enables a resolution of 1024 x 768 at 256 colors or 1280 x 1024 at 16 colors. Standard I/O features include an SPP/EPP/ECP-compatible bidirectional parallel port, dual floppy-disk port, two high-performance 16C550 serial ports (COM2 can be configured for RS-232/-422/-485), and two PCI EIDE hard disk ports that support up to four drives.

The PC-560 is priced at $725.



Micro Computer Specialists, Inc.
25986 Fortune Way
Vista, CA 92083
(760) 598-2 177
Fax: (760) 598-2450
mcsi@mcsi 1 .com • www.mcsi 1 .com

#516

Tom Scott

Windows CE for Embedded Applications

Although Tom admits that Windows *CE* can't possibly fit every embedded application, he points out the niches it can fulfill. And, he demonstrates how it meets the marketplace demand for more standardization between products.

Windows CE, Microsoft's latest addition to the Windows family of OSs, was unveiled in late 1996. Although it only ran on one type of device-the hand-held PC—each device proudly touted its presence by displaying the Windows CE logo and running familiar-looking desktop applications.

It's certainly a technical feat to develop an OS that can run scaleddown versions of Microsoft Excel, Word, and Internet Explorer in 4 MB of ROM and 2 MB of RAM. But, you're probably wondering what this has to do with general embedded systems.

Windows CE will be seen initially on portable and mobile applications. But, its positioning, pricing, and footprintwill have an impact on the entire embedded market. Microsoft has ratcheted deeper into the embedded market, delivering a platform for the industrial automation and communications markets as well as the price-conscious medical and games markets.

Is this the end of the proprietary RTOS market? Well, not this year, and certainly not for all applications.

However, it delivers a serious and continuing blow as the embedded market must recognize that the islands of functionality are disappearing and connected applications are the future.

Time-to-market pressures are increasing for embedded systems, and the use of a commercial OS has huge advantages in leveraging off-the-shelf components. The use of a commercial embedded OS with ties to the desktop has the opportunity to leverage even more.

The leveraging argument applies to know-how as well as code, and Microsoft is quick to pointoutthe more than 500,000 Win32 developers. Microsoft wants to reduce the technical barriers that prevent Win32 developers from applying their expertise to Windows CE.

Existing commercial OSs in the embedded arena have also felt market pressure to provide support for the Win32 API. Many RTOS vendors have responded with an emulation layer. But, the staying power of thisapproach isquestionable. The full Win32 API is expanding all the time, adding improvements to the development environment.

In this article, I describe some Windows CE features to help you evaluate its suitability for your next embedded project. I also discuss the real-time functions and needed enhancements expected to be available. I hope you'll gain insight into how Windows CE device drivers are developed and, most importantly, how to get started.

IN EMBEDDED PROJECTS

The embedded market is large and diverse, and it's not unusual for an OS to target specific areas. So, the first question is whether Windows CE addresses the marketsector appropriate **for** your next project.

Windows CE has many features to choose from-especially in communication, development environment, and user interaction. Table 1 lists some of the more important **features** announced for this **year.**

Microsoft expects to provide two Windows CE releases per year and is working with OEMs to develop Windows CE for

Kernel	Fixed priority scheduling, power management, PC Card services, Win32 process/thread and virtual memory model ROM execute in place, demand paging from compressed ROM
Communication	PPP TCP/IP, IrDA, TAPI, Unimodem, Winsock, Remote Access API, LAN support (NDIS drivers and SMB redirector)
USER and GDI	Overlapping windows, event management, user interface controls, dialog boxes, interprocess communication, UNICODE, 24-bit color
Development	ActiveX subset, Java, Visual Basic subset, MFC

Table 1—As well as** having the features expected of an embedded kernel, Windows **CE** 2.0 excels in the areas of **communications, user interface, and development environment.

new types of devices. Third parties will also provide new components to be added to the basic kernel configuration.

As you'd **expect** from an embedded OS, Windows CE is broken down into numerous components that can be configured into a system. The selected componentsaffect RAM/ ROM requirements as well as royalties.

The memory requirements of a minimally configured Windows CE depend on who **you** talk to. Configurations as small as 100-KB RAM and 150-KB ROM have reportedly been built.

A more typical configuration supporting the registry-RAM/ROM file system, kernel debugger, a simple nonGUI application, and device driver-can require 250 KB of RAM and 500 KB of ROM. A full configuration, as you might find in a handheld PC, may need 2 MB of RAM and 4 MB of ROM (including applications).

Processor support limits Windows CE to new 32-bit designs. Although Windows CE runs on more processors than any other Microsoft OS, it supports a fraction of the processors used in embedded systems. Fortunately, support has been announced for many of the next-generation 32-bit processors likely to be in embedded systems.

But, the low power and high integration of the many processors supported by Windows CE seem to reflect its initial emphasis on mobile computing. I'm hoping this situation will change as vendors realize the applicability of Windows CE for general embedded computing.

Architecturally, Windows CE only runs on processors that are 32-bit Little Endian with an MMU. The MMU gives Windows CE the opportunity to contain application faults, but it limits its use to the higher end 32-bit processor families. It's conceivable that Windows CE could influence the design of future processors.

Getting Windows CE support for a currently unsupported processor is something beyond most project developers. Full source code isn't available, and processor ports are performed **by Microsoft** on behalf of their semiconductor partners.

Becausesourceisn'tavailable, Windows CE can't be used in highly critical control applications. For example, if you design flightcontrolsystems, WindowsCE isn'tgood. For such systems, simplicity is a virtue. You need source to verify functionality.

Combining a simple real-time executive, which does have source **available,** with Windows CE may address some of these concerns. VenturCom's RTX architecture follows this approach and may take Windows CE into these areas.

HARD REAL TIME

Many embedded applications need hard real-time performance in addition to ROMability. Windows CE has **many of** the features you'd expect from an RTOS.

Scheduling is fixed **priority with** priority-inheritance support in its locking protocol. Interrupt latency is kept low by offloading most interrupt processing to interrupt service threads. Multiprocessing and multithreading are supported as well.

The limit of 32 processes with 32 MB of virtual address space may be a problem for some real-time use. But, there's no limit on the number of threads, and many RTOSs today only support a single-process model.

There are some gaps, however, with complex real-time systems. Although it supports fixed priority scheduling and priority inheritance, Windows CE only has seven priority levels. The highesttwo priority levels are reserved for interrupt service threads.

For systems that share the processor among a modest number of tasks and interrupt sources, this limited scheduling structure may suffice. But for many real-time applications, there aren't enough priority levels to adequately manage system activity.

Paging may also be problematic. To conserve RAM usage, Windows CE compresses code and data stored in its RAM/ ROM file system. Compressed code pages are faulted into RAM, where they're decompressed before execution. For determinism, de vice drivers aren't paged, but pages for other user processes can be, affecting determinism.

Many real-timeapplicationsneed high-resolution timer services. Windows CE doesn't implement Win32 waitable timers, so the Win32 API limits the time specification to 1-ms resolution. Implementation of the API is often with a clock period ten times larger. In stark contrast, real-time applications can require 1 00-µs resolution.

VenturCom will address these and other real-time limitations with its RTX architecture, originally developed for Windows NT (see Naren Nachiappan's "Embedded PCs Go Industrial," INK75 and 77). RTX defines a real-time API (RTAPI) specifically for hard real-time use within a Win32 environment and provides a real-time subsystem (RTSS) that deterministically implements this API.

RTAPI defines these interfaces in addition to those provided by the Win32 API:

- high-resolution clocks and timers
- fixed-priority scheduling with 128 priorities provided by the RTSS
- I/O bus and physical memory access from user processes
- device interrupts and interrupt priority setting
- interprocess communication with semaphores, messages, and shared memory

RTX distinguishes between processes using only RTAPI calls and those mixing RTAPI and Win32 calls. RTAPI-only processes run in the RTSS, have deterministic response

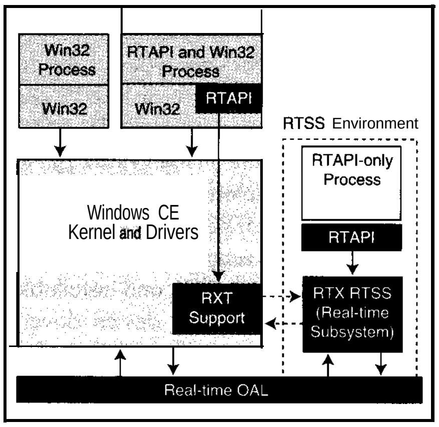


Figure ***1—This** diagram depicts **the various modules of VenturCom's hard real-time extensions (RTX) for Windows CE and their interrelationships. RTAPI is VenturCom's real-time API.***

with 128 priorities, and communicate with other system processes through semaphores, messages, and shared memory. Processes mixing Win32 and RTAPI calls have the scheduling limitations of WinCE.

The basic architecture of RTX shown in Figure 1 consists of:

- real-time OAL—lets hardware interrupts be vectored to RTX before being passed on to Windows CE
- RTX RTSS-performs scheduling and other supportfunctionsfor RTAPI-only processes
- RTX support module-provides support from Windows CE for resources managed by Windows CE
- RTAPI DLLs—implement the real-time API for use with Win32 calls

Using the same real-time architecture for Windows CE and NT provides interesting potential for upward expansion. Real-time NT applications can move to CE to take advantage of more deeply embedded devices. Conversely, applications that outgrow the resource limitations of Windows CE devices can move to NT.

DEVICE DRIVERS

Usually, embedded applications largely consist of custom software. Developing Win32 software that interfaces with the user, other processes, or device drivers is well-documented.

Other than Windows CE supporting a subset of the Win32 API, the approach for Windows CE is the same as for Windows NT. Developing device drivers is different, but it's probably already familiar to most.

Unlike NT, Windows CE doesn't have an elaborate **API** forwriting device drivers. CE device drivers consist of Win32 DLLs and user threads (see Listing 1). By relying as much as possible on the Win32 API for the device-driver interface as well as application code, RAM/ROM is conserved.

Windows CE supports two categories of device drivers-built in and installable. I discuss only installable device drivers, since they have the most flexibility to adapt to new types of devices.

The basic interface between the CE kernel and installable device drivers is the stream interface. It specifies the entry points provided by the DLL implementing an installable device driver (see Table 2).

The XXX in the table is unique to each device driver and associates the device name with the driver's entry points. These entry points are invoked by the CE kernel in response to system management and to Win32 applications accessing the device.

Device drivers are accessible to Win32 API through a filename convention. The Windows CE file system recognizes a filename consisting of three uppercase letters, a digit, and a colon as a device name. The letters correspond to the Xxx prefix used in the stream interface.

A device driver can be called to manage a number of devices it controls. The digit is a device index the device driver uses to distinguish among these devices.

Windows CE uses a combination of the device name and preset registry values to associate an entry point in a DLL with a Win32 call. If the Win32 CreateFile() call specifies COM1: as the file to open, for example, the kernel calls COM_Open().

Based on information provided in the registry, the kernel invokes this entry point from Seria1.dll. Registry entries for device drivers are set up as part of Windows CE configuration.

In Windows CE, user threads process interrupts. The *XXX_*Init() interface initializes device drivers. An important responsibility of this routine is to create the Interrupt Service Thread (IST) using the Win32 CreateThread() and setting the thread's priority via SetThreadPriority().

The driver is also responsible for creating an event with CreateEvent() and registering it with the kernel by a call to InterruptInitialize(). Once the event is registered, the kernel signals the registered event every time the interrupt occurs. The IST merely repetitively waits for the registered event to occur before processing the interrupt.

Windows CE is multithreaded, so it's necessary to synchronize the access of threads to shared data structures. Drivers use Win32 critical sections for this. For event notification (e.g., arrival of data from a device), drivers use Win32 events.

Windows CE goes far with the Win32 API, but additions are needed. Windows CE adds a few APIs for use by interrupt service routines (ISRs), such as Interrupt-Initialize() and InterruptDone().

There's also a routine to disable interrupts, but since interrupt processing is performed by user threads, it's seldom used. Windows CE adds an API so device registers can be mapped into the address space

listing I--This code fragment demonstrates how the Win32 API is used by Windows CE device drivers. Win32 interfaces are used for synchronization, ***interrupt-service*** thread creation, and event notification.

```
HANDLE hIREQEvent;                // Event associated with interrupt
HANDLE hInterruptThread;
CRITICAL_SECTION hCriticalSection; // Shared data synchronization
DWORD
FOO_Init(DWORD dwContext){        // Initialize the device
  hIREQEvent = CreateEvent( NULL, FALSE, FALSE, NULL);
  InitializeCriticalSection(&hCriticalSection);
  // Code to map device registers goes here
  hThread = CreateThread(0,       // Security Attributes
    0,                            // Stack Size
    (LPTHREAD_START_ROUTINE) FOO_InteruptThread,
    0,                            // Thread Parameters
    0,                            // Creation Flags
    NULL
    );
  SetThreadPriority(hThread,THREAD_PRIORITY HIGHEST);
  // Associate event with SYSINTR_FOO interrupt
  InterruptInitialize(SYSINTR_FOO, hIRQEvent, NULL, 0);
  return 1;}                      // Return success
ULONG                             // Interrupt service thread
FOO_InterruptThread(VOID)
{
  while (TRUE) {
    WaitForSingleObject(hIQEvent, INFINITE); // Wait for interrupt
    EnterCriticalSection(&hCriticalSection);
    // Device dependent code to dismiss hardware interrupt goes here
    InterruptDone(SYSINTR_FOO); // Unmask this interrupt
    // Code to process interrupt goes here
    LeaveCriticalSection(&hCriticalSection); }
  return TRUE;
}
```

of the process accessing the device.

There isn't much in the way of hardware abstraction. Abstraction (e.g., busabstraction) imposes overhead, and for custom hardware devices, the designer knows the exact configuration that has to be supported.

For high-volume devices, it pays to optimize out any unused generalization from the hardware. For lower volume systems (e.g., thoseaddressed by board-level computers), bus abstractions have a place and may need to be introduced into Windows CE.

Interface	Invoked by
XXX-Close	CloseHandle to close device (Indirect)
XXX_Deinit	Device Manager to deinitialize device after last use
XXX_Init	Device Manager to initialize device for first use
XXX_IOControl	DeviceIOControl (and by Device Manager) to send command to device (Indirect)
XXX_Open	CreateFile to open device for reading and/or writing (Indirect)
XXX_PowerDown	OS to power down device
XXX_PowerUp	OS to restore power to device
XXX_Read	ReadFile to read data from device (Indirect)
XXX_Seek	SetFilePointer to move data pointer in device (Indirect)
XXX_Write	WriteFile to write data to device (Indirect)

Table **2—Installable** device drivers, a type of Windows ***CE*** driver with a flexible programming mode/, are ***DLLs that implement*** the entry points of ***the*** Stream Interface. The XXX is a three-character prefix ***that's*** unique ***to each driver in a system.***

Windows CE abstracts the identification and manipulation of interrupts listed in Table 3. The OEM Adaptation Layer (OAL) is responsible for identifying interrupt sources for the kernel and implementing routines that let the kernel mask and unmask these interrupt sources.

Each source gets an ID as part of porting Windows CE to a platform. The IDs for many standard interruptsources (e.g., SYSINTR_KEYBOARD) are predefined, and OEMs define others to support their platforms.

Early in Windows CE initialization, the kernel calls OEMInit(), which registers ISRs by calling the Windows CE-provided HookInterrupt(). ISRs aren't meant to do much processing in Windows CE. Typically, they merely identify the source of the interrupt and return the interrupt ID.

Drivers associate events with interrupt IDs via InterruptInitializeO. On the return of an ISR, the kernel signals the event associated with the interrupt ID returned by the ISR. When an interrupt occurs, only the applied interrupt is masked-all other interrupts remain unmasked. The applied interrupt remains masked until the IST calls InterruptDone().

Though hardware abstraction is minimal, there are mechanisms for code reuse. Reuse is obviously important for reducing development time and code size. Many Windows CE drivers are split into a platform-independent part--the Model Device Driver (MDD)-and a platform-specific part--the Platform Device Driver (PDD).

The MDD implements the behavior expected from a class of device (e.g., a serial port). It interfaces to the kernel using the stream interface and exposes a model-specific API to be implemented for each platform Windows CE is ported to. Windows CE provides a number of MDDs in the OAK (more about that later). The PDD implements the hardware details.

GETTING STARTED

What you need to get started depends on whether you're writing user applications for an existing device, creating drivers for new peripherals, or developing a new Windows CE-based system.

To start developing Win32 user applications for Windows CE, you need an 'x86 PC running Windows NT, Visual C++ 5.0, and Visual C++ for Windows CE. This last item provides the cross-development tools for all supported CE platforms.

You may not even need the target hardware to get started. The tools include emulation environments for hand-held PCs and other Windows CE devices Microsoft has developed hardware specifications for.

Given its wide availability, VC++ for Windows CE is the best place to look for answers regarding the subset of Win32 available in Windows CE. For the most part, the Win32 API available on Windows CE is a subset of that available on Windows NT, but there are some differences in function noted in the online documentation.

If you're migrating an application from Windows NT to CE. VC++ for Windows CE is the place to start.

Cross-development tools support download and debug through a serial/network connection to the target device. Full source-code debugging is supported. With some care, applicationscan be built to run natively on Windows NT as well as emulated and actual Windows CE platforms.

For serious system development, you need the OEM Adaptation Kit (OAK), which enables developers to configure and build kernels. It includes Windows CE binaries and libraries for each of the supported processors and sources for the OEM Abstraction Layer (OAL) and drivers.

The OAL abstracts the hardware platform for Windows CE, similar to the Hardware Abstraction Layer of Windows NT. Sources to the MDD and PDD drivers are provided, usually for the devices that are present on the Windows CE hardware reference platforms provided by semiconductor manufacturers.

The OAK not only builds kernels but also a development environment for applications. The Win32 API depends to a small extent on which components have been configured into the kernel.

As a side effect of building a kernel, headers and libraries corresponding to the configured components are built, conserving target resources but requiring application developers to be aware of the specific device it's being developed for. This situation is unavoidable for embedded systems.

And, there are other issues you need to be aware of. The infrastructure for developing systems using Windows CE on the small scale is undergoing growing pains. The early hardware reference platforms for Windows CE are very much reference platforms for hand-held PC designs.

To date, no board-level Windows CE products are available, putting Windows CE development out of the reach of many small-scale designs since custom hardware

OEMInit	Perform platform initialization (initialize interrupt vectors, etc.)
OEMInterruptDisable	Mask the specified interrupt
OEMInterruptDone	Called indirectly by interrupt, done to unmask specified interrupt
OEMInterruptEnable	Unmask the specified interrupt

Tab/e ***3—Windows*** CE has an OEM abstraction layer ***(OAL)*** adapted ***to*** each platform. Here are some ***OAL*** routines that must be implemented for each ***platform.***

must be built. Don't expect this situation to last long, however, as Compact-PCI and PC/104+ system vendors eye Windows CE for new opportunities.

The introduction of board-level Windows CE products will necessitate changes in the existing development model. Projects that don't have thevolumes to justify custom hardware also need an easy way to configure kernels. In this case, VenturCom's Component Integrator (CI) will have wide applicability for Windows CE.

CI is a GUI-based tool developed for componentizing Windows NT. It runs on an 'x86 workstation running Windows NT, the same host system as the Windows CE development environment, and can build highly customized Windows NT kernels.

The heart of CI is a knowledge base describing the files, registry entries, and dependenciesassociated with the hundreds of components of Windows NT. VenturCom will be extending this to Windows CE components to enable CI to build Windows CE as well as Windows NT targets for use in embedded systems.

WINDOW TO THE FUTURE

With Windows CE, Microsoft has provided a base technology targeted at the embedded-systems market and traditional RTOSs. Unlike other Win32 API desktop OSs in its arsenal, Microsoft has tuned the Win32 API supported by Windows CE to specifically address the small footprint needed for the lower end of the embedded market.

There will be growing pains, but Windows CE has the potential to dominate this space. It will be independent value-added resellers making the refinements needed to compete across all the vertical markets, but they're lining up rapidly.

took for the emergence of board-level Windows CE systems and the associated changes in Windows CE. This shift will put Windows CE within the grasp of system integrators, whose volumes don't justify the expense of developing new hardware. Though volumes of individual designs are low by consumer electronics standards, it can be argued that this is a lion's share of the embedded market.

