EMBEDDED PC SPECIAL SECTION: EMBEDDED PC

CIRCUIT CELLAR INK®

THE COMPUTER APPLICATIONS JOURNAL

#81 APRIL 1997