And watch the boundary between Windows CE and NT. Together, these two OSs are a formidable force in the market, leaving the RTOSs room below and on standalone applications only. EPC

Tom Scott, VenturCom's vice president of consulting services and special projects, is responsible for bidding and managing opportunitydriven engineering projects that demonstrate and use VenturCom's embedded and real-time application development tools and *extensions for Windows NT and Windows Cf. You may reach Tom at tscott@vci.com.*

SOURCES

Windows CE
Microsoft Corp.
One Microsoft Way
Redmond, WA 98052
(206) 882-8080
Fox: (206) 936-7329
www.microsoft.com

RTX, Component Integrator
VenturCom
2 15 First St.
Cambridge, MA 02 142
(617) 661.1230
Fax: (617) 577-I 605
www.vci.com

I R S

4 10 Very Useful
41 1 Moderately Useful
412 Not Useful

PC/104

Industrial I/O

REMOTE DATA ACQUISITION

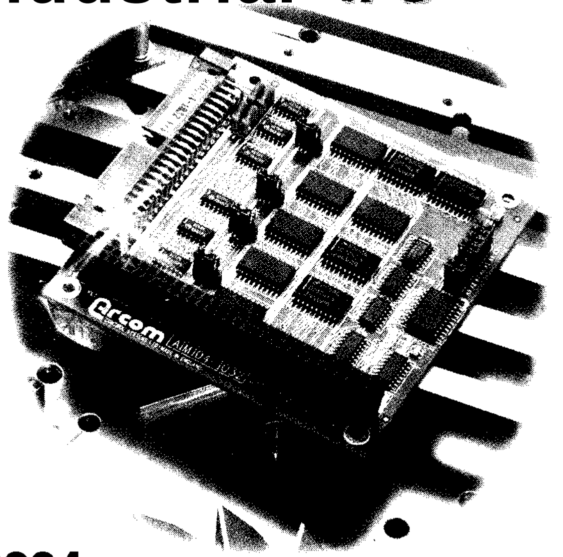
INDUSTRIAL NETWORKING

MACHINE CONTROL

Arcom's new **range** of **PC/104** modules **are** designed for embedded machine control and even includes on-board **isolation & relays, so that you can interface directly** to your **plant** and equipment **signals...**

Arcom's *wide* ***range*** of ***PC/104*** modules ***Includes:***

- **Digital I/O**
- **Multi-ADC/DAC**
- Multi-relay
- **CANbus - DeviceNET**
- **Serial** ports
- **Ethernet**
- VGA



Toll Free: 888-941-2224

Fax: **816-941-7807** Fax-on-Demand: **800-747-1097 E-mail:sales@arcomcontrols.com**

arcom CONTROL SYSTEMS
www.arcomcontrols.com

#122

Edward Steinfeld

Windows CE is Ready, But for What?

The embedded market has been traditionally commandeered by those producing tight code, integration, compact real estate, and low cost. However, Microsoft wants to enter the picture. Edward questions whether they can adjust.

Microsoft's launch of the Windows CE operating system for embedded systems raises the question, "How does Microsoft define the embedded-systems market?"

True embedded PCs are generally characterized as singlepurpose computers. Even if end users know a PC-compatible computer is embedded in the device, they don't interact with the device as a desktop computer.

However, Microsoft only plans to build Windows CE into devices which are merely smallerversionsofdesktop PCs (e.g., palmtops and laptops).

As well, in the traditional real-time embedded computing market, most electronic-equipment manufacturers make products in the range of 300-3000 **per year.** Microsoftsellsdirecttocompanieswhosevolume is 25,000 units a year or greater. Anything less goes through a distributor [1].

Only in consumer products do we see such high levels of usage. Is Microsoft ready to take on the additional supportand lower volume that are inevitable parts of the embedded-PC market?

WinCE AS AN EMBEDDED KERNEL

The announcement of Windows CE as a platform for real-time embedded applications has both good and bad aspects.

The good part is the publicity Win32 will receive. Phar tap Software has been supporting real-time Win32, first in the TNT DOS-Extender and then in the ETS Kernel. It was, in fact, the first to support the Win32 API native with an embedded kernel.

Today, QNX, Wind River, and other vendors have added a Win32 API layer to their embedded kernels-the APIAccess Win32 layer (developed by Willows, but now **owned by** Award Software). And, Ventur-Corn is shipping a real-time kernel using the Win32 API as a layer for Windows NT.

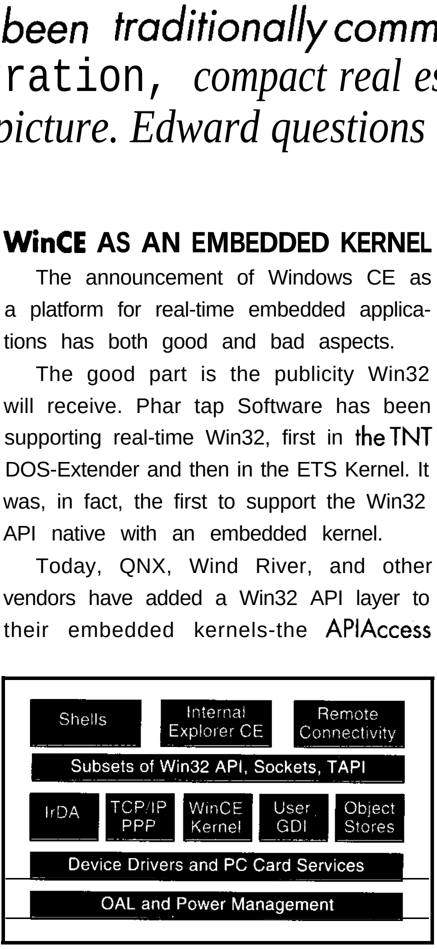


Figure I-Any *of **the** elements in the Windows **CE** architecture can be replaced or removed for embedded systems.*

Support is growing for real-time embedded applications development using Visual C++ and Borland C++ compilers. Windows CE will accelerate that movement. For **years,** using common compilers and a single API has been a dream in embedded computing.

On the down side is Windows CE's lack of "embeddedness" and true real-time functionality [2]. Windows CE suits small portable computer products, but it falls apart as a blood analyzer, robot, machine tool, or other intelligent machine.

Itsstrength lies in its WinNT/95 look and feel. But as an embedded kernel, it lacks the pluses of real-time kernels like Wind River's VxWorks or Phar tap's Real-time ETS Kernel.

Compared to Windows NT, WinCE is small-less than 1 MB. A typical kernel should use 500-KB ROM and 100-KB RAM [1]. However, it's also a multiprocess system with the inherited overhead of Windows 95.

Microsoft is targeting portable computing and automotive applications—'places where the product being sold is a computer. Given the Windows look and feel, ease of development, and general acceptance of Windows, it should do well in these markets.

MISSING FEATURES

For WinCE, **Microsoft supports** a subset of the Win32 API (about 500 functions). No new APIs were added, according to R. Kanemori, Microsoft Senior OEM Account Manager. By contrast, Phar lap Software uses 140 of the Win32 APIs in its product and felt the API set needed another 140 functions for embedded and real-time applications.

Without added APIs, how does Microsoft expect you to initialize a chipset if there's no BIOS? How can you set an event flag in an ISR, or track the thread execution sequence in a complex system? How do you change the system-clock frequency or time slice?

With Windows CE, you can't do any of these using a high-level language and ready-made functions. Both the Win32 API and Windows CE kernel need these additions to properly support real-time and embedded applications.

DEVELOPMENT TOOLS

Development requires Visual C++ for Windows CE—a special crossdevelopment version of Visual C++. In addition, OEMs must purchase the OEM Adaptation Kit (OAK) containing the Windows CE code.

The OAK is free to OEMs when they purchase a sufficient quantity of CE kernel licenses. According to one customer, cost is $70 per 1000. It's intended to let the OEM write or modify the OEM Abstraction Layer (OAL) built between the hardware and the rest of the OS (sound like Windows NT?).

According to Annabooks, a Windows CE kernel distributor, you also need Microsoft's device-driver development kit (DDK) [3]. Eclipse, another CE distributor, recommends their Eclipse development system for debugging and execution [4].

To emulate the target environment during development, you have to develop on a Windows NT system. A set of special development DLLs is provided with Visual C++ for Windows CE for this purpose.

The target requires built-in networking to debug via a serial line supporting UDP or TCP/IP PPP. According to Microsoft, the UDP transfer method isn't always reliable. The chip or board vendor will probably have to provide this support. Today (prior to 2.0), TCP/IP has no general network support,

OTHER PROBLEMS

And, there are other drawbacks. For one, the system supports Unicode only.

Unicode is the 16-bit representation of characters that stores and displays Latin and Asian character sets. This feature is important for local language displays. But, most intelligent machines have minimal displays that don't require such a feature.

For the OEM, Unicode has other prob lems besides its double memory space. If an application needs to parse a string or search a text array, the software must be rewritten to accommodate this wide-character storage. It will be difficult to rewrite existing C applications to accommodate Unicode.

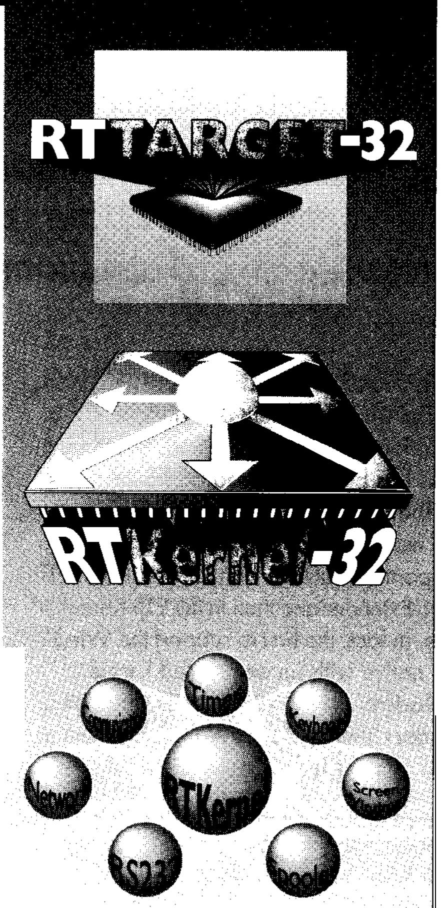
Priority scheduling in Windows CE is different from both Windows NT and Windows 95. Unlike Windows NT's four **priority classes,** there's just one class of priorities in the CE kernel and only eight levels (O-7). **Microsoft indicates** that there are only seven levels (O-7, minus level 6) [5].

Professional Development Tools for x86 Embedded Systems

RTTarget-32
Cross development system
for 32-bit embedded systems.
Boot code, locator, cross debugger.
Supports Intel 386 and higher, as little as 16kRAM/ROM.
For Borland C/C++, Microsoft C/C++, Watcom C/C++, and Delphi.
Windows NT compatible API!

RTKernel-32
Professional, high-performance
real-time multitasking kernel
for RTTarget-32.
Supports Intel 386 and higher.
For Borland C/C++, Microsoft C/C++, and Watcom C/C++.
Windows NT compatible API!

RTKernel
Professional, high-performance real-time multitasking kernel for DOS and 16-bit embedded systems.
For Borland C/C++, Microsoft C/C++, and Borland Pascal.
Runs under DOS and Paradigm Tools!



No run-time royalties!
Evaluation Kit available!

http://www.on-time.com

North America:
On Time
88 Christian Avenue
Setauket, NY 11733, USA
Phone (516) 689-6654
Fax (516) 689-1172
email info@on-time.com

International:
On Time Informatik GmbH
Hofweg 49
22085 Hamburg, GERMANY
Phone +49-40-2279405
Fax +49-40-2279263
email info@on-time.de

On Time
REAL-TIME AND SYSTEM SOFTWARE

Windows CE V. 1.01 is out now. In the second quarter of 1997, V.2.0 was scheduled for beta testing, with final versions shipping by the end of September 1997.

The old hat that Microsoft V.3.0 is the first useful version may hold true here as well. Even Microsoft says, "...the 2.0 version may be more appropriate for some embedded systems designers than 1.01" [6]. V.2.0 will be the first to support the '486 and PowerPC.

WHAT'S MISSING?

The Windows CE system supports a subset of the Win32 API. Nothing new has been added. And, many useful APIs or functions are lacking.

First of all, there's no way to initialize the CPU and support chips other then the few odd chips currently supported (i.e., the chip manufacturer or OEM must write their own BIOS). Developers of custom boards using chips other than the reference designs have no easy way to initialize their chipset.

You can't set events or semaphores from an ISR. This feature would allow for short and fast ISRs, permitting the kernel to start threads waiting for events produced by ISRs.

There's no API to suspend the scheduler, so the application can't raise itself above the kernel to execute a critical function without the possibility of the kernel resuming and passing control to another thread. But, without the ability to set events and semaphores from ISRs, this is of less importance.

In both Windows 95 and NT, the kernel can take control over even the highest-priority thread-not a feature you want in a critical application. VenturCom plans to include scheduler suspension in its real-time Windows CE kernel.

Windows CE also doesn't include an API for priority-inversion avoidance (a quirk resolved in every serious real-time kernel). Will Microsoft solve this problem for WinCE like it did in both Windows NT and Windows 95 kernels [7]?

The Windows 95 kernel senses when a higher priority thread is blocked frpm access to a critical section by a lower priority threqd blocked from running by a mid-level priority thread. The kernel temporarily raises the priority of the lower priority thread to let it complete its operation and release the critical section. VenturCom plans to have priority-inheritance avoidance in its version of the Windows CE kernel.

Changing the system clock's frequency is also not supported. This ability lets the kernel have control more often, and increasing the frequency enables finer control of functions.

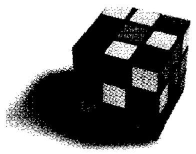
There also isn't an API for changing the length of the round-robin time slice. In both Windows NT and Windows 95 kernels, threads executing at the same priority enter into a round-robin scheduler.

This setup gives each thread CPU execution time. But since I can't find documentation for a round-robin scheduler, I doubt there's a way or need to change the time slice.

WinCE does not support sequence-of-events logging, either, which offers a sequence list of thread execution and the parameters they pass as the kernel activates them or they call kernel functions.

This debugging tool is useful when many threads have different priorities. A sequence-of-events logging mechanism is the only way to find the sequence that a set of threads actually executed.

Direct access to I/O registers requires kernel-level drivers to support new or different hardware. Hence, there are lengthy context switch times when the application accesses I/O.



Par•a•digm: your source for the most high-powered, comprehensive set, of time-saving software and hardware development tools for embedded application development.

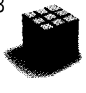
1: **Paradigm** LOCATE the most popular tool for creating embedded C/C++ applications with Borland and Microsoft compilers; 2: **Paradigm DEBUG** the only x86 debugger with C++, RTOS, scripting language, and full in-circuit emulator support; 3: **Paradigm SUPPORT** the best, technical support, in the industry supplied to our customers for free.

Developing real-time embedded applications doesn't have to be time consuming or difficult-you just need to have the right tools. Paradigm alone has the high performance development tools you need to streamline the embedded system software development process so your Intel and AMD x86 applications are ready in record time. Paradigm's complete suite of tools work with industry standard C/C++ compilers from Borland and Microsoft, as well as hardware development tools from Applied Microsystems, Beacon Development Tools and other popular in-circuit emulator vendors.

Call us at 800-537-5043 today and let us take care of all your development tool needs, so you can keep your focus where you need it-on your application.

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



#123

Microsoft says thread-switch times are less than 1 00 µs (about four times slower) than most real-timeembedded kernels. VxWorks and others use a single process (single address space, everyone resides in ring 0) system, making context switches easy and fast.

Application-level drivers don't require a rigid driver format. Windows CE has a registry system similar to Windows 95, where applications can be registered with the kernel and receive interruptsdirectly, butthat's not the same as getting maximum throughput via direct access to I/O registers.

As well, WinCE does not support a full-function embedded Web server. There isn't enough RAM or ROM for complete TCP/IP support and a Web server, but networking is expected to be available in V.2.0.

The only announced Web server for the product has no local file access, so it can't send HTML pages stored in the system. The kernel needs to support in-memory files to emulate a Web server's function. (See Phar tap's "world's smallest Web server.")

The universal GUI-the Web browser-is used in standard CE systems to communicate as Web PCs with intelligent embedded systems. Maybe Spyglass's WWWAccess will rescue Microsoft here, too.

However, because of the modular architecture of Windows CE (see Figure 1), most of these missing functions can be added or existing functions modified.

Windows CE has so much overhead that Microsoft won't support its kernel on '386 CPUs. But, the Intel '386 EX and RadiSys EXPLR2 board are popular for developers.

By the end of September, we should have seen support for generic '486s (if virtual memory is supported), Pentium, and PowerPC chips. But, the **only chips** supported now are the Hitachi SH-3, NEC Vr4100 series, and Philips PR3 1500 (MIPS R3000 RISC core) and UCB1 1 **00-not** exactly mainstream with embedded-system OEMs.

IT FITS SOME APPLICATIONS

Portable computer and automobile applications are Microsoft's primary targets. And while Microsoft also mentions some veryembeddedapplications (e.g., temperature and robot controllers), this seems more like a pipe dream than reality.

The WinCE kernel is a multiprocess OS, and even if you develop for a single process, the application and kernel are separate processes, entailing much higher overhead on system calls and I/O accesses.

If your product is not cost sensitive, requires a graphical local display (especially one looking like Windows NT/95, as you see in Photo 1), or is sold as a portable computer, using WinCE as an embedded system makes sense. For the majority of embedded computing products, however, it's either overkill or it won't perform real-time tasks with any determinism.

I believe the best outcome of Microsoft's entry of Windows CE in the embedded space will be the tools developed for the 32-bit 'x86 market.

Visual Basic and Visual C++ support in multiplechip platforms, improvementofthe Win32 API, and the promotion of a standard API (Win32 API) for embedded systems will help solidify that sector of the embedded computing market using 32-bit 'x86 platforms. EPC

Edward Steinfeld ***over** 25 **years' experience in real-time and embedded computing. He has marketed embedded and real-time products to OEMs and resellers for DEC, VenturCom, and Phar lap Software. He now heads his own company, Automata International Marketing. You may reach Edward at stein@ma.ultranet.com.***

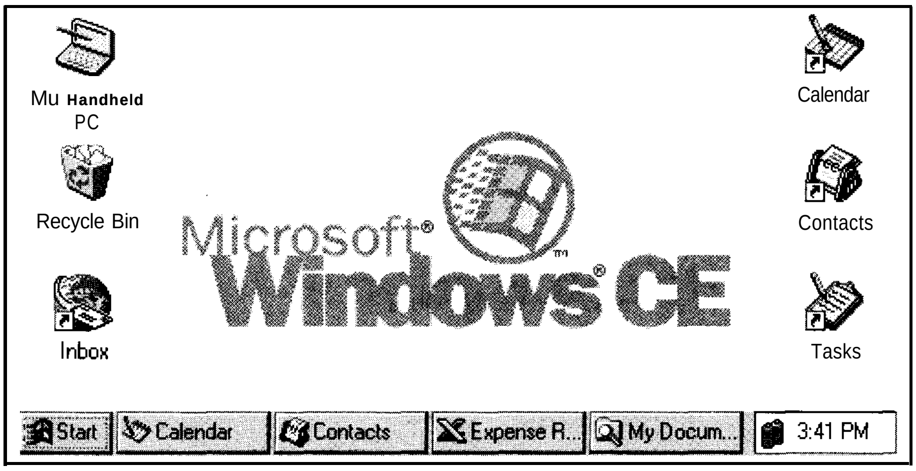


photo ***1—This*** splash screen of a Windows *CE system* shows you the available applications.

REFERENCES

[1] Microsoft-sponsored Embedded PC Solutions Seminar Series, Boston, MA, June 24, 1997.
[2] M. Timmerman and J.-C. Monfret, "Windows NT as Real-Time OS?" Real *Time Magazine,* www.realtimeinfo.be/encyc/magazine/97q2/winntasrtos.htm.
[3] Annasoft Systems, The *Annasoft* Guide to *Systems that Work,* San Diego, CA, 1997.
[4] T. Wong, "Enabling the Windows CE Tsunami," *Microsoft Embedded Rev.,* p. 13, March 10, 1997.
[5] N. Fishman And J. Richter, "The Windows CE SDK: The Tools You Need to Program the Hand-held PC," Microsoft Systems Journal, 17–28, April 1997.
[6] F. Fite, Jr., "Windows CE: The New Choice for Dedicated Systems," *Microsoft Embedded Rev.,* p. 29, March 10, 1997.
[7] www.microsoft.com/msdn/sdk/platforms/doc/sdk/win32/sys/ src/prothred_10.htm.

SOURCES

APIAccess
Award Software Intl.
777 E. Middlefield Rd.
Mountain View, CA 94043
(4 15) 9684433
Fax: (4 15) 526-2392
www.award.com

Windows CE
Microsoft Corp.
One Microsoft Way
Redmond, WA 98052
(206) 882-8080
Fax: (206) 936-7329
www.microsoft.com/windowsce/developers

Real-time ETS Kernel
Phar Lap Software
60 Aberdeen Ave.
Cambridge, MA 02138
(617) 661-1510
Fax: (617) 876-2972
BBS: (617) 661-l 009
www.pharlap.com
smallest.pharlap.com

WWWAccess
Spyglass, Inc.
1240 E. Diehl Rd.
Naperville, IL 60563
(630) 505-l 010
Fox: (630) 505.4944
www.spyglass.com

Windows CE tools and license
Annasoft Systems
11838 Bernardo Plaza Ct.. Ste. 3 13
San Diego, CA 92128
(6 19) 673-0870
Fax: (619) 673-l 432
www.annabooks.com

Eclipse Intl.
381 E. Evelyn Ave.
Mountain View, CA 94041
(415) 691-6555
Fax: (415) 428-0292
www.eclipseint.com

VenturCom
2 15 First St.
Cambridge, MA 02 142
(617) 661.1230
Fax: (617) 577-l 605
www.vci.com

IRS

413 Very Useful
414 Moderately Useful
4 15 Not Useful

Ingo Cyliax

Remote Internet Data Logging and Sensing

Why *not* build remote PC/ 7 04 Internet-based data loggers? *All* you need is one *32-bit* PC/*1* 04 CPU, a modem, an RTOS, application code, and some integration. After looking at what benefits this brings the market, *Ingo* shows how easily it's done.

In college, I worked as a part-time programmer for an electric-meter company. One of the projects was a translation system that converted data retrieved via modem from a remote electric meter into billing and usage reports.

The technology was exciting at the time. Before, utility companies hired meter personnel to read and swap magnetic tapes or memory modules containing the details needed for the reports. Handling such tasks remotely via the phone line was particularly appealing to utility companies operating in the north (brrr...).

Our translation system had to communicate with two meter styles-our remote metering product and the competitors'. Utilities rarely buy everything from onevendor, and since there were no **standards for** exchange of metering data, we had to implement two different protocols to talk to the meters.

Of course, our competitor wouldn't divulge their protocol specs. We had to obtain (don't ask) a competitor's unit and reverse engineer the protocol (finding the CRC polynomial was the hardest).

The system finally worked with both units. The experience was good, since I learned much about 8085s, modems, and daylight-saving time-zone calculations.

A lot has happened since then. I graduated, got married, had children, even lost some hair. But, more interesting to you is that the Internet exploded onto the scene.

Almost everywhere, you see URLs and E-mail addresses advertised. This sudden popularity has one important infrastructure impact. Internet service providers (ISPs) have a point of presence (POP) in almost every calling zone.

This new **situation** gives rise to some interesting possibilities. In particular, why not build remote PC/I 04 Internet-based data loggers?

I can embed the necessary drivers and software in a PC/ 104-based data logger so it dials up an ISP, connects to the Internet, and dumps data on an Internet server. It's what the Internet does best-transferring information regardless of the geography or content. Figure 1 shows the big picture.

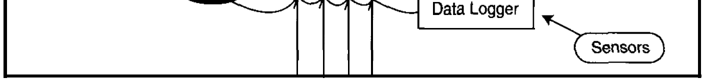
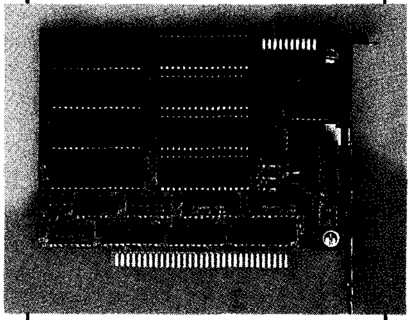


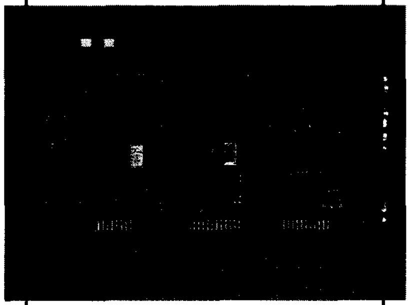
Figure I-Here's the big ***picture.*** The data logger uses a telephone line to dial up a ***local*** ISP. Since the ***ISP*** is connected to the Internet, data and ***files*** are *sent via the* Internet to a Web server, which can be anywhere on the Internet. The data can be retrieved with a standard Web browser or ***data-***analysis software for ***further*** processing.

VMAX®

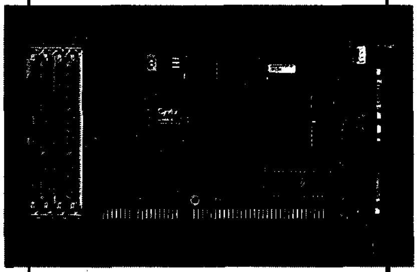
QUALITY PRODUCTS
RESPONSIBLE SERVICE
RELIABLE DELIVERY



ENHANCED SOLID STATE DRIVE — $89.00
5 Flavors: 2 or 3 Drives, 2M, 4M, 32M, 2M with Post LEDs
Either Drive Boots, FFS included
1/2 Card, XT Height, Customs too



MULTI PORT I/O BOARD — $90.00
4 Serial Ports, 2 Bi-Parallel Ports, 2 Drive IDE, 2 Floppy Interface, 16 Bit Standard, 8 Bit Adaptable, Low Profile 4.2", Full Cable Set



486 66MHz SINGLE CARD COMPUTER — $335"
Up to 2.5Meg Flash/Sram drive
Compact-XT height ½ card size
Industry Standard PC-104 port
L2 cache to 64K—DRAM to 16Meg
Dual IDE/Floppy connectors

All Tempustech VMAX® products are PC Bus Compatible. Made in the U.S.A., 30 Day Money Back Guarantee
*Qty 1, Qty breaks start at 5 pieces.

TEMPUSTECH, INC.
TEL: (800) 634-0701
FAX: (941) 643-4981
E-Mail: cpusales@tempustech.com
I-Net: www.tempustech.com

Fax for fast response! 295 Airport Road Naples, FL 34104

#124

listing I--This is a configuration file for a protocol stack module. The most critical information *in* it is the ***IP*** addresses for the name server ***(so*** we can look up ***host*** names) ond the default gateway.

```
[Winsock]
numsockets=50               ; Number of active TCP sockets
gatewayaddr=199.106.23.10;  Gateway IP address
bootp=no                    ; Use BOOTP protocol for IP addresses
dnsserveraddr=199.106.23.9; DNS server IP address
dnsretries=default          ; Number of DNS retries
dnsretrywait=default        ; Wait time for DNS retry
```

WHAT FOR?

Since ISPs have POPs almost everywhere, access is a just local call away. Sure, there's a monthly fee, but it's usually reasonableand inmanycasesoffersunlimited access. You can even share one account among many data loggers.

Such opportunity, ofcourse, frees us from maintaining our own central modem pool. We pay the ISP to do it at a fraction of the cost, since we're only using a small chunk of the ISP's bandwidth to the Internet. After all, we're interested in the infrastructure.

The Internet is very robust. The military designed it to be an easy-to-deploy robust technology. And even more amazing, it still works when many different vendors with their own (sometimes broken) implementations participate.

Even now that the communication channels for the backbones are hopelessly saturated, the Internet still functions-slowly, but it doesn't just crash all the time. Since my application doesn't require interactive response times, even these slowdowns due to congestion don't bother me.

The Internet is also technology and implementation independent. When the next Pentium MMX+ Turbo-based motherboardscome out, they'll be able to communicate with my MC68030-based workstation (see my "MC68030 Workstation" MicroSeries, INK

*Listing **2—This program** illustrates how to use the socket API. If opens a* connection to my Web server at ***<www.ezcomm.com>*** and *retrieves the* index page ***<index.html>.***

```
#include <windows.h>
#include <stdio.h>
char buf[256];
char data[256];
main() {
  int s;
  struct sockaddr_in sin;
  struct hostent *he;
  int i,n;
  WSADATA ws;
  if(WSAStartup(0x101,&ws)){
    printf("can't init WinSock\n");
    WSACleanup();
    exit(1); }
  if (!(he = gethostbyname("www.ezcomm.com"))){
    printf("can't get hostname\n");
    WSACleanup();
    exit(1); }
  s = socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
  if(s == INVALID_SOCKET){
    printf("can't open socket\n");
    WSACleanup();
    exit(1); }
  sin.sin_family = AF_INET;
  sin.sin_port = htons(80);
  memcpy(&(sin.sin_addr.s_addr),he->h_addr,sizeof(int));
  if(connect(s,(struct sockaddr *)&sin,sizeof(sin))){
    printf("can't open connection %d\n", WSAGetLastError());
    WSACleanup();
    exit(1); }
  sprintf(buf,"GET /index.html\n");
  send(s,buf,strlen(buf),0);
  while((n = recv(s,data,1,0)) == 1){
    putchar(data[0]); }
  closesocket(s);
  WSACleanup();
  exit(0); }
```

Listing 3—*Here's* the configumtion file for the Web upload/download plug-in. *All* the control and configuration files look like Windows *.INI files.*

```
[connection]
host=www.weathercentral.com    ; Name of host Web server
when=0:00,6:00,12:00,18:00     ; Call host every 6 h
retry=3                        ; Try up to 3 times to make a connection
retrydelay=0:10                ; Wait 10 min. between each retry
controlfile=/control/truro_ma/control.ini   ; Name of control file
                               ; for Truro. MA weather station
```

86, 87, and 88). All this makes me wonder what the oldest Internet node out there is. Send me E-mail if you know.

By basing data loggers on PC/104 cards, it's possible to build someverysmall and robust systems. Many PC/I 04 cards are available in extended temperature ranges. Also, with solid state disks (SSDs) available on most PC/I 04 CPU modules, mechanical disk drives can be eliminated. And since PC/104 systems are smaller and lighter, they can go places desktop PC-based systems can't.

Many applications can take advantage of this scheme. Seismological sensor networks, for example, can run on small PC-based embedded systems. With a GPS or WWV receiver (for accurate time), seismometer, and modem, the logger can log time-correlated sensor data to the SSD and dial up when there's an event.

Since they can be so small, remote sensors can live anywhere. They can also share existing telephone lines with other devices (or people). No land line available? No problem, call an ISP via a cell-phone.

The amateur seismology Web page (see References) has information on building and interfacing a seismometer to a PC as well as on time references (see "Video Timecode Fundamentals," *INK* 77).

A remote medical monitor at the home of a person with a history of medical conditions is another possibility. Awireless sensor could dial up and dump data regularly, enabling medical professionals to monitor how the individual responds to treatment.

Listing *4-This* PPP **configuration** file specifies the *phone number to dial and the account* ***to log in to.***

```
[ppp]
port=1                     ; Port modem is connected to
irq=auto                   ;IRQ comm port is using or auto
baudrate=0                 ; Baud rate or auto
16550=yes                  ; If 16550 chip present. enable it?
serinitconnect=yes         ; Initiate connection
modempresent=yes           ; Is modem present?
ipaddress=199.106.23.192 ;  IP address of remote Web server
subnetmask=255.255.255.0; Subnet mask for IP address
phonenumber=487-0199       ; Phone number or blank (for direct connect)
username=wcentral          ; User name for logon
password=genesis           ; Password for logon
sescriptfile=a:\config\sescript.ini   ; Path to send/expect script
                                      ; or none
[handshaking]
CTS=yes                    ; Hardware handshaking options
DSR=no                     ; Either yes or no
DTR=no
RTS=yes
```

Listing 5—This *is a sample chat script used* ***to*** *log in* ***to*** *an ISP.*

```
[sescript]
send0="CLIENT"
expect0="CLIENTSERV"
timeout0=1
retries0=0
;
send1="START"
expect1="OK"
timeout1=1
retries1=0
```

IF YOU DO

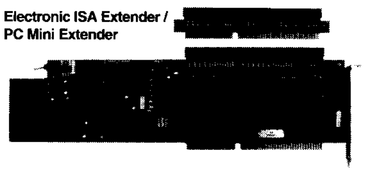
FUNCTIONAL TESTS

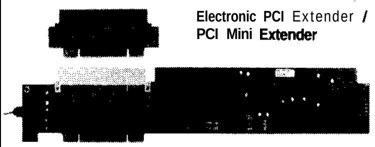
YOU NEED

HOT SWAPPING

ELECTRONIC EXTENDERS

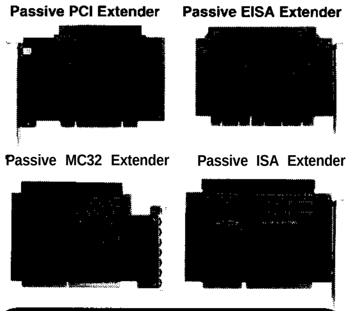
For PCI, ISA, VXI, EISA,VESA, Micro Channel & NuBus. Custom Designs available.





- **Insert/Remove Cards With PC Power On!**
- **Save Time Testing And Developing Cards**
- **Save Wear On Your PC From Rebooting**
- **Adjustable Overcurrent Sensing Circuitry**
- **NO Fuses, All Electronic For Reliability**
- **Single Switch Operation W/Auto RESET**
- **Optional Software Control Of All Features**
- **Breadboard Area For Custom Circuitry**
- **And More...**

Full line of passive extenders: PCI, ISA, VXI, EISA, VESA, Micro Channel & NuBus



AZ-COM, INC.

3343 VINCENT RD., #D
PLEASANT HILL, CA 94523 U.S.A.
TEL: 510-947-1000
FAX: 510-947-1900

24-Hour Fax on Demand:
510-947-1000 Ext.7
***EMail*: sales@az-com.com**
VISIT OUR HOME PAGE AT:
http://az-com.com/

#125

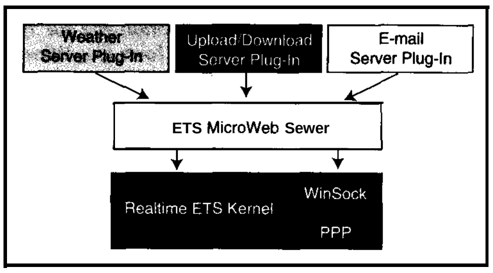


Figure 2-At the core of this system *is the* Realtime *ETS* Kernel *and the ETS Micro* Web *Server, which are extended and customized for a specific application using plug-ins.*

Another application is weather monitoring. Phar tap's "world's smallest Web server" is a PC/I 04-based weather station on the Internet. It uses a Real Time Devices-based PC/I 04 486slc CPU module with a Win-Systems PC/I 04 NE2000 Ethernet card to implement a self-contained Internet host (see Photo 1). Point your browser here, and you can read the current temperature, humidity, and barometric pressure as well as download weather history for the day.

With this technology, it's quite easy to imagine a PC/I 04-based remote weather station that collects data offline. It would connect to the Internet using the serial port on the CPU module and upload weather data to a server.

Phar tap has a complete toolsuite implementing this scheme. It's called a "remote web server," instead of a "host web server" that's permanently connected to the Internet.

HOW IS IT DONE?

To embed remote Internet, take a 32-bit PC/ 104 CPU ('386 or better) with a PC-compatible COM port and external modem or PC/I 04 modem card. Add a PC-based RTOS with protocol stack and PPP driver, write some application code, and integrate.

The protocol stack-sometimes called the TCP/IP module or plug-in-is a collection of routines and drivers implementing the IP suite. It used to be exotic to have a TCP/IP implementation that wasn't part of a larger OS (e.g., Unix or VMS). But with the Internet explosion, good-quality TCP/IP implementations usually come with many RTOSs and other operating systems.

At the bottom of the protocol stack is the interface to the network device drivers. Network interfaces-like Ethernet cards or serial ports-send and receive packets.

For a remote Internet device, I only care about the PPP network interface, which sends packets over a serial line. PPP was developed as a standard datalink protocol between routers over high-speed synchronous serial lines, but it has been adapted to work over asynchronous serial lines and modems. PPP is the datalink protocol of choice for Internet modem connections.

PPP drivers, which communicate over dial-up modem networks, normally implement a scripting language. This language writes "chat" programs, which tell thedriver where to call, how to log in to the ISP, and how to set up a PPP connection.

Once the modem connects and the chat script logs in to the ISP, the PPP protocol takes over and negotiates parameters like the pair of IP addresses to use in this point-to-point link. Once PPP is happy with its negotiations, the PPP network interface is marked as "UP," and the protocol stack can then send and receive IP packets via PPP.

To function fully, the protocol stack needs two parameters-the default gateway address and the name server address. These are normally derived from a configuration file [see Listing 1).

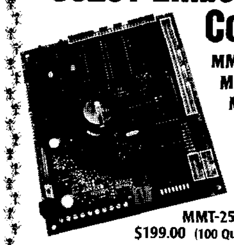
Most protocol modules implement the Berkeley socket library API on the application side. Under Windows, the socket interface exists in the form of the WinSock API.

Having a standard network API is nice, since it lets me design and test network

Listing 6—This control file was downloaded from a Web sewer. Much like a batch file, it specifies what the ***upload/download*** module has to transfer.

```
[directories]
src=/weather                ; Source directory on remote Web server
dest=/remotes/truro_ma      ; Destination directory on host Web server

[upload]
latest.htm=current.htm      ; Current weather readings
daily.htm=daily.htm         ; Daily weather readings in HTML format
daily.txt=daily.txt         ; Daily weather readings in ASCII format
=/mail/sendmail?who=sysadmin@weathercentral.com&file=daily.htm
```

80251 Embedded Controllers



MMT-188EB
MMT-Z180
MMT-HC11
MMT-196
MMT-31
MMT-251
MMT-PIC44
& OTHERS

MMT-251
$199.00 (100 Quantity)

Midwest Micro-Tek is proud to offer its newest line of controllers based on the 8031/51/251 architecture. The 8031 comes in at a surprisingly low cost of $89.00 (100 quantity). Custom work welcome.

MIDWEST MICRO-TEK
2308 East Sixth Street
Brookings, SD 57006
Phone 605.697.8521
Fax 605.692.5112
www.midwestmicro-tek.com
Guaranteed to be in your Future

MIDWEST Micro-Tek

#126

AMX The Real-Time Multitasking Kernel

680x0, 683xx	PowerPC™ family
80386 protected mode	i960® family
80x86/88 real mode	R3000, LR33xxx

NEW: • PowerPC support • *KwikLook* AMX-aware debug tool

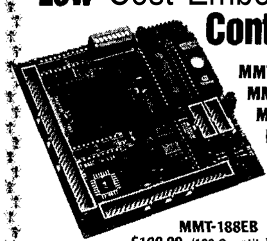
- Compact, ROMable, fast interrupt response
- Preemptive, priority based task scheduler
- Mailbox, semaphore, resource, event, list, buffer and memory managers
- Configuration Builder utility
- Comprehensive documentation
- No royalties, source code included

For a sample of KwikLook and description of AMX,
Phone: *(604)* 734-2796
Fax: (604) 734-8114
E-mail: ***amxsales@kadak.com***
Web: ***http://www.kadak.com***

KADAK Products Ltd.
206 1847 West Broadway
Vancouver, BC, Canada V6J 1Y5

#127

Low Cost Embedded **Controllers**



MMT-188EB
MMT-Z180
MMT-HC11
MMT-196
MMT-31
MMT-251
MMT-PIC44
& OTHERS

MMT-188EB
$199.00 (100 Quantity)

If you're interested in getting the most out of your project, put the most into it. Call or Fax us for complete data sheets and CPU options.
Custom work welcome.

MIDWEST MICRO-TEK
2308 East Sixth Street
Brookings, SD 57006
Phone 605.697.8521
Fax 605.692.5112
www.midwestmicro.tek.com
Guaranteed to be in your Future

MIDWEST Micro-Tek

#128

code on my development machine before targeting the code in an embedded system. Listing 2 shows how the socket API works.

The WinSock library gets initialized with WSAStartup(). If this call is successful, I can go on to the real socket code.

I then allocate a communication socket, thus the name "socket API," via socket (). The arguments indicate the kind of protocol I want and the type of connection.

I'm only concerned with two connection types-SOCK_DGRAMandSOCK_STREAM. A stream-based socket permits a reliable end-to-end communication path, whereas datagrams are packet oriented (with lower overhead) but inherently unreliable.

Once I have a socket, I need the IP address and port number of the host and service I want to connect to. Here, I look up the IP address using gethostbyname() and port number 80, which is the Web server (HTTP protocol).

Connect() establishestheconnection. Connect() canfailformanyreasons,and WSAGetLastError() canfindoutwhy.

Once the session is established, I can send and receive data. Many Internet application protocols (e.g., HTTP and ftp) are implemented via a stream-type connection. I use HTTP (i.e., the protocol for the WWW) to get a page from my Web server.

To read a page on a Web server, I send GET /index. html, which tells the Web server to open and send i ndex. html. The program uses send (), which copies the data from buffer memory and sends it to the remote socket. I receive data with the recv() call.

Stream connections are reliable-no need for checksum or CRC error detection. It's all handled at lower levels of the protocols. Whateveryousendvia send () comes out the other end with the recv() call as long as this virtual data circuit persists.

Once the data is read from the server, recv() returns 0, indicating that there's nothing else to read and the server breaks the connection. I then stop receiving and call closesocket() to release the socket and WSACleanup() to release any resources the WinSock module used.

This is how you write an application to communicate with other Internet hosts. In fact, all standard Internet protocols (e.g., WWW, telnet, ftp, and E-mail) can be implemented with these simple but powerful primitives.

USE THE WEB, LUKE

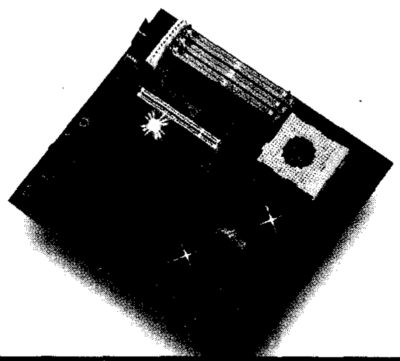
The most widely used Internet application protocol today is HTTP. Every ISP has a Web server, almost every business on the Internet has a Web site, and all desktop systems come with at least one browser, not counting commercial and public-domain browsers for current and older OSs.

Web servers are normally information sources (i.e., the information is retrieved from the server). But, I want to put data on a host Web server from my data logger.

*Listing **7—This** simple plug-in reports the current date and time in a remote Web sewer.*

```
/*datetime.c—sample date/time plug-in adapted from
/*wwwhello.c, which comes with Phar Lap's MicroWeb Server */
/* date_htm-Main HTML page for a date/time plug-in */
BOOL _cdecl  date_htm(REQ_INPUTS *pInp, REQ_OUTPUTS *pOutp){
  char buff[512];
  if(!hpg_CreatePage(pOutp))              /* Open empty HTML doc */
    return  FALSE;
  html_tag("<title>"); /* Start with HTML header */
  html_text("\"Date Time Page\"");
  html_tag("</title>\n");
  html_tag("<h3>");
  html_text("\"Current Date and Time\"");
  html_tag("</h3>\n");
  html_tag("<hr>\n");
  html_tag("</head>\n");
  html_tag("<body>\n"); /* Grab and convert current date and time */
  html_tag("<pre>\n");
  _strdate(buff);
  html_text("Current date: %s\n", buff);
  _strtime(buff);
  html_text("Current time: %s\n", buff);
  html_tag("</pre>\n");
  html_tag("<hr>\n");
  html_tag("</body>\n");
  return hpg_DoneNotCachedPage(pOutp);} /* Done-return HTML page */
```

. 7 ISA slots
• 486 support



The G486VPC motherboard combines quality and affordability with an industrial design to meet your needs. Seven full length ISA slots ensure expandability for the cards that you use.

Our engineering staff will gladly discuss custom motherboard designs. FCC & UL certified systems are also available.

- Six full length 16-bit ISA, One shared 8-bit ISA/PCI slot
- Intel, AMD & SGS-Thomson 486 CPU support
- VIA chipset
- Up to 64MB RAM, 256KB cache

DFI®

30

I can do this via the HTTP POST method. I showed you how to use GET to retrieve pages from a Web server. POST works the opposite way. The client (here, the remote data logger) connects to the server and issues POST with the data I want to send as input. It's the same command used to submit forms from Web pages (i.e., when you sign a guest book or order something via the Internet). POST is often implemented with a server's CGI script, which is a special batch file that processes received data.

Or, I could use PUT, which replaces one Web page with another. Several Web-authoring tools (e.g., FrontPage) use it to upload Web pages. However, PUT is not implemented by all ISPs, since it is harder to administrate than POST.

PUTTING IT TOGETHER

One nice feature of PC-based embedded systems is the availability of off-the shelf RTOSs and components. These RTOSs are typically lean and mean.

Most optimization has been taken care of, since the architecture is known in advance. And since many of the hardware devices (e.g., serial ports and Ethernet cards) are standard, the same driver works with other vendor's hardware. System integration is a breeze.

An RTOS that's particularly good for this application is Phar lap's Realtime ETS Kernel. As the core component of Phar Lap's TNT Embedded Toolsuite development system, it's a deterministic, thread-based, hard real-time kernel supporting a Phar Lap-defined subset of the Win32 API.

It has everything I need to build a remote Internet data logger, including the ETS MicroWeb Server-a collection of libraries and plug-ins linking with my application to implement a remote Web server. The ETS MicroWeb Server runs under the Realtime ETS Kernel on the target system.

With this package, all that's left is to buy a PC/I 04 CPU module, configure the software, and write a module to perform my data-collection application. I had a demo system running in less than a day. Figure 2 gives you an idea of how it fits together.

This system has a small memory footprint-perfect for PC/I 04 CPU modules. The Realtime ETS Kernel is only -32 Kb, and a complete system with the WinSock library and MicroWeb Server fit in less than 1 Mb of memory. Most '486-based PC/I 04 modules come with 4 Mb of DRAM-plenty of room for sophisticated Internet-based apps.

The ETS MicroWeb Server is expanded by using plug-ins, which are standard Windows-based dynamic load libraries (DLLs). This makes the development environment particularly nice, since standard Windows-based C/C++ compilers can be used.

But, don't tear the shrink wrap on the latest Visual C++ or Borland C++ quite yet. You might not get to write code.

The Realtime ETS Kernel comes with a PPP driver, WinSock API library, E-mail plug-in, ETS MicroWeb Server, and Web based upload/download plug-in. I already discussed the PPP and API, so let's look at the upload/download plug-in.

This plug-in is a DLL that transfers Web pages between Web servers. It might sound odd, but let's look at how this works for the data logger.

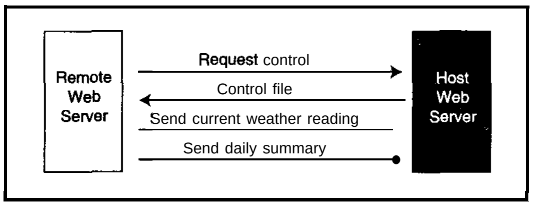


Figure 3—Here's what a sample upload session might look like.

As Listing 3 shows, the plug-in has a configuration file on the data logger that controls when to wake up and what to do.

At wake up, the upload/download plug-in opens a HTTP connection to the host Web server specified (www.weathercentral.corn) and downloads a control file (/con - trol/truro_ma/control. ini). The PPP driver then wakes up and consults its configuration file to find out where it needs to dial up to establish an Internet ccnnection.

Listing 4 has a sample of the PPP configuration file, and Listing 5 shows the chat script used for this connection. Once the PPP module connects, the upload/download plug-in downloads the control file.

In the sample control file in Listing 6, the plug-in uploads the local file /weather/current. htm to the host Web server as /remotes/truro_ma/latest. htm. It also sends daily.htm and dai1y.txt. After all that's done, it E-mails the file dai1y.htm to <sysadmin@weathercentral.com>. Figure 3 shows the transactions of a typical upload session.

Notably, the local file references in the control scripts are really HTTP resources on the data logger's MicroWeb Server. These resources can refer to a method in a DLL or a file in the file system.

The reference to /weather/current. h tm in the example refers to a method that generates HTML pages dynamically with Phar Lap's HTML-on-the-fly library. Once all the data is transferred, the PPP plug-in times out and hangs up the modem.

Pretty slick! All that's left is coming up with a method that presents the data to be uploaded as a dynamic HTML page.

Listing 7 shows a simple method that reads the current date and time and formats the output as a HTML document. Of course, if I wanted to build a remoteweather station, I could just use Phar Lap's weather-station plug-in and be done with it.

What else can I do? By adding definitions in a [download] section, I can instruct the remote Web server to grab files from the host Web server and store them in

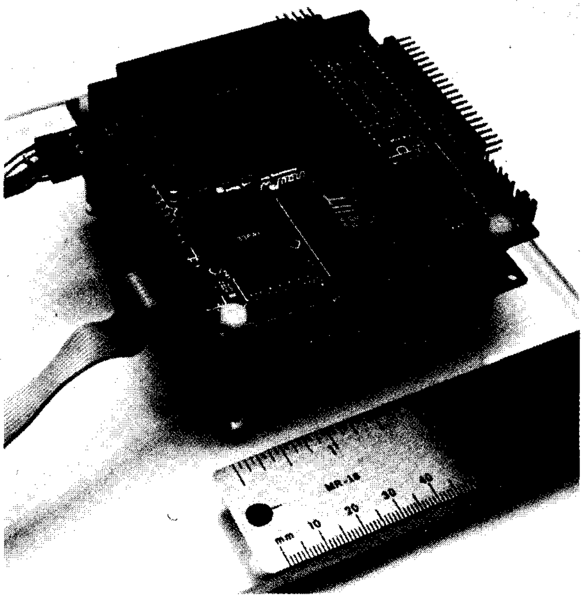


Photo 1—Phar Lap's PC/104 based "smallest Web server" fits in less than a 4" x 4" x 2" footprint using modules from RTD and WinSystems. It implements a complete Internet host with Phar Lap's ETS MicroWeb Sower and Realtime ETS Kernel.

RAM or on the SSD. This mechanism can update configuration files and software on the data logger remotely. Once the data transfers to the host Web server, standard Web browsers or application-specific data-analysis software retrieve and process it.

SECURITY CONCERNS

Now, I have a way for a remote data logger to transfer files to and from the Internet on demand using standard Internet protocols. But, what about security?

There are three things to be concerned about. An intruder may try to break into the data logger from the network and wreak havoc, someone can try to read or alter the data on the host Web server, and data can be intercepted while traversing the Internet.

The first issue is pretty easy to address. The data logger is only online during short periods of time. While online, you can control what services are available to the Internet by the type of plug-ins you have installed and Phar Lap's security plug-in, which restricts access to the remote Web server by specific Internet addresses.

Transfers to and from the remote Web server **are** controlled via configuration files. With this mechanism, I can control which data transfers to which host Web server.

The host Web server usually implements several types of security methods by implementing authentication (i.e., accounts and passwords) and access lists that restrictwhich host accesses certain areas on the host Web server. Also, since this host server usually resides with some ISP and is shared by paying subscribers, the ISP operator should take a keen interest in practicing good security.

To prevent someone from listening to our traffic on the Internet, we can use standard encryption methods (e.g., RSA) already in placeforsending sensitivedata (e.g., credit-card information) over the Internet. Using RSA, the remote Web server requests a public key for a resource from the host Web server. This key then encrypts the message.

While the message is encrypted using a public key, it can only be decrypted with the host server's private key, which is kept secret. The message is unintelligible while traversing the Internet.

You can also use secret key algorithms like DES to encode messages. In this system, the same key encodes and decodes, making key management and distribution harder.

While none of these methods are *100%* safe, they protect us from the semi-serious bad guy. Ultimately, it comes down to how sensitive your data is and how hard you want to make it for someone to get it.

ON THE 'NET

A desktop system used to build Internet-ready applications. With Phar Lap's tool kit, I can build Internet-ready embedded devices with a 32-bit PC/I 04 module and a modem.

Now, I just have to figure out how to filter the data-logger messages from the never-ending volume of get-rich-quick scams. Too bad off-the-shelf embedded Internet technology didn't exist 15 years ago. I'd probably have more hair now and be a better Web programmer, too. EPC.PCQ

Ingo Cyliax is a research engineer in *the* Analog *VLSI* and Robotics lab and teaches hardware design in the computer-science department at *Indiana* University. He a/so does software and hardware development with Derivation *Systems, a San* Diego-based formal-synthesis company. You may reach *Ingo* at *cyliax@EZComm.com.*

REFERENCES

Text

Phar Lap Software, *TNT Embedded* Technologies Guide *book,* Cambridge, MA, 1996.

B. Quinn and D. Shute, *Windows Socket Network Programming,* Addison-Wesley, Reading, MA, 1996.

Internet

Phar Lap's weather station, smallest.pharlap.com

Amateur seismology how-to, psn.quake.net/equip.html

SOURCES

486SU PC/ 104 CPU Modules
Real Time Devices USA
200 Innovation Blvd.
State College, PA 16804
(8 14) 234.8087
Fax: (8 14) 234-52 18
sales@rtdusa.com
www.rtdusa.com

PC/104 modules (CPU, Ethernet, modem)
WinSystems
715 Stadium Dr.
Arlington, Texas 7601 1
(8 17) 274.7553
Fox: (817) 548.1358
www.winsystems.com

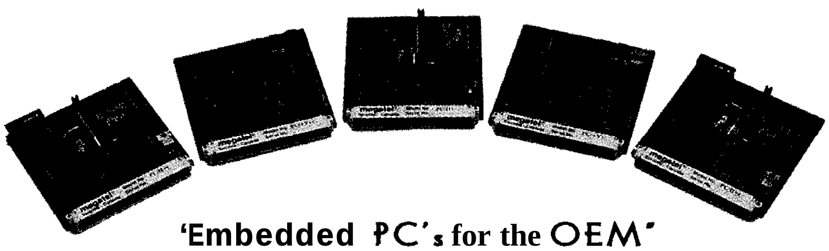
RTOS, Web server, **IP** stock
Phar Lop Software
60 Aberdeen Ave.
Combdridge, MA 02 138
(617) 661-1510
Fox: (617) 876.2972
info@pharlap.com
www.pharlap.com

IRS

4 16 Very Useful
417 Moderately Useful
418 Not Useful

Small Space Solutions

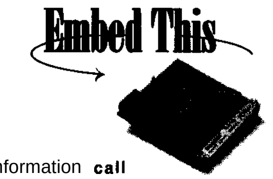
Megatel Offers a Wide Range of Embedded PC Architecture



'Embedded PC's for the OEM'

On-Board Features such as:

- 486DX,™ 386,™ 8088 or V40 Processors
- Local Bus Super VGA supporting LCD Panels
- Up to 16MB DRAM
- Ethernet Local Area Network
- On-board programmable Flash™
- Low Power Consumption
- Operation +5 volts only (typical)
- 2 IBM compatible RS-232C Serial Ports
- 8-bit general purpose Parallel I/O Port
- Only 4" x 4" form factor
- ISA or PC/104 Bus compatible options



For **more** information **call**
Toll Free: 1'669'762.5572 or 416.245'2953
fax: 416•245•6505 **email:** sales@megatel.ca
Website: www.megatel.ca

megatel® **1•888•SMALL•PC**

#131

Fred Eady

Interlaces and GUI-Buildfng Packages

Part 1: Web-Accessible Virtual Front Panel

Someti nes *the* product is perfect, *but* you can't make *an* interface for *a* reasonable price. *tred* checks **his** fool *drawer to* see if anything can come *to the* fescue. He *Finds em* Wure's **embedded application** *enabler and Advantech's KM-4862.*

You just designed the ultimate electronic device. Every ounce of functionality was squeezed in. The board layout is tight, the component count low. It can stand alone in a closet until redemption day.

You thought of everything, every contingency. If marketing didn't think of it, you did. And, it's all in the current firmware.

It's perfect-but the user interface is going to cost more than the product!

Now what? No engineering bonus. No innovation award. What **you** saved in hardware design will be spent on the interface. You need a friend (and a cheap front panel) about now, huh?

The good news is that we can call on some friends-old and new-to help us out of this jam.

The Internet has become so popular, it's a household word. And, riding its coattails is the Web browser, taking a place as the world's most familiar GUI.

Web-oriented devices and software are everywhere. Microsoft's Visual Studio is fully Web compliant. Even Visual Basic 5 has a specialized integrated Web browser. Every OS and software development-tool package from QNX to Phar lap ships with Web-capable functionality.

Today's embedded hardware can support any Web-oriented software package. Most anything can be done on the Web somehow, some way.

So, let's put our friends to the test, stir in some Java, and sweeten it with a bit of emWare. The mix yields a feature-rich virtual user interface that can be accessed via a serial interface or Web browser.

Add a PCM-4862 embedded platform from Advantech, and whop! Instant Web accessible virtual front panel on the cheap. This time around, I'll introduce you to the tools we'll use to design and build our own embedded virtual front panel.

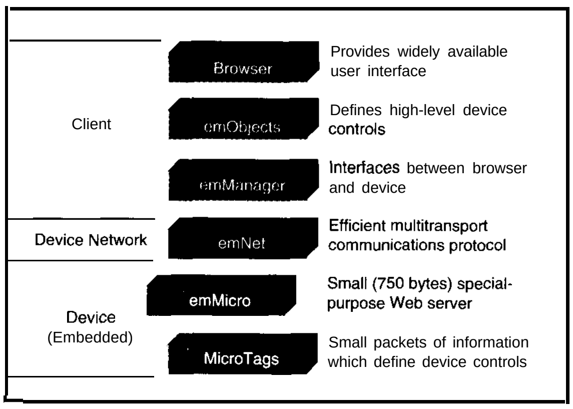

Client
Device Network
Device (Embedded)
Browser
emObjects
emManager
emNet
emMicro
MicroTags
Provides widely available user interface
Defines high-level device controls
Interfaces between browser and device
Efficient multitransport communications protocol
Small (750 bytes) special-purpose Web server
Small packets of information which define device controls

Figure I-Notice ***that*** *the embedded-PC application is at* ***the top of the food chain here.***

WHAT'S EMWARE?

emWare brings a unique embedded-application enabler to the Internet world. Embedded Micro Interface Technology (EMIT) lets users control, configure, and manage devices not previously connected to the Web using available Web browsers.

Web browsers are in everyone's hands these days. Their point-and-click interfaces are easy to use and provide a perfect GUI.

One advantage to emWare's using a Web browser is that little-to-no new software-development effort needs to be put into the embedded-PC interface. Anyone who can point and click can manipulate remote devices with emWare.

With standard Web-browser programmability and emWare controls, you can present an almost identical virtual image of the remote device--complete with interactive controls and functions the user can manipulate via the browser interface.

The EMIT Software Development Kit (SDK) does this via a Netscape plug-in. The interaction with the Web-browser software is the real power behind EMIT.

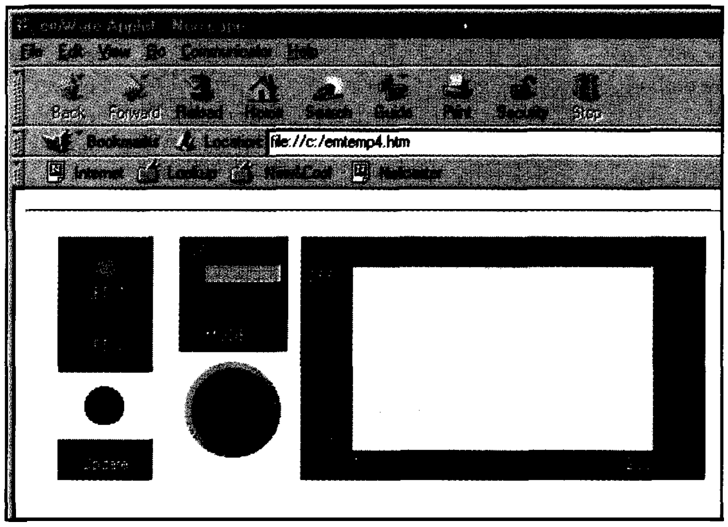


Photo l-Knobs, switches, *LEDs,* graphs-all remotely controlled. You can trash those *hole* punches and socket wrenches you had to use before to fabricate your front panels.

The remote device could be a software-configurable black box without external buttons, indicators, switches, or controls. Using standard browser tools and emWare's object-oriented interface logic, the designer can fabricate a virtual front panel simulating the black box's internal electronic controls.

Since much of the standard interface coding can be eliminated, emWare can cut production costs by a factor of 100. That remote black box is now less costly to manufacture, since no physical user-interface components are required.

Future functionality within the black box's firmware and hardware can be designed in at initial manufacture and activated or deactivated by changing the black box's "look" to the user at the interface.

And, the black box is easier and less costly to maintain. The Internet connection provides a reliable and cost-effective way to offer remote firmware and software support.

The technician's virtual view can be configured differently from the user's view to aid in remote diagnostics. The black box view can even be customized for each user.

EMIT ARCHITECTURE

EMIT operates as a "fat client, skinny server" system with the client being the embedded-system-based Web browser and the server being an intelligent remote electronic device.

emNet is a serial communications protocol that provides network connectivity between the client and server. emNet is transport-layer independent and supports all standard point-to-point, Internet, and Intranet protocols. Its ability to interface to any Web browser and its lack of concern for protocol assure platform independence.

The remote-server engine, emMicro, consumes as little as 30 bytes of RAM and 750 bytes of ROM. This configuration is minimal, so most applications end up with a little over 1 KB of ROM. Together, emMicro and emManager form a true distributed micro HTTP server with extensions supporting the unique requirements of electronic devices.

Minimal required resources aren't the only skimpy attributes of the micro Web server. The conversations between the embedded PC and the server are highly condensed data packets known as MicroTags.

MicroTags are 1 byte or more in size and correspond one-to-one with emObjects at the embedded PC. emObjects is a small library of preprogrammed functions consisting of JPEG/GIF graphics or Java applets. They give the browser the high-level controls necessary to interface with the remote device.

The actual interface is assembled with emObjects that translate to the virtual controls and buttons the user sees via the browser. emManager expands each MicroTag using a process called dynamic expansion.

Once received and processed, each HTML page of MicroTags is displaced with corresponding embedded-PC emObjects by emManager. They are then converted into information presented to the user via the embedded-PC browser.

Using tiny MicroTags eliminates having to ship large chunks of data from the server, reducing link bandwidth requirements and increasing user-display speed. The EMIT architecture is illustrated in Figure 1.

CREATING THE USER INTERFACE

All EMIT components work together to

listing l-Here's the *glue* behind that cheap LED array.

```
<title>emWare Example 1 Applet</title>
<!-- Uses the Java flow layout for placing two emObjects. Object0
 is a slider controlling an embedded variable (test0). Object1 is
 a T-segment display of results of setting embedded variable -->
<hr>
<!-- EMBED TYPE tag needed to activate emWare Netscape plug-in -->
<EMBED TYPE=application/x-emj name="emitjri" HIDDEN>
<!-- emApContainer is the Container Applet for emObjects. Width
 and height must be specified with the code declaration. -->
<applet code=emApContainer.class name="Example1" width=300
 height=220>
<!-- Specifying the number of objects reduces the number of objects
 the applet will search for. Default is 256. Make sure the largest
 object number is less than objects. -->
<PARAM NAME="OBJECTS" VALUE="2">
<PARAM NAME="OBJECT0" VALUE="emJava.emSliders.emSlider">
<PARAM NAME="OBJECT0_BackgroundColor" VALUE="gray">
<PARAM NAME="OBJECT0_ForegroundColor" VALUE="white">
<PARAM NAME="OBJECT0_Min" VALUE="0">
<PARAM NAME="OBJECT0_UpdateContinuously" VALUE="true">
<PARAM NAME="OBJECT0_JriVariable_0" VALUE="Test0 ActionEvent">
<PARAM NAME="OBJECT1" VALUE="emJava.emDisplays.em7SegDisplay">
<PARAM NAME="OBJECT1_BackgroundColor" VALUE="gray">
<PARAM NAME="OBJECT1_ForegroundColor" VALUE="red">
<PARAM NAME="OBJECT1_JriLink_0" VALUE="Test1 Value">
</applet>
<hr>
```

electronic device. Users work directly with the Web browser at the embedded PC.

Via the browser, the user requests the interface for the device they need to work with. The browser sends the request to emManager, which converts or translates the high-level user request and sends it to emMicro at the device.

emManager requests an HTML page containing MicroTags, which is how the interface is stored within the device. All MicroTags for device controls are embedded within an HTML page, so the entire device interface is contained in one HTML page. When emMicro receives the request, it serves the page back to emManager.

PROCESSING USER REQUESTS

Say a user wants a switch toggled at the remote device. Once the request is initiated, the browser invokes emManager and a request is sent to emMicro at the device.

Receiving the request causes the controller or processor to perform the request. Toggling the switch changes the information in the device and the MicroTag.

The embedded-PC view must be updated to show the new switch condition. emMicro updates the variables in the MicroTag and sends them to the embedded PC. emManager receives the updated HTML page and displays the results of the request on the embedded browser screen.

PUTTING EMIT TO WORK

First, plan the interface. When users connect to the device, the Web browser accesses the device via HTML pages. The pages have an applet tag, identifying a Java program to load and place onscreen.

The emWare Cont. a i ner applet is programmed and available to the Web browser after emWare is installed. The HTML files control the look and feel of the interface.

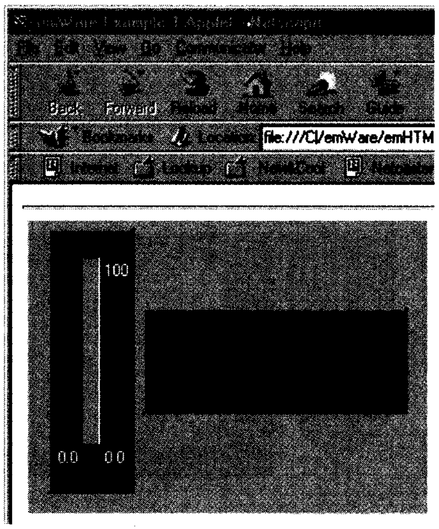


photo 2—One really cheap LED array!

Files for the high-level interface may be installed on the device's processor or at the embedded PC. Installing the files on the embedded PC saves space but requires device-specific files to be loaded before connecting to the device.

The only con with loading the interface on the device is that extra space is needed in the device ROM. Loading the interface on the device enables any user with emWare client software on their embedded PC to use the device without delay.

Once you decide where the interface will reside, it's time to lay out the screen. emWare provides many ready-to-roll switches, gauges, LEDs, and knobs, or you can design custom controls. Photo 1 shows some of the built-in emWare controls.

Once the layout is pinned down, the next step is to code the high-level interface. You need to create an HTML page that:

- activates the emWare NetScape plug-in EMITJRI
- runs the emWare Java applet emApContainer and defines its properties
- contains applet parameter tags defining the interface's device controls

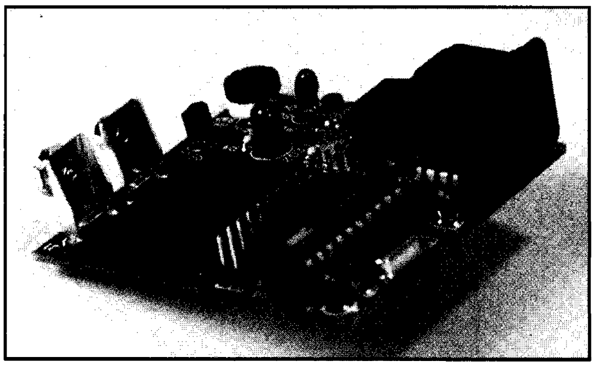


photo 3—The core of the PCM-4862 is solid. You can hang as much or as little as you need off this SBC.

First, the HTML page activates EMITJRI, whidh extends Netscape with functionality tailored to the needs of the embedded device.

Next, the HTML page must run emApContainer. This applet creates emObjects dynamically, sets emObject properties, links emObjects to events, controls object layout, and links emObject events and properties to the device controller variables.

The HTML page defines and sets emApContainer's properties and creates emObject definitions. This process is set forth in Listing 1. Photo 2 shows the resulting interface.

PUTTING IT ON THE TABLE

With the high-level interface complete, programming theembedded device to work with emWare is a simple matter of integrating emMicro with the application software controlling the interface.

This task is accomplished with data tables. emWare uses data tables (see Listing 2) so emMicro and the device application program can pass data variables between themselves.

Next, you must set up the Capabilities Document, Internal Variable Table, and External Variable Table.

The Capabilities Document is a table of information telling emManager what capabilities the controller or processor supports, so itcancommunicateappropriately.

The internal Variable Table is used to find the internal memory addresses where variable information is stored.

The External Variable Table tells the high-level interface about the device's variables. It ties the high-level interface at the embedded PC to the low-level controller variables at the device.

You then incorporate the emMicro functionality into the device's application code by calling the emMicro initialization routine and performing any additional initialization necessary. Standard include files containing the emWare code let you perform this function.

After all variables and program functions are tested, the interface is compiled, compressed, and placed into the device for final testing.

ADVANTECH'S PCM-4862

The fat-client concept implies that the embedded PC is robust enough to run a Web browser and emWare client software. It further implies that a GUI-capable operating system is resident as well.

Advantech's PCM-4862 All-in-One SBC is a good choice for our virtual front panel. The small-footprint 100-MHz '486 processor is complemented by the usual peripherals, including onboard Ethernetcapability.

Mine comes with 16 MB of RAM and a 1.44-MB SSD. Count the I/O connectors in Photo 3 to get a good idea about the PCM-4862's capability.

Normally, we'd place the embedded PC in the resourceprudent "remote" role. But, since the PCM-4862 is more than enough machine to handle the emWare 1 .O SDK and Netscape under Windows 95, it's not a problem.

listing **2—It's** not as complicated as it looks, but every byte has to be in its *place.*

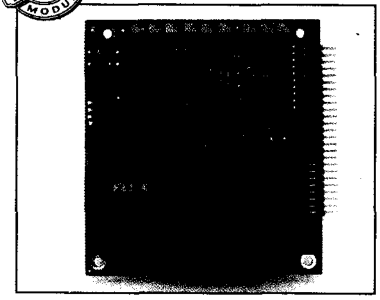
```
·*emWare Data Tables *
Capabilities
  .byte 02          ; Byte 0, DO Packet Document Retrieval
                    ; Endian specification, D1-D3 Byte ordering
  .byte 00, 07      ; Bytes 1 & 2, Max Packet Size   (7)
  .byte 00          ; Byte 3. Protocol Level     (0)
  .byte 01          ; Byte 4, Protocol Revision (1)
  .byte 00, 01      ; Bytes 5 & 6, Device ID          (0)
  .byte 00, 05      ; Bytes 7 & 8, Product ID         (0)
  .byte 00, 01      ; Bytes 9 & 10. Manufacturer ID   (0)
  .byte 00, 110     ; Bytes 11 & 12, emWare Revision (1.1)
  .byte 00011111b   ; Byte 13, Tables Present:
        ;   DO Static Documents, D1 Variable Table
        ;   D2 Dynamic Documents, D3 Function Table, D4 Event Table

Caplen = ($ - Capabilities) - 1
;*Internal variable table retrieves variable address from host and
; refers to variables according to where they appear in EMvartable.
; Table contains actual address where these variables reside
; Edit this table to include names of all your variables
ivartable
  .byte Knob
  .byte Red
  .byte Green
  .byte Mode
  .byte Button
  .byte ChgMode
  .byte 0           ;Place holders for nonvolatile variables
  .byte 0           ;Place holders for nonvolatile variables
  .byte 0           ;Place holders for nonvolatile variables
  .byte 0           ;Place holders for nonvolatile variables
;**variable table sent in response to GetVarSymbols request from host
EMvartable
  .byte 10                  ;= # of variables
  .byte 0
  .byte BYTETYPE+READWRITE  ;= TYPE byte for variable 1
  .byte TKNRESPONSE
  .byte BYTETYPE+READWRITE+SEQUENCED ;= TYPE byte for variable 2
  .byte 0
  .byte BYTETYPE+READWRITE  ;= TYPE byte for variable 3
  .byte 0
  .byte BYTETYPE+READWRITE  ;= TYPE byte for variable 4
  .byte 0
  .byte BYTETYPE+READWRITE  ;= TYPE byte for variable 5
  .byte 0
  .byte BYTETYPE+READWRITE  ;= TYPE byte for variable 6
  .byte NONVOLATILE         ; In nonvolatile storage
  .byte BYTETYPE+READWRITE
  .byte NONVOLATILE         ,In  nonvolatile  storage
  .byte BYTETYPE+READWRITE
  .byte NONVOLATILE+ARRAY   ; In nonvolatile storage
  .byte BYTETYPE+READWRITE
  .byte NONVOLATILE         ; In nonvolatile storage
  .byte BYTETYPE+READWRITE
  .text "Knob\000"          ; Null terminated string for variable 0
  .text "Red\000"
  .text "Green\000"
  .text "Mode\000"
  .text "Button\000"
  .text "ChgMode\000"
  .text "NV1\000"
  .text "NV2\000"
  .text "Data(200\000"      ; Array of 200 values
  .text "NV3\000"
EMvartablelen = ($ - EMvartable) - 1
```

rtd®

Sets The Pace
In PC/104
Data Acquisition

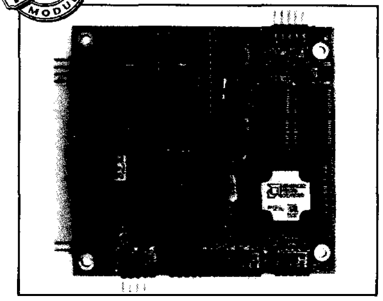
Scan 16 Channels...
Any Sequence...
Any Gain...
500 KHz

EMBEDDED-PC PC/104 MODULES


DM6420 500 KHz Analog I/O Module
with Channel-Gain Table and FIFO

With Companion
Am5x86™ 133 MHz
PC/I 04 **cpuModules**

EMBEDDED-PC PC/104 MODULES


The CMV586DX133 offers
versatile embedded functionality

Our PC/I 04 and **ISA Bus**
Product Lines Feature
Intelligent DAS Cards With
Embedded PC and DSP,
Analog and Digital I/O, CPU,
Shared Memory, SVGA, PCMCIA,
CAN Bus and GPS Modules

rtd® **Real Time Devices USA, Inc.**
200 Innovation Boulevard
State College, PA 16804-0906 USA
Tel:1(814) 234-8087 • Fax:1(814) 234-5218
URL:www.rtdusa.com
E-Mail:sales@rtdusa.com

RTD Scandinavia Oy
Helsinki, Finland
Tel:358-9-346-4538
Fax: 358-9-346-4539

RTD is a founder of the PC/104 Consortium

132

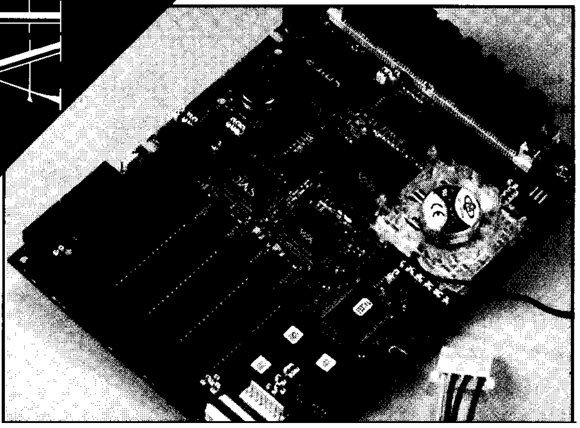


Photo 4—**It's** amazing. Turning the pot changes the LED's intensity on the ***Netscape*** *screen. You can also do the same* with *the virtual* knob *from* Netscape.

This emWare SDK version is limited to a single serial interface through Netscape only. Future releases will include functionality for modem, proxy, and multiple interface ports as well as Internet Explorer support, which is where the Ethernet interface will come in handy.

In the next release, total emWare emulation will be available. Using the two PCM-4862 serial ports, we'll design and test our emWare/Advantech virtual front panel using only embedded PCM4862 hardware.

WHILE WE'RE WAITING

Now, the only server micro supported is the 8051. I hear the nextversion will include other processors, too. Although I've turned the who's-on-what-end-of-the-link embedded-PC paradigm around, who's to say the remotedevice running emMicro couldn't be an embedded PC?

Next month, I'll take a more in-depth tour of emWare's internals and how to embed them whileassembling our own virtual front panel on the Advantech PCM-4862.

I'll choose a processor, design the interface, and emulate it on an Advantech embedded PC. I hear a point-and-click design interface will be included with the new version of emWare. I'll show you that, too.

If you want to bone up on emWare, an emWare SDK is currently available that includes documentation and a demo board so you can put EMIT to work immediately. The reprogrammable SDK demo board shown in Photo 4 consists of an 805 1 micro with supporting EEPROM, LEDs, switches, and pots. A demonstration program lets you remotely control all those goodies and quickly get up to speed with emWare. APC.EPC

Fred Eady has over 20 years' experience as a systems engineer. He has worked with computers and communication systems large and small, simple and complex. His forte is embedded-systems design *and communications.* Fredmaybe *reached at fred@edtp.com.*

SOURCES

EMIT, SDK
emWare
1225 E. Fort Union Blvd., Ste. 220
Midvale. **UT** 84047
(801) **256-3883**
Fax: (80 1) 256.9267
www.emware.com

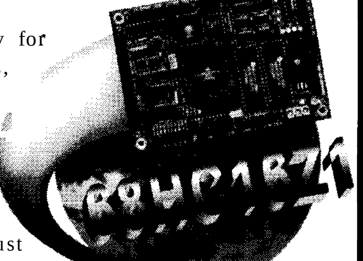
PCM-4862 All-in-One SBC
American Advantech Corp.
750 E. Arques Ave.
Sunnyvale, CA *94086*
(408) 245-6678
Fax: (408) 245-8268
www.advantech-usa.com

IRS

4 19 Very Useful
420 Moderately Useful
42 1 Not Useful

$249.95 QTY. 1

SINGLE BOARD COMPUTER



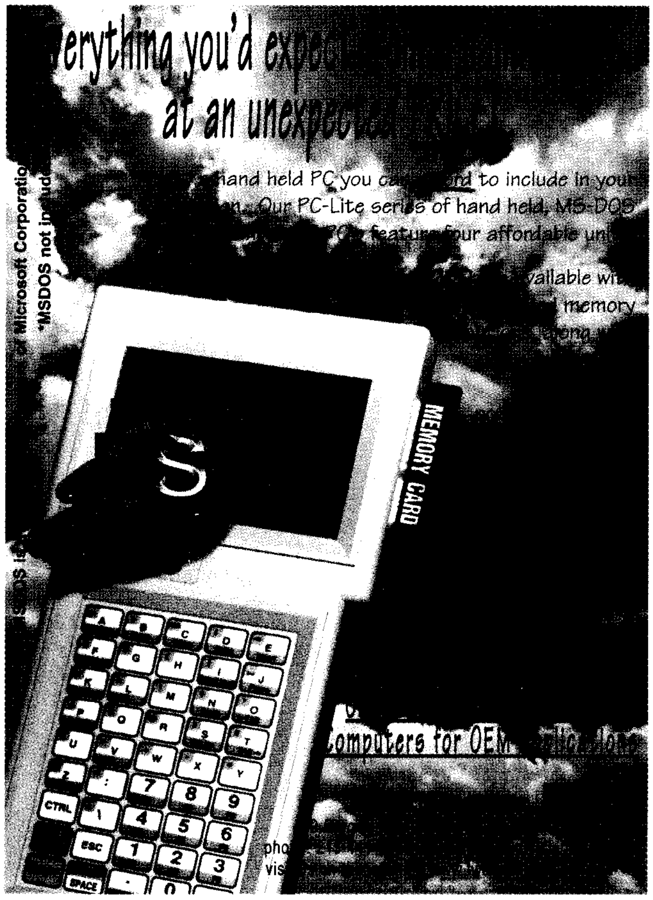
This board comes ready for demanding applications, with 16 bit DSP capability, Analog & Digital I/O, and **68HC11 Compatibility.** Backdoor Debug and up to one Megabyte of battery Backed RAM creates a robust development environment.

* 16 Programmable I/O lines
* 8 High Drive Digital Outputs
* 8 Channels of Fast 10 bit A/D
* Optional 8 Channel, 8 bit D/A
* Up to 2 RS232/485 Serial Ports
* Backlit capable LCD Interface
* Optional 16 Key, Keypad & Interface
* 16 bit Timer/Counters with PWM & Clock/Calendar
* 64K RAM & EEPROM included plus Flash Capability

1985 - 1997
OVER
12
YEARS OF SINGLE BOARD SOLUTIONS

EMAC, inc.
618-529-4525 Fax 457-0110 BBS 529-5708
11 EMAC WAY, CARBONDALE, IL 62901
WORLD WIDE WEB: http://www.emacinc.com

#133



#134

MC68030 Workstation

MICRO SERIES

Ingo Cyliax

Cross-Development Environment and Downloading

Part 3 of 3

Ingo finishes his three-part series on pulling together an affordable 68000 development board. The final step—software—is based on tools freely available on the Internet. He can even boot the board over the network.

At a university, it's sometimes easier to justify spending money on hardware than software. And besides, people in computer-science departments tend to pride themselves in their ability to roll their own.

But, I didn't fancy spending the next five years writing and supporting yet another C compiler. Instead, I opted to look for freely available tools on the 'Net.

Several C compilers and 68000 assemblers are out there, so in this article, I discuss some of the packages I used. Of course, I still had plenty to do to set up a suitable environment, so I wasn't at all deprived of the experience to roll my own.

In this final installment of the MC68030 series, I start out with a look at some software-development tools and how network booting works. I also discuss some applications that were done on the MC68030 system described in Parts 1 and 2. One of the tools is a network application that permits students to log in to a Unix machine to edit and compile programs and then download them for execution and debugging.

COMPILERS AND ASSEMBLERS

The MC68030 comes from a long line of 68000 chips that was first introduced about 17 years ago. Naturally, a large collection of software has accumulated over the years. There are many assemblers, and almost all programming languages have been implemented on the 68000.

Originally, the computer-science faculty at Indiana University intended for students to program this system in assembly language only. For this, I chose asm68, which was written by Paul McKee.

This assembler is a stand-alone system that takes Motorola-format assembly-language files and generates a Motorola S-record hex file. It also generates a traditional assembly listing and a symbol table file.

asm68 comes in source-code form and is fairly easy to compile for other systems, since it doesn't require many services from the OS. I've built this assembler under Minix and Unix, and students also have built it under DOS on their PCs at home. Being able to run it under Unix lets students edit their source files in the familiar programming environment of our network of Sun workstations.

While asm68 is very compact and enables the students to explore architectural features of the 68030 system in lab, it doesn't allow multimodule programs.

Multimodule programs are when the student can use modules developed in an earlier lab (e.g., their own keyboard driver) and link them into a more sophisticated program. With asm68, the student has to cut and paste everything into one monolithic file and assemble it.

Besides the limitation of single-file modules, the instructors thought students would benefit from writing some of the high-level functionality in C, while maintaining exposure to the hardware interface through assembly-language modules. Once the low-level modules were written, the students would have more flexibility in writing more complex programs.

By mixing C and assembly language, students gain some insight in how high-level language constructs are implemented on a machine like the MC68030.

GNU TO THE RESCUE

Luckily, since I was already using the GNU-C environment to develop the monitor for this system and other applications, this was a no brainer. I could use the same development environment being used to develop the firmware for the system in the students' labs.

Well, that was easier said than done. Let's first take a look at GNU-C (the compiler) and GNU-as (the assembler) to write software for the stand-alone software and operating system.

The GNU-C compiler has been around for quite some time. It was originally written by Richard Stahlman of the free-software foundation (FSF). In fact GNU, which stands for "GNU is Not Unix," is a whole tool suite of utilities that the FSF develops and gives away free.

Other popular utilities besides GNU-C are Emacs and Ghostscript. Due to the open philosophy, many people and organizations contribute to these tools by porting them to new environments, adding features and functionality, as well as fixing bugs.

The current GNU-C supports too many architectures to list here. It can be used as a system compiler for a particular architecture/operating system and, in some cases, is even better than the vendor-supplied compiler.

For example, Linux and FreeBSD, both freely available Unix-like operating systems, use GNU-C as the system compiler. Also, many software packages can compile with GNU-C, which makes it sort of a standard C dialect across many platforms.

One feature in particular that's not as well known to GNU-C users also

Listing I--This *BOOTP* **fable** *resides on the server. If contains the mapping between the Ethernet and Internet addresses as well as the name of the boot image to download.*

```
# Legend:
# first field — hostname (should be full domain name)
# hd — home directory
# bf — bootfile
# cs — cookie servers
# ds — domain name servers
# gw — gateways
# ha — hardware address
# ht — hardware type
# im — impress servers
# ip — host IP address
# lg — log servers
# lp — LPR servers
# ns — IEN-116 name servers
# rl — resource location protocol servers
# sm — subnet mask
# tc — template host (points to similar host entry)
# to — time offset (seconds)
# ts — time servers
#
# Be careful about including backslashes where they're needed. Weird
# things can happen when a backslash is omitted where one is intended
#
default:sm=255.255.255.0:hd=/tftpboot:bf=null:\
  :ds=0.0.0.0:ns=0.0.0.0:ts=0.0.0.0:to=18000:
rmc:tc=default:ip=198.88.16.3:ht=ethernet:ha=0000C034D810:bf=net030
reset:tc=default:ip=198.88.16.4:ht=ethernet:ha=0000C022DC10:bf=net030
rw:tc=default:ip=198.88.16.5:ht=ethernet:ha=0000C0945D14:bf=net030
ocs:tc=default:ip=198.88.16.6:ht=ethernet:ha=02608C754292:bf=net030
ipl2:tc=default:ip=198.88.16.7:ht=ethernet:ha=0000C0FE6214:bf=net030
ipl0:tc=default:ip=198.88.16.9:ht=ethernet:ha=0000C0138314:bf=net030
ipend:tc=default:ip=198.88.16.10:ht=ethernet:ha=0000C0DEE710:bf=net03
halt:tc=default:ip=198.88.16.11:ht=ethernet:ha=0000C0CE5016:bf=net030
fc1:tc=default:ip=198.88.16.13:ht=ethernet:ha=0000C0AC9A14:bf=net030
ecs:tc=default:ip=198.88.16.15:ht=ethernet:ha=0000C08D6514:bf=net030
ds:tc=default:ip=198.88.16.18:ht=ethernet:ha=0000C0E15E14:bf=net030
siz1:tc=default:ip=198.88.16.1:ht=ethernet:ha=02608C172716:bf=net030
siz0:tc=default:ip=198.88.16.2:ht=ethernet:ha=02608C751977:bf=net030
ipl1:tc=default:ip=198.88.16.8:ht=ethernet:ha=02608C754301:bf=net030
#fc2:tc=default:ip=198.88.16.12:ht=ethernet:ha=02608C754301:bf=net030
fc0:tc=default:ip=198.88.16.14:ht=ethernet:ha=02608C285527:bf=net030
```

makes it a good tool for embedded-systems programming. It enables GNU-C to be built as a cross-compiler and -assembler for many architectures.

I use GNU-C mostly as a cross-compiler for the 68000 and ColdFire, Motorola's new architecture (see Tom Cantrell's "Motorola Lights ColdFire," *INK* 77), on Sun workstations. However, the 68000 cross-compiler can be built for almost any OS.

As a C compiler, it behaves as you'd expect, compiling old-style C as well as ANSI C into an object module. It can also compile C++ and Objective C and even provides a pretty complete run-time support for the 68000 architecture (e.g., floating-point-math emulation libraries). GNU-C also has many optimization switches and can optimize code for almost all the 68000 variants.

GNU-C is quite amazing, but the assembler is interesting, too. GNU-as can be configured to work as a 68000 assembler, accepting both Motorola and MIT syntaxes-the two prevailing assembler syntaxes for the 68000 family. It can be configured to generate different object file formats (e.g., COFF), which are all supported by the GNU linker.

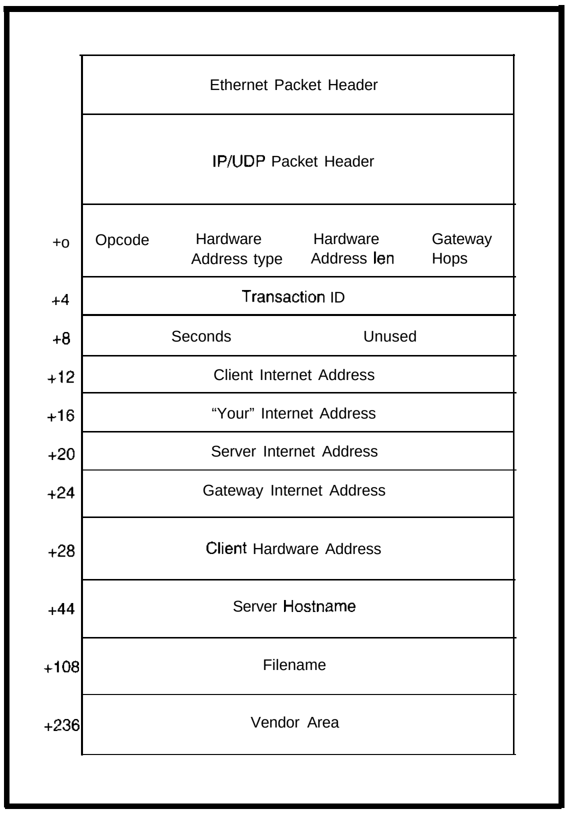


Figure 1—*The client broadcasts a BOOTP request (opcode = 1) on the Ethernet The serverthen sends a reply (opcode = 2) to the client with a//the information the client needs to boot a file from a server.*

There is also a source-level debugger which has remote debugging capability. So at this point, I can compile C files, assemble 68000 assembly-language modules, and link them all together into a program on my Unix workstation. Nice, but how do I get it into the 68030 system?

There are several options. I can generate a Motorola S-record hex file and download it over one of the serial ports on the 68030 system. But, this method takes a long time for any program bigger than a few kilobytes.

I can also extract an image of the program and its data and copy it to a floppy that can be used to boot. This option is pretty nice, but it requires a floppy drive on the workstation that can write floppy disks in "raw" format.

One of the fastest and most convenient methods is network booting. In Part 2, I wrote about the features of the monitor on my 68030 system and its ability to boot from the network. Here's how it's done.

NETWORK BOOTING

Without getting too in-depth about Ethernet, let me describe what goes on. Last month, I described the packet-level driver needed to send and receive packets to and from Ethernet.

Each packet sent on the net needs to have a destination address, which in the case of Ethernet, is sometimes called the hardware address. The hardware address is a serial number that's unique for each Ethernet card.

Since hardware addresses are assigned by the manufacturer, they're not so useful when it comes to sending a packet to another machine, unless it happens to be on the same network segment. The hardware address has no information about how to route packets between network segments.

RS-485 NOW!

Standard RS-485 Network Software

Deploy a rugged, proven Cimetrics RS-485 solution in just days. Software, full documentation, development tools. All backed by outstanding service and support from Cimetrics.

- Process Control
- HVAC & Lighting Control
- Point-of-sale
- Transportation
- Fire & Security
- Telecommunication
- Medical Equipment
- Laboratory Data Acquisition

Comprehensive Microcontroller Master & Slave Support

PC (DOS/Win/NT)	68HC05	PIC16C74
8051	68HC11	PIC17C42
8096	68HC16	H8/300H
80C196	68302	TMS370
80C186EB	68332	Z180
	68333	

Cimetrics TECHNOLOGY

55 Temple Place • Boston, MA 02111-1300
Tel: 617-350-7550 • Fax: 617-350-7552
Website: www.cimetrics.com

#135

$200 4A C Compiler?

You heard right. A quality C compiler designed for the 8051 microcontroller family, just $200, **including** the Intel compatible assembler and linker. A great companion to our fine Single Board Computers, like those below. **CALL NOW!**

552SBC

80C552 a '5 1 Compatible Micro
40 Bits of Digital I/O
8 Channels of **10** Bit A/D
3 Serial Ports (RS-232 or 422/485)
2 Pulse Width Modulation Outputs
6 Capture/Compare Inputs
1 Real Time Clock
64K bytes Static RAM
1+ UVPROM Socket
512 bytes of Serial EEPROM
1 Watchdog
1 Power Fail Interrupt
1 On-Board Power Regulation

Priced at just $299 in single quantities. Call about our 552SBC C Development Kit, just $448.

99 MHz 8051!

Our popular 8031SBC can now be shipped with Dallas Semi's hyperactive **DS80C320**, an 8051 on steroids. Averaging 3x faster than the standard 51, your project can really scream! **Call or email for pricing and brochures today!**

Other versions of the **8031SBC** hove processors with **on-chip** capture registers, **EEPROM,** IIC, **A/D** and more. Call or **email** for **a** list!

8031SBC as low as $49

Call for your custom product needs. Quick Response.

HiTech Equipment Corp.
9672 Via Excelencia
San Diego, CA 92126
(Fax: (619) 530-1458)

Since 1983

(619) 566-1892

Internet e-mail: info@hte.com
World Wide Web: **www.hte.com**

#137

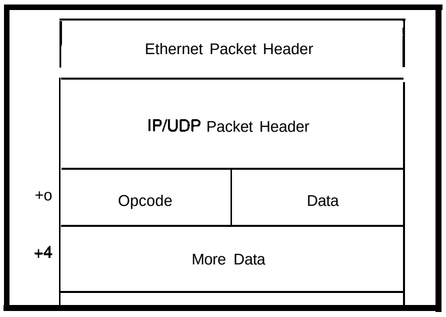


Figure 2—The client *sends a read* **request packet** *(opcode = 1), in which the filename is specified in the data. After the* **server** *responds with an acknowledge (opcode = 4), the client asks for data* **blocks** *(opcode = 2).*

On the Internet, routing is done with the Internet address. An Internet address is composed of two parts-a network number and a host number. I'm not going to discuss routing issues any further here, so let's just say that Internet addresses are assigned by network administrators to make the routing between hosts possible.

But, how does a machine figure out the hardware address of a host it wants to send a packet to? And furthermore, how does the machine discover its own Internet address?

I'll answer the second question first. The machine already knows its own hardware address.

The monitor uses two protocols to do this. The boot protocol (BOOTP) is used to discover the Internet numbers of the machine and server as well as the filename of the boot image to load. Figure 1 depicts a BOOTP packet.

The BOOTP protocol is simple. The monitor fills out a BOOTP request packet and uses the packet driver to transmit it. A special broadcast address ensures that all machines on the segment can see this packet, since it has no idea which machines are present on the local Ethernet segment.

A BOOTP server host has a table that maps the hardware address from which the packet came into an Internet address and a filename. These are then sent back to the BOOTP client that originally broadcasted the request. Listing 1 shows the BOOTP table for our lab.

The client receives the packet and extracts its Internet address, the Internet address of the server, and the filename to download for the boot image. The monitor then uses the trivial file transfer protocol (tftp) to read the file from the boot server.

To discover the server's hardware address, the monitor broadcasts an address resolution protocol (ARP) request packet. This packet basically asks, "Hey! Who out there has Internet address *xx.xx.xx.xx*?"

All hosts on the local segment can hear this packet. If their Internet address matches the one requested, they respond with a packet giving the hardware address. Figure 2 illustrates a tftp packet, and Figure 3 gives you a look at the ARP-packet format.

The boot monitor can now contact the server using its hardware and Internet addresses and download the requested file via tftp. Our BOOTP and tftp servers are usually the same machine (i.e., a Sun workstation).

The tftp protocol is fairly simple. The client requests a certain file and downloads it a packet at time. Each packet has a sequence number, and it is resent if an acknowledge isn't received within a timeout interval.

This network-booting scheme has been around for a while, and Unix workstations from many vendors have the appropriate BOOTP and tftn server programs-as part of their normal software distribution.

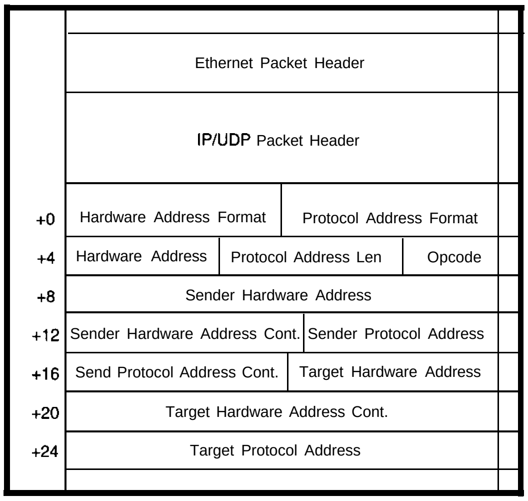


Figure 3—When *a host wants to find the hardware address for another host, it broadcasts a* **request** *packet (opcode = 1). The target host responds by* **filling** *in its hardware address and sending if back as a response (opcode = 2). The protocol address is the* **Internet** *address.*

Even though this boot method is fairly popular and reliable, there's one problem that makes it unsuitable for downloading students' programs in the lab. There's no privacy. Since the boot client has no way to authenticate itself except by its hardware address, the system must have provisions for students to only be able to download their own programs.

Furthermore, tftp has no authentication method, so it essentially acts as a public-access server. Also, the boot monitor has no way for students to log in remotely to edit and compile programs. Something is clearly missing.

Here's where the network monitor comes in. It enables the student to log in to their Unix account to edit, compile, and download their code in a secure manner over the network.

The network monitor is based on the ka9q network OS. This software, originally written by Phil Karn, is used by many amateur radio operators to communicate using packet radio. It implements a TCP/IP protocol stack with several clients like telnet and ftp.

I adapted this program to run on my MC68030 system by stripping out unneeded features and interfacing it to some of the drivers I've written. The students boot the network monitor through the network using the BOOTP and tftp boot process. Once it's running, they can telnet to their Unix account anywhere on campus.

Once they're ready to debug, they can download the image of their program into the 32-Kb SRAM, which is guaranteed to be available on the 68030 system, by using the normal Internet ftp protocol. ftp lets the student log in to their own account and download only their files.

Once their program is downloaded into SRAM, they can interrupt out of the network monitor via a front-panel push button, which sends an NMI, and enter the boot-PROM-based debugging monitor. The student can also resume the network monitor, assuming their code didn't disrupt the saved context of the network monitor.

CLASSWORK

When students take the computer-architecture lab, they've generally only been exposed to an introductory C programming course. So, they have quite a lot to learn.

They start with the basics of 68000 assembly language and architecture by doing exercises in which they fill in missing code segments or explain the state certain registers are in after executing instructions. The first lab exercise consists of logging in to their account and assembling a small example which they then download into the machine using the network.

Once they master a subset of the 68000 assembly language, they quickly take off and start coding more complex programs. One of the first really hard labs is the interrupt lab, in which they have to write an interrupt-driven keyboard driver based on a polled keyboard driver from an earlier lab.

They then also learn to interface C modules to their assembly-language modules which implement the interrupt service routine. The labs' complexity increases until they culminate into the final lab. The final lab consists of taking various interrupt-based I/O drivers and timer routines and implementing a game (e.g., Tetris or Missile Command).

FUTURE DIRECTIONS

One of the things I'd like to do with this system is create a more integrated debugging environment. I might do that by implementing some kind of network-based debugging interface to the 68030 and integrating it with GNU's source-level debugger. Another possibility is to redesign the 68030 system using a newer processor technology that implements a hardware debugging port. Some interesting processors include Motorola's ColdFire and PowerPC.

This article concludes my series on the 68030 system I built for our computer-architecture lab. I hope that I've fueled some interest in the development of open architectures and structures that are suitable for academics and others as well. ■

Ingo Cyliax is a research engineer in the Analog VLSI and Robotics Lab and teaches hardware design in the computer-science department at Indiana University. He also does software and hardware development with Derivation Systems, a San Diego-based formal-synthesis company. You may reach Ingo at cyliax@EZComm.com.

SOFTWARE

The schematics, PCB artwork, and sources for both monitors are available at <ftp.cs.indiana.edu/pub/goo/mc68030>. The computer-architecture class using the 68030 system for the lab has a Web site at <www.cs.indiana.edu/classes/c335home.html>. To find out more about networking, including RFCs on the protocol discussed, check out <ftp.digital.com/pub/net/info>. Phil Karn's ka9q network OS can be found at <ftp.digital.com/pub/net/ka9q>.

REFERENCES

D. Comer, *Internetworking with TCP/IP, Principles, Protocols and Architecture,* Prentice Hall, Englewood Cliffs, NJ, 1988.

W. Ford and W. Top, *Assembly Language and Systems Programming for the M68000 Family,* D.C. Heath and Co., Lexington, MA, 1989.

A.S. Tannebaum, *Computer Networks,* Prentice Hall, Englewood Cliffs, NJ, 1981.

J.F. Wakerly, *Microcomputer Architecture and Programming, The 68000 Family,* John Wiley & Sons, New York, NY, 1992.

SOURCE

MC68xxx, ColdFire, PowerPC
Motorola
MCU Information Line
P.O. Box 13026
Austin, TX 7871 l-3026
(512) 328-2268
Fax: (512) 891-4465
freeware.aus.sps.mot.com

I R S

422 Very Useful
423 Moderately Useful
424 Not Useful

FROM THE BENCH

Jeff Bachiochi

Nonintrusive Interfacing—Using Kid Gloves

Nobody has ever accused Jeff of growing up, especially his kids. Follow along as he unobtrusively updates an already modernized classic toy.

Stairsteps were a natural. Circles, on the other hand, were frustratingly elusive. Good coordination was a must. Hours of intensive creativity produced only a brief period of admiration. And then, it was gone with shake.

Have you guessed? You saw one on a previous cover of *INK.* It's red with white knobs. One knob draws horizontally, while the other draws vertically.

Yeah, that's it-the Etch-A-Sketch from The Ohio Art Company. You had one as a tyke, right?

The simplicity was elegant. Instead of drawing on a surface, the Etch starts by coating the surface with a reflective material and then proceeds to remove the coating as you draw.

As the knobs move their appropriate axes, the wiper moves at the intersection of the axes. The wiper dislodges the coating from the inside of the top transparent surface so it drops off the surface, producing a nonreflective line.

The only drawback was no permanent record of the masterpiece. And, then along came electronics. 'Ohio Art jumped on the "batteries are better" bandwagon and produced the Etch-A-Sketch Animator shown in Photo 1.

The Animator has an LCD screen of 30 high x 40 wide square pixels. Each pixel is about one-tenth of an inch. Two familiar knobs move the cursor (one square pixel] around the screen.

I felt right at home drawing stair-stepped circles just like I did when I was young. With square pixels, it's impossible to draw perfect circles.

Eight new pushbuttons adorn the electronic sketch pad. Three buttons set the creating mode-move, draw, and erase. With these, you move without affecting the pixels you pass over, leave a trail of "on" pixels, or leave a trail of "off" pixels using the x and y direction knobs.

The next three buttons-save, recall, and next-control frames. A

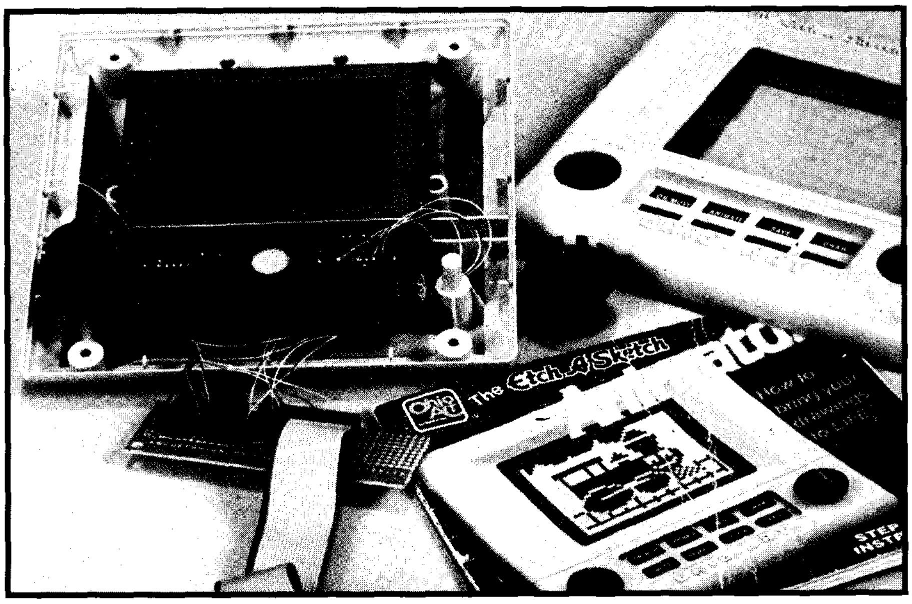


Photo I-The *Animator is a rugged electronic version of its predecessor, the Etch-a-Sketch. This photo shows control circuitry wired to the Animator, enabling external control without affecting its manual operation capabilities.*

frame is where you place your finished picture (you're allowed up to 12). The frame controls let you store the active picture to the present frame number, copy the last frame to the next frame, and move to the next frame.

The last two buttons are special effects. The reverse button changes the state of all pixels (on to off and off to on). The animate button enables you to play any of the 12 frames back in sequence. The sequence can be up to 96 frames in any order.

After three minutes of nonuse, the Animator turns itself off. As long as the batteries remain in the unit, all the frames are stored. Changing the batteries erases all the frames.

LOOK, BUT DON'T TOUCH

"Yeah, Dad, you can borrow it, but please don't ruin it. I still play with it." Kristafer, my youngest, reluctantly passes over his treasure.

I'm thinking, there must be a way to make use of this thing without damaging it. It has a great LC display. Where's my screwdriver?

OK. On the inside, it has what you'd expect. An LCD driver chip, a processor, and a few glue chips.

All the functions are multiplexed. Take the x-y control knobs, for instance. They're similar to rotary encoders. For each axis, an LED is aimed at two phototransistors.

The knob, which looks like one of those cookie-cutter-style hole saws, is placed over the receivers so the LED's light is blocked by large teeth on the knob. The receivers are strategically placed such that when the knob is turned, the LED's light hits one and then the other receiver.

Comparing the phases of the phototransistors' outputs tells which direction the knob is turning. The trick here is that the transmitting LED is only active for a particular time slot. So, the phototransistor's open-collector output is only actively low during that particular time slot.

To simulate the x-y knobs, I used four TTL outputs-a pair for each knob. By toggling the pairs of outputs in software, I create the changing phases, which are interpreted as rotating a knob one way or the other.

These signals are ORed (74HC32) with the control signal enabling the Animator's LEDs (see Figure 1), which creates time-slotted outputs. These outputs are buffered with open-collector drivers so they can be connected in parallel to each of the Animator's four phototransistor outputs.

The function buttons are much simpler to control since there is no direction involved. There are two rows of four buttons. One button—on/move-operates differently, so really, there are only three buttons in the top row of this matrix.

Different time-slot signals, active low, are applied to the rows. When a button is pushed, these time slots are transferred to the column pulling them low.

Again, I used four TTL signals to control these functions. A dual 2-4 open-collector decoder (74LS 156) worked nicely. Two of the TTL signals are used as address inputs, performing dual l-of-4 output selection. One set is used for each row.

The '156 has two sets of enables for each decoder. One set is fed by the time-slot signals to keep the outputs active only at the appropriate times.

The other set is controlled by my other two TTL signals-the key press enables. By selecting an address and enabling a row output, a time-slotted signal is placed in parallel with a button's column, faking a physical button push.

The last button I want to discuss is the on/move button. It has a double function and cannot be controlled with a simple open-collector driver.

When the Animator is in sleep mode, a condition it enters when activity has ceased for 3 min., all execution halts to save battery power. The processor outputs a high to one side of the on/move button. The other side goes into power-on circuitry.

To mimic the on/move button, I placed a FET

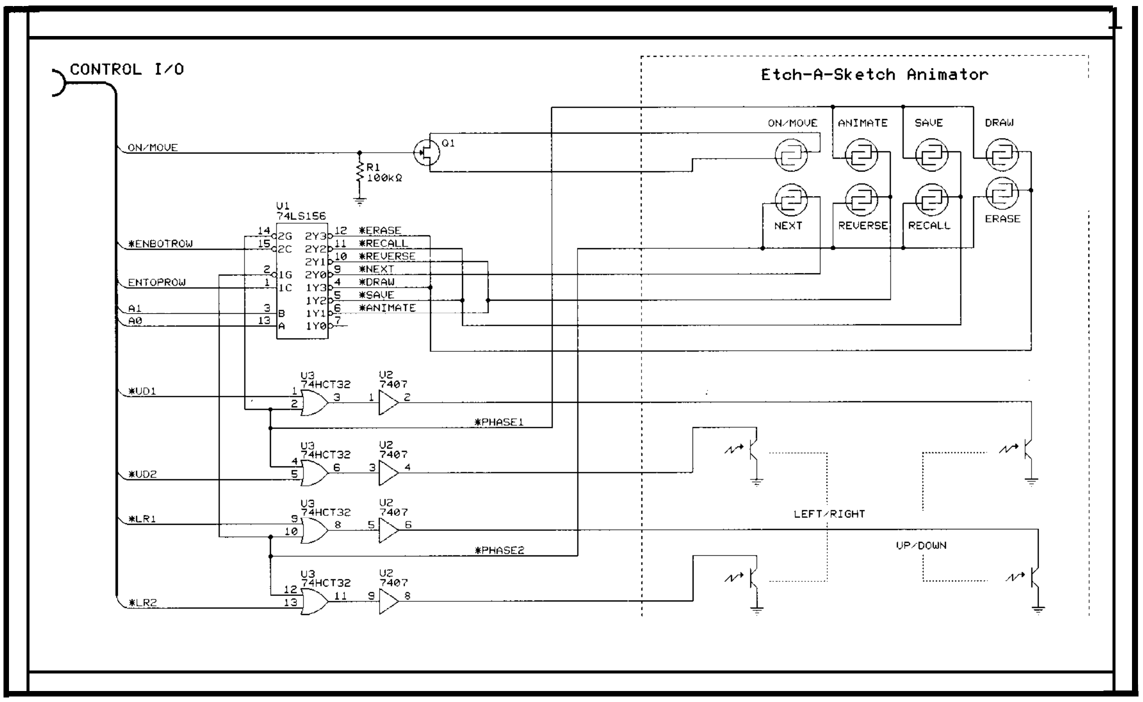


Figure 1--The I/O interface circuitry uses a nonintrusive approach that allows human and machine to coexist.

across its contacts, with its gatc biasing thc FET off. A ninth TTL signal pulses the FET's gate to imitate the button being pushed. I could have uscd FETs on all eight pushbuttons, but thc costs would have been considcrably higher.

FIRST BREATH

As you can imagine, this interface could bc controlled by, most any device. It simply nccds nine digital outputs. I chosc to use a Domino microcontrollcr to check for signs of lifc.

Thc Domino has a floating-point BASIC intcrprctcr with lots of code mcmory space. I first constructed routines to drivc the outputs correctly to cnablc each of thc ten functions the Animator would recognize-move, left, right, up, down, draw, crasc, rcversc, next, savc, recall, animate, crasc frame, and crasc animation.

Sincc this is an open-loop system, I wanted to makc surc no commands would bc missed. Therefore, each function must execute no fastcr than thc minimum time-slot repetition ratc (~22ms) or thcrc's a chance it may not bc seen. On thc othcr hand, bcing too slow wasn't a problem since it's legal to hold a button as long as you want.

After each routine was functioning properly, I wrote a test scqucncc to scan the wholc LCD frame. First, simply moving thc cursor through the complete frame would give me thc minimum timc necessary to scan a frame. It would also give mc the basic code necessary for navigating a framc.

Ncxt, using this code and alternately drawing and erasing cach pixel in a checkerboard pattern, I could see what kind of maximumtime was necessary to draw an entire frame. Unlike a television scan, I don't waste time returning. to thc beginning of each row. At the cnd of each row, I just drop down a row and draw in the oppositc direction.

The minimum time to scan thc complete frame is 300 s. Whcn draw and erase commands wcrc added to thc scan to produce a chcckcrboard pattern, the time increased to 360 s.

I figurc this could bc dccrcascd two-thirds by rcducing thc control commands from the present in-line calculatcd port values to the constant equivalents. The minimum calculated scan time based on-thc minimum time slot repetition rate would bc:

$$40 \times 30 \times \text{timc} \times 4$$

which is:

$$\frac{\text{pixels::}}{\text{row}}\ \frac{\text{pixels}}{\text{column}} \times \text{min. scan timc} \times \frac{\text{pcriods}}{\text{stcp}}$$

TEXT AND GRAPHICS FRAMES

A 6 x 6 block of pixels is of sufficicnt size to hold any alphanumeric character (including a blank column for interletter spacing or a blank row for interline spacing). Many letters require only four columns.

Thc 40 x 30 pixel screen can thcrcforc hold about 40 characters. Since thc frames are drawn as a uriit, character sizc can be mixed and can frccflow on the frame.

As a point-of-salt display, the animation can be used to automatically

PROTEUS The IVth Generation

PROTEUS **IV for Windows combines Advanced Schematic Capture, Analogue & Digital Simulation with PCB Layout**

Runs on Windows 3. I, 95 8 NT

Schematic Capture

- Told control of drawing appearance.
- Mouse driven place. edit, move & delete.
- Netlists, Electrical Rules Check.

Circuit *Simulation*

- Demand Driven Simulation with Recording.
- Mixed Mode Anologue & Digital Simulation

PCB Layout

- Auto Component Placement!!!
- Rip-Up & Retry for 100% Routing of boards.
- Layers: 16 copper, 2 silk. 4 mechanical.

Most Powerful EDS System on the market!!!

R4 SYSTEMS INC.
1100 GORHAM ST. Suite 11B-332
NEWMARKET ONTARIO
CANADA L3Y 7V1
905 898-0665 FAX 905 898-0683
BBS 905 898-0508 (9600,8,N,1)

FREE DEMO
WRITE OR CALL
TODAY

Download Demo - http://www.r4systems.on.ca
Internet email info@r4systems.on.ca

Think ***Circuit Cellar INK*** is a great technical resource?

Do you know anyone who might appreciate it just-as much?

Most subscribers get to know us by reading someone else's copy. Satisfied readers are our best supporters.

We'll send you a bunch of complimentary copies so you can show us off. Give us your name, where you live, and how many you need.

Contact Rose Mansella
(860) 8752199
rose.mansella@circuitcellar.com

Thanks for your help.

Shipping costs for all foreign and Canadian complimentary copies must be prepaid by credit card.

scan through any of the 12 frames of graphics in any 96-frame sequence. Or, the Animator may be commanded to increment the frame using a variety of interframe pauses based on the material displayed. This option gives text-based screens more display time.

To prevent the Animator from powering down into a power-saving mode, a command must be sent at least once every couple minutes. Various keys can be used (none of which will alter the presently displayed frame).

FRAME INPUT MODE

Although the Etch-A-Sketch Animator is capable of storing 12 different frames, like its ancestor, your work is gone forever once you erase a frame to draw a new creation. Frame input mode using the Domino's serial input port can accept 30 strings of 40 characters each.

Using a text editor, you can create a file that uses "." (period] and "X" to signify an erased and drawn pixel. A carriage return at the end of every row delimits the line.

You can quickly see and edit the image, and then-voila-it's saved permanently as a file. The file can be loaded into the Domino and placed into any of the Animator's 12 frames. A serial EEPROM may be used as local nonvolatile storage.

JOY STICKING

Because the Domino has an optional two-channel ADC, an x-y joystick can be connected for better local control. Apply +5 V across the joystick's potentiometers, and route the wipers into the two A/D inputs.

The A/D conversion values are used to decide which direction pulses to send to the Animator. Now, this is more like playing a video game—unless you remember Pong.

So, what did I accomplish? For the most part, it was an exercise in interfacing-the nonintrusive control of an established system.

At some point, you may be asked to provide additional functions to a system where you're not allowed to alter the operation of an existing system. Redesigns are not always possible.

And as an engineer, you must be ready to accept the challenge-no matter what the constraints. Even if they come from a nine-year-old. ❑

Jeff Bachiochi (pronounced "BAH-key-AH-key") is an electrical engineer on Circuit Cellar INK's *engineering staff. His background includes product design and manufacturing. He may be reached at jeff.bachiochi@circuitcellar.com.*

SOURCES

Domino
Micromint, Inc.
4 Park St.
Vernon, CT 06066
(860) 871-6170
Fax: (860) 872-2204
www.micromint.com

I R S

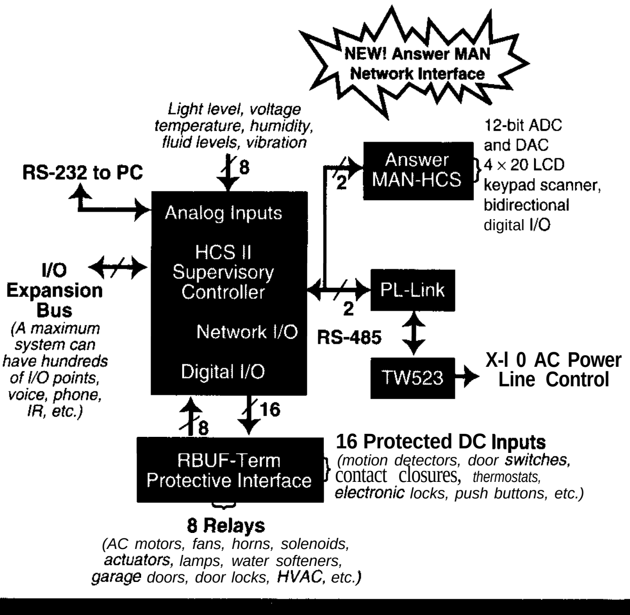
425 Very Useful
426 Moderately Useful
427 Not Useful

CIRCUIT CELLAR HCS II

Affordable Home Automation and Security

Our concept of building a home control system is based on the principles embodied in building an original Heathkit. Besides giving you a project which details everything from the theory of operation to instructions for fabricating a functioning system, we give you the sense of confidence that comes from success.

HCS
POWER


NEW! Answer MAN Network Interface
Light level, voltage temperature, humidity, fluid levels, vibration
8
RS-232 to PC
Analog Inputs
HCS II Supervisory Controller
Network I/O
Digital I/O
I/O Expansion Bus
(A maximum system can have hundreds of I/O points, voice, phone, IR, etc.)
2
Answer MAN-HCS
12-bit ADC and DAC 4 x 20 LCD keypad scanner, bidirectional digital I/O
PL-Link
2
RS-485
TW523
X-10 AC Power Line Control
8
16
RBUF-Term Protective Interface
16 Protected DC Inputs
(motion detectors, door switches, contact closures, thermostats, electronic locks, push buttons, etc.)
8 Relays
(AC motors, fans, horns, solenoids, actuators, lamps, water softeners, garage doors, door locks, HVAC, etc.)

The HCS II is designed with an economical central core and an expandable network. A basic system consists of the HCS II central processor, which incorporates direct analog and digital I/O with real-time event triggering; a PL-Link wireless X- 10 AC power-line interface which has sixteen contact-closure/voltage inputs and eight 3-A AC/DC relay outputs for solenoids, motors, lamps, alarm horns, and so forth.

The house event control sequence is written on a PC in a unique, user-friendly control language called XPRESS and stored on the HCS in nonvolatile memory. Building them is as easy as 1,2,3.

Of course, we have lots of extra stuff like our new Answer MAN network module when you want to expand the basic system. The full HCS has phone and modem interfaces, infrared remote control, voice synthesizers, LCDs, and much more.

A Circuit Cellar HCS II is not for passive people and definitely not for the masses. It is for a special technical breed that wants a challenge with a tangible reward. Discover the world of manageable and affordable home automation and security.

Circuit Cellar, Inc.
4 Park St., Vernon, CT 06066 • (860) 875-2751
Fax: (860) 871-0411 • www.circuitcellar.com

★ AFFORDABLE HOME AUTOMATION - HCSII ★

SILICON UPDATE

Tom Cantrell

USB Micro

Cables are the bane of any wannabe neat and tidy PC setup. As USB becomes more and more popular, PCs will begin losing some of those cables. Tom checks out the latest chip that should help the USB cause along.

I suppose it's a bit ironic that, sitting in the heart of Silicon Valley and writing for one of the highest of high-tech mags, my own personal stable of computers practically qualifies as a museum.

In fact, I wrote this month's "Silicon Update" on the same machine I used for my first column lo those many years ago. It's a Mac 11x with a whopping 8 MB of RAM running at a blazing 15.6672 MHz!

My wordsmithing has been well-served by this puppy, thanks to a 21" mono (fewer bits to push around) CRT. Since for me, the writing process is actually 99.9% staring at the screen, I've never really missed the MIPS. The IIx is supplemented by a truly muttly LC that, hopped up (liberally speaking) with an accelerator, barely manages to cope with the 'Net and a scanner.

On the PC front, the situation is even more laughable since I only use a PC for simple engineering stuff and don't expect it to handle more than a bit of cross-ASM and BASIC twiddling or EPROM programming from time to time. The last big development on my PC front was when I upgraded from '286 DOS to '386 DOS + Windows 3.1 a few years back.

Heaven forbid you should look in closet and find the original "toaster" Mac (512 KB of RAM and *serial* add-on hard disk). Or, maybe you'll unearth a true relic-the IMSAI 8080 with 8" floppies [and all those lamentably lost switches and LEDs}.

I suppose I'm the PC exec's worst nightmare. More consumers like me, and the PC market is going to be about as exciting as the typewriter biz.

Although I appreciate the challenge of getting the most out of old gadgets, I know it's only a matter of time before my hand (and wallet) is forced. In my case, I'm hoping to stall until a) I see whether Steve J. can somehow pull a rabbit out of his Mac and/orb) Bill G. gets his Windows 9? act together so I can bypass at least one whole generation of upgrade hassles.

A key inflection point on the PC front revolves around the next-generation I/O scheme that relies on the Universal Serial Bus (USB) and Firewire [IEEE 1394) interfaces. I'd really rather not muck with the ungracefully aging mishmash of cables and connectors cluttering today's PCs.

ASLEEP AT THE WHEEL

Migrating from old standards to new is often characterized as a chicken-and-egg affair. What people forget is that you can't just stick an egg on the shelf and reasonably hope to get a chicken anytime soon.

On the USB front, progress requires both a cheap and plentiful collection of hardware add-ons (e.g., mice, joysticks, keyboards, etc.) and an OS that knows what to do with them.

In this case, there are plenty of eggs, thanks to Intel's ever-increasing ability to decide what PC hardware looks like. Once they started building USB into their chipsets and motherboards, it didn't take a guru to see the future. Thus, as I write this (and certainly by the time you read it), it's actually getting harder to find a PC without USB ports on the back than one with.

However, even if you've got 'em, you may not know it. Short of piddling snippets of code in arcane OSRs (operating-system service releases), Microsoft isn't really blessing USB until the next OS release (Windows 98).

And without significant OS support, add-on device suppliers and retailers know that trying to force-march

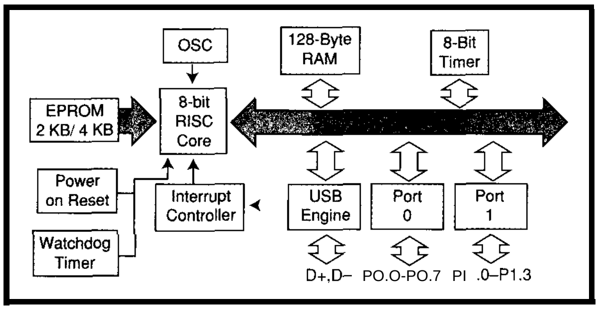


Figure 1—*The CY7C6x family targets low-speed (1.5 Mbps) USB apps by combining an OTP EPROM-based RISC core, I/O, timers, and a direct-connect USB interface.*

users onto USB won't happen. Without a chicken, the USB eggs aren't hatchable.

Anyway, short of plaintive calls to Redmond, there's nothing much you or I can do to influence developments on the OS front. But, observing that these software developments eventually muddle through, it's only a matter of time before USB finally becomes the big deal everyone hopes for.

With that in mind, the hardware guys are still laying eggs at a furious pace. Let's take a look at an interesting example-the CY7C6*x* USB Micro from Cypress Semi.

WELL FARED

Perusing the flyer from a local computer shop, I find mice and joysticks averaging about $20, with many under $10 and even one for $0 (fine print points out you do have to cough up for the stamp to get the rebate). Obviously, no $3-5 chip is going to cut it in such a cost-sensitive market.

So, before getting into the bits and bytes, let me point out that the Cypress chips start at under $2 in volume and the company claims prices will bust the buck barrier once things get rolling.

These entry-level parts only target low-speed (1.5 Mbps) apps. (Cypress also offers higher end hub ICs, which handle the 12-Mbps segment.) Those of you who aren't up to snuff on these and other USB basics can check out my "Oh, Say Can USB!" article (*INK* 74) and log on to the Web (www.usb.org).

The variety of low-speed offerings are differentiated by OTP EPROM capacity (2-8 KB), pin count (18- to 48-pin), and package (DIP, SSOP, SOIC).

All can be described by taking a look at the minimalist (2-KB EPROM, 128-byte RAM, 18 pin) CY7C63000, since the higher end chips simply add general-purpose I/O lines.

As shown in Figure 1, the chip combines an 8-bit MPU with a USB Serial Interface Engine (SIE) and transceivers (for direct cable connection) with miscellaneous accessories such as watchdog and general-purpose timers, power-on reset, and clock generator. The latter relies on a PLL to boost an external 6-MHz clock to 12 MHz on chip.

For lowest cost, a ceramic resonator can be used, but I'd watch out for accuracy and drift relative to the USB specifications. Easing the use of a crystal, Cypress eliminates the need for the external bias resistors and feedback capacitors typically required. A small touch, but one welcome when your gadget sells for $0.

The pinout in Figure 2 is simple enough, composed of power (V,, for operation, V_{PP} for programming), ground, 12 I/O lines (8 PORT0 and 4 PORT1), the differential USB signals D+ and D-, and two clock lines (XTALIN and XTALOUT).

Photo 1 -Meeting the *under-a-buck price challenge calls for an extremely small die-an achievement made possible by the tiny size of the CPU core.*

The only nonobvious pin is CEXT, which is a dual-function open-collector output and Schmitt-trigger input. It's intended to act as a wake-up alarm clock to take the chip out of low-power (100 µA) suspend mode.

The idea is to connect an external RC, discharge it using CEXT as an output, switch it to input mode, and go to sleep. Once the pin charges up, the chip will wake up in 256 us. Cypress calls the feature Instant On, but it isn't so much the speed as the fact that it overcomes the lack of a reset pin and bypasses the housecleaning (i.e., register init) associated with power-on reset.

Reflecting the targeted apps, the I/O lines deliver capabilities uniquely targeting pointing devices. For instance, all the pins feature Schmitt-trigger inputs and optional pullups (handy for buttons), while four of them (PORT1) offer LED driving capability to accommodate the typical LED/phototransistor-based optical mouse setup.

To tailor power consumption and slew rate, the outputs include a unique 16-level programmable drive capability, shown in Figure 3. And as inputs, each line can act as an interrupt with programmable edge polarity.

The 11 -bit timer isn't general purpose but rather is dedicated to specific time constants (i.e., it's read only]. Running at 1 µs (i.e., ⅙ the crystal frequency), the seventh and ninth bits generate periodic 128-µs and 1.024-ms interrupts, respectively. The latter, further prescaled by 8 (i.e., 8.192 ms) serves as the timebase for the watchdog timer.

Also, 16 bytes of the on-chip RAM (128 or 256 bytes, depending on the chip] are dedicated as 8-byte FIFOs for the two endpoints supported by the chip. Endpoint 0, required of all USB devices, is used during initialization to enable the host PC to determine what is out there-a process known as enumeration. Endpoint 1 is where the bulk of the action occurs.

The rest of the RAM handles separate program and data stacks with user variables sandwiched in between.

REDUCED-IN-SIZE COMPUTER?

The Cypress MPU core reflects the latest RISC trend-namely, "Call it a RISC, no matter what it is." In fact, with an accumulator (A), single index register (X), condition codes (C and Z), and variable-length (1–2 byte, 4–14 clock) instructions, it looks more like a 6805 or 6502 than anything found in a computer-science book.

Anyway, I've long gotten over any feelings of righteous indignation over abuse of the RISC term, and Cypress certainly isn't alone in promulgating such dubious marketspeak. RISC, SHMISC-who gives a hoot? These days, all chips are either an 'x86, RISC, or VLIW, the irony being the 'x86s are themselves morphing into RISC (and soon, VLIW?) inside.

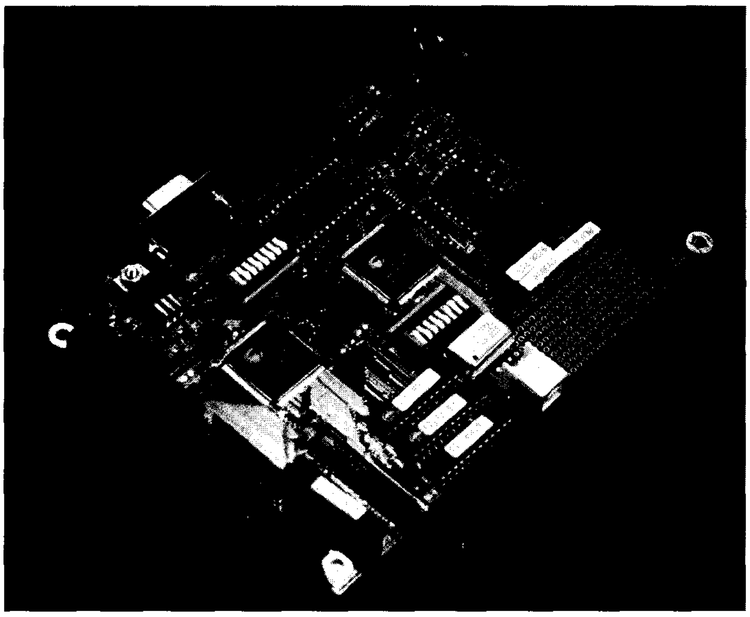


Photo 2—*The Cypress* **Development System relies on** *PLDs* **to achieve functionality** *between* **a** *low-end EV* **board** *and a full-featured ICE.*

The good news is, with only a few dozen instructions and three measly addressing modes [i.e., direct, indexed, and immediate), this puppy is real easy to understand. It should not take anyone beyond neophyte stage more than about 10 minutes in the datasheet to qualify as an expert.

Of course, beyond the typical moves, branches, and ALU ops, there are a few quirks to keep things interesting. For one, you as a programmer only have direct access (via JMPs and CALLs) to the lower 12 bits of the PC. For chips with more than 4 KB of ROM, this means jumping through a few hoops.

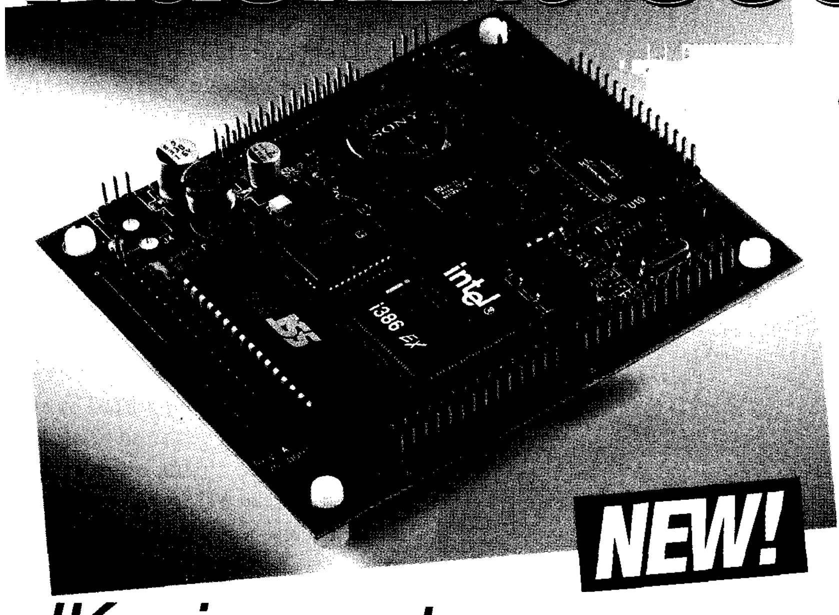
Furthermore, the PC doesn't automatically roll over past 8 bits, so a special instruction (XPAGE) is provided to increment bit 9. Fortunately, an assembler option automatically inserts X PAGE [with aligning N 0 P, if necessary) at each page boundary so you don't have to keep track of it.

Alternatively, those turned on by clever optimization can stop the op-

386EX Embedded Controller with DOS *Complete, Ready-to-Go, No Hidden Costs!*

FlashLite 386EX

$99 Developer's Kit includes BORLAND C 2.0 Starter Kit



- Includes DOS, utilities, flash file support.
- 32-bit processor runs 386 protected mode
- Complete COM1/COM2 PC compatibility
  - 386EX / 25 MHz
  - Up to 512K SRAM, up to 512K Flash
  - Sockets to add 512K Flash/RAM/EPROM
  - 2 Serial Ports
  - 24 Parallel I/O lines
  - Watchdog Timer
  - 3 Counter/Timers
  - 3 free interrupt lines
  - Battery-backed RTC
  - 2 DMA Channels
  - High-efficiency switching power supply
  - IBM PC speaker
  - Program in Borland C or QuickBasic

QTY 100 **$197**
$239 QTY 1
includes 128K RAM 512K flash & DOS

JK microsystems +510-236-1151 +510-236-2999 fax www.jkmicro.com
Cost effective controller? for industry

Fast, low-cost A/D converter and relay boards are also available.

#139

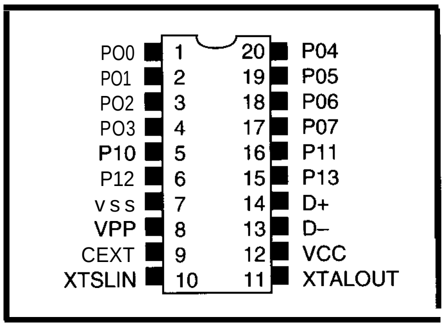


Figure 2—*The pinout of the 20-pin version shows 12 general-purpose I/Os (PORT0 and PORT1), power and clock* **connections, the differential** *USB* **pair** *(D+ and D–), and CEXT, which delivers a periodic wake-up call when connected to an external resistor.*

tion, make sure their loops are exactly 256 bytes, and voilà, "Look Ma, a loop with no JMP."

A more useful streamliner-the IPRET instruction-writes the accumulator to an I/O port (I), pops the old accumulator off the stack (P), and returns (RET), replacing the three instructions otherwise needed to skirt the accumulator bottleneck.

In fact, in a refreshing departure, Cypress even badmouths the conventional posture (if not wisdom) of HLL, pointing out that ASM is the way to go. I don't know how long their PR machine can hold out against the clamor for C, but I give 'em Brownie points for trying.

Bringing to mind the original VW bug, the CPU core may not have a lot of horsepower or fancy options, but it's certainly a miserly silicon sipper. Just take a look at Photo 1 to see how tiny a share of real estate the CPU gets—more astounding when you realize the entire die is less than 0.1" on a side!

DRIVER TRAINING

Getting up to speed with the Cypress chips is easy, thanks to their $495 USB development board (see Photo 2). The unit delivers functionality somewhere between that of the typical low-end ROM monitor EV board and a full-featured ICE.

Like the former, it relies on downloading your program into RAM and delivers full-speed operation, but only when not otherwise occupied handling a breakpoint or dumping a watch variable. However, it does exhibit ICE-like capabilities in that there's no memory or I/O resources stolen from your program-a key point since forgoing any of the chip's limited memory space would likely cramp your style.

The cute trick making this all possible is that the unit isn't built around a standard CY7C*x* part. Indeed, it uses the equivalent of a bondout chip often found in full-featured emulators. I say "equivalent" because, instead of a monolithic custom IC, the setup uses a pair of the company's high-end PLDs programmed to deliver both the functionality of the USB chip and ancillary logic (e.g., the UART that connects the board to a PC).

This configuration permits internal signals (e.g., address, data, int request and acknowledge, etc.) that are inaccessible on a standard chip to be brought out to headers. So, you can hook up a logic analyzer or trace buffer to track low-level operations.

There are a few minor gotchas. For example, the general-purpose I/O lines' programmable current-drive feature isn't available nor are the internal pullups. These minor differences can easily be handled as last-second software bolt-ons, and they certainly aren't showstoppers when developing the guts of your applications.

Closing the hood for now, the bottom line is that the board delivers the usual and adequate mix of capabilities [e.g., breakpoints, single step, memory and I/O access, etc.). Nothing too fancy, mind you (i.e., no breakpoint qualifiers), but it's enough to get your USB app cranked up and rolling.

On the software side, a workmanlike Windows (3.1 and 95 compatible, thank you) front end does the job without a lot of muss and fuss.

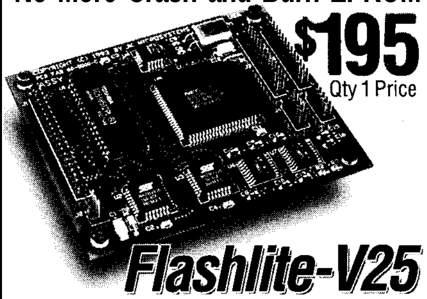
I did have a bit of trouble during installation. My floppy choked on one of the files, but I simply dismissed the ominous-sounding error messages with a thoughtfully provided `ignore` button. (I've encountered other software that punches you out with no recourse in similar circumstances.)

I paused just briefly before jetting past the '486-or-above warning in the documentation. The fact that I got past those potholes, and haven't encountered any since, is a good sign. Overall, the software seems robust and easy to use.

DOS-ON-BOARD

CONTROLLERS AT 8051 PRICES

Use Your PC Development Tools
No More Crash and Burn EPROM



***DOS Single* Board Computer**
with FLASH Memory 'disk drive'

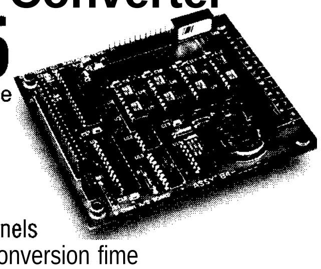
✓ 8/10 MHz CPU	✓ 2 DMA channels
✓ 512K bytes RAM	✓ 2 Timers
✓ 256K/512K FLASH	✓ 4 Interrupt Lines
✓ 2 Serial Ports	✓ 8 Analog Inputs
✓ 20 Parallel I/O Lines	✓ Xmodem Support

INCLUDES DOS & Utilities

USE YOUR TURBO C COMPILER OR QUICKBASIC COMPILER
SAVE TIME, MONEY AND HEADACHES

A/D Converter

$95
Qty 1 Price

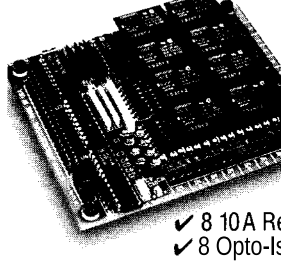


✓ 12 Bits
✓ 8 Channels
✓ 6 µs Conversion fime
✓ Clock/Calendar Option
✓ Includes Drivers & Apps.

GET YOUR EMBEDDED CONTROLLER PROJECT RUNNING *FAST!* WITH THESE ACCESSORIES

Relay I/O

$139
Qty 1 Price



✓ 8 10A Relays
✓ 8 Opto-Isolated Inputs

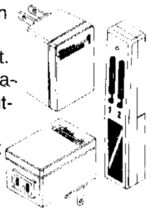
JK microsystems
Cost Effective Controllers for Industry

TO ORDER (510) 2364151
FAX (510) 236-2999—email: jkmicro@dsp.com
Visit our WEB site-www.jkmicro.com
1275 Yuba Ave., San Pablo, CA 94806

#144

ALL ELECTRONICS CORPORATION

RF Remote Control



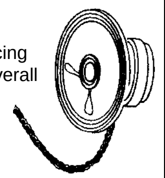
"Total Command"™ by Leviton
Operate TV, lamps and other appliances from up to 100 feet. Includes two receivers on separate frequencies and a two button hand-held transmitter that controls them. Requires 9 volt battery (not included).
CAT # TCR-1
$17.95 each

Ultrabright LED

PAINFULLY BRIGHT RED LED
2500 to 4000 mcd @ 20 ma. T 1 3/4 (5 mm diameter) red LED. Significantly brighter than conventional LEDs. Water clear in off-state.
CAT # LED-42 2 for $1.20

10 for $5.00 • 100 for $45.00
1000 for $400.00

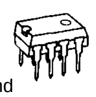
2 1/2" 8 Ohm Speaker



Panasonic # EAS65P76A3
8 ohm 0.4 watt speaker with soft-edge paper cone producing good sound quality. 0.93" overall depth. 2.25" wire leads with 2 contact (0.1" spacing) socket connector.
Large quantity available.
CAT # SK-255
$1.00 each

10 for $8.50
100 for $70.00

Low Power Audio Amp



Motorola MC341 19P
Low power audio amplifier suitable or speakerphones or talking picture frames. Operates on 2 - 16 volts and drives speakers of 8 ohms or greater. Output power exceeds 250 mW with 32 ohm speaker. Hook-up sheet. Large quantity available.
60¢ each
CAT # MC34119P
50 Pieces - $25.00

ORDER TOLL FREE
1-800-826-5432
CHARGE ORDERS to Visa, Mastercard, American Express or Discover

TERMS NO MINIMUM ORDER Shipping and handling for the 48 continental U S A $5 00 per order All others including AK. HI, PR or Canada must pay full shipping All orders delivered in CALIFORNIA must Include local state sales tax Quantities Limited NO COD Prices subiect to change without notice

CALL, WRITE FAX or E-MAIL for our FREE 96 Page CATALOG
Outside the U.S.A. send $3.00 postage.

MAIL ORDERS TO:
ALL ELECTRONICS CORPORATION
P.O. Box 567
Van Nuys, CA 91408
FAX (818)781-2653
e-mail allcorp@allcorp.com
internet - http://www.allcorp.com/

#145

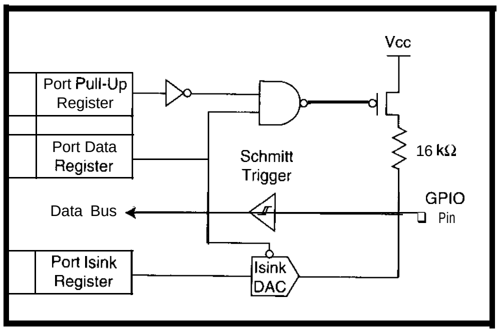


Figure 3—*The general-purpose I/O lines (PORT0 and PORT1) are especially versatile, including programmable pullups, Schmitt-trigger input conditioning, and programmable current drive. In addition, PORT1 can drive LEDs (up to 24 mA per line, 60 mA total), and all lines can serve as interrupt inputs with programmable polarity.*

Some might argue the 19.2-kbps communication rate is a little poky, since every second seems to count when you're in the middle of an edit-ASM-download-debug frenzy. The small size of programs lets Cypress skate by on this one.

I do suggest that they (and all others who use PC serial connections) include a port-finder feature that simply sits in a loop waiting for you to get your cable, gender changer, and port lashup right. I had to spend a bit more than the usual few minutes getting connected.

The package also includes an assembler (DOS based, which is fine with me) and particularly helpful example programs, including one for a mouse. Anyone who's ever delved into a new I/O protocol, particularly one as complex as USB, knows that having an actual example to study is much better than simply wading through the spec from scratch.

In fact, the `MOUSE.DOC` file gives directions on how to hack a mouse by removing the microcontroller and jacking the Cypress chip in (i.e., connect to the LEDs, phototransistors, and buttons). If all goes well, you can connect the whole mess to a USB-enabled PC, subject to the fine-print caveats about having the right betas and OSRs, tweaking the registry, and so on.

ALL ABOARD

The Cypress chips aren't the first targeting USB, and they won't be the last. Nevertheless, the fact that they're real-and real cheap (not to mention nicely supported)-gives them a leg up.

Remember, even if you're not in the mouse or joystick biz, USB is going to be a big deal. For instance, perusing *INK*, you'll see plenty of ads and articles describing myriad data-acquisition and process-control gizmos that connect to a PC serial port, all of which are likely candidates for hopping on USB.

Keithley Instruments, for example, has already announced the Smartlink family of USB-based distributed I/O modules. And they're priced the same as their older RS-232 counterparts.

Yes, departure of the USB bus is a little overdue. But once the driver wakes up (perhaps after gulping down some Java?), expect to see the USB pedal hit the metal. ❑

Tom Cantrell has been working on chip, board, and systems design and marketing in Silicon Valley for more than ten years. You may reach him by E-mail at tom.cantrell@circuitcellar.corn, by telephone at (510) 657-0264, or by fax at (510) 657-5441.

SOURCES

CY7C6x USB Micro
Cypress Semiconductor
3901 N. First St.
San Jose, CA 95134
(408) 943-2600
Fax: (408) 943-6848
www.cypress.com

Smartlink
Keithley Instruments, Inc.
28775 Aurora Rd.
Cleveland, OH 44139
(216) 248-0400
Fax: (216) 248-6168
www.keithley.com

USB Implementers Forum
2111 NE 25th Ave., MS/JF2-56
Hillsboro, OR 971245961
(503) 264-0590
Fax: (503) 693-7975
www.usb.org

I R S

428 Very Useful
429 Moderately Useful
430 Not Useful

PRIORITY INTERRUPT

Your Unbiased Advocate

It was a tough job, but somebody had to do it. What am I talking about? Judging a contest, of course.

Believe me. I don't volunteer gracefully. As an unbiased advocate dedicated to the low-cost dissemination of embedded control technology, I'm expected to participate. But, as a sometimes bitter and warped overkill-conscious hardware-oriented design engineer, they better watch out.

All kidding aside, being a contest judge sounds like a glamorous job. It isn't! Take my current experience with INK's just-concluded *Embedded PC* Design Contest. Next month, we'll have a write-up on the entries, but I'll tell you now that Mark Roberts is the big first-prize winner. It also turned out that he won third prize as well! If you think that's strange, then how do you feel about the second-prize winner, Brad Reed, also winning a fourth-place honorable mention?

Somewhere between my saying to Janice, "You've got to be kidding!" and "Who the hell's going to believe this?," I blotted my brow and said, "I guess it's editorial time."

I swear on a stack of Pentiums that the results of the contest are absolutely honest and the fact that two winners walked off with over half the prizes is purely coincidental. The judges never saw the entry forms, and all the contest materials we reviewed had the contestant's name or other identifying features removed. While we, as judges, might have seen some similarity in the writing style and quality of multiple single-author entries, I made a special attempt to judge each individually. I'm sure the others did the same. While multiple entries from a single contestant were not typical, there were a significant number. In fact, two industrious contestants submitted three.

The reality of the outcome is that the quality of the entry spoke for itself. Mark Roberts won because he followed the rules of the contest and executed it precisely. If he had the formula correct enough to win first place, it is only certification of his credible talents that a second entry following the same formula would be an equal contender. Similarly, Brad Reed zeroed in on the correct recipe for winning. The result was a real fight for the top spots.

While most contest rules are basic, the *EPC* contest rules were specifically intended to promote participation rather than excessive entry expense. Unlike the physical-demonstration design contests that we've sponsored in the past, an *EPC* contest entry consisted of a detailed engineering specification instead. As you might guess, the content and quality of these submissions varied greatly. It's only logical that someone familiar with writing specifications or proposals would have a leg up on the process.

Nonetheless, when there is $10,750 in prize money on the table, a review of the process is always in order. Given the remarkable conclusions, you can bet we reviewed it quickly. In the end, my opinion is that the judges were a damn picky lot. Perhaps because the judges are professionals with serious credentials and extensive embedded-control knowledge, they evaluated these specifications as real proposals. While I should have expected nothing less from them (or myself), they reacted severely to contestants who tended toward overkill in their selection of hardware components or who chose software with inadequate capability, regardless of how unique or daring the concept. The judges sifted through the piles of paper and simply selected specifications worthy of being presented in a real company environment.

It's obvious that a computer professional used to the technical-proposal process and a good writer may have a decided advantage. As you might have guessed, a crash-and-burn solder-sniffing engineer (like myself) has a different mindset.

In the end, however, the sponsors' goals were achieved. The winning entries demonstrated relevant use of the sponsors' products and descriptions of the winning entries will certainly pique reader interest. If you sat on the bench for the *Embedded PC* Design Contest, you have another opportunity with Design98. Of course, the "unbiased advocate" gets drafted again as a tie-breaker judge. Sometimes I wonder if I should be a bit more biased.

Steve

steve.ciarcia@circuitcellar.com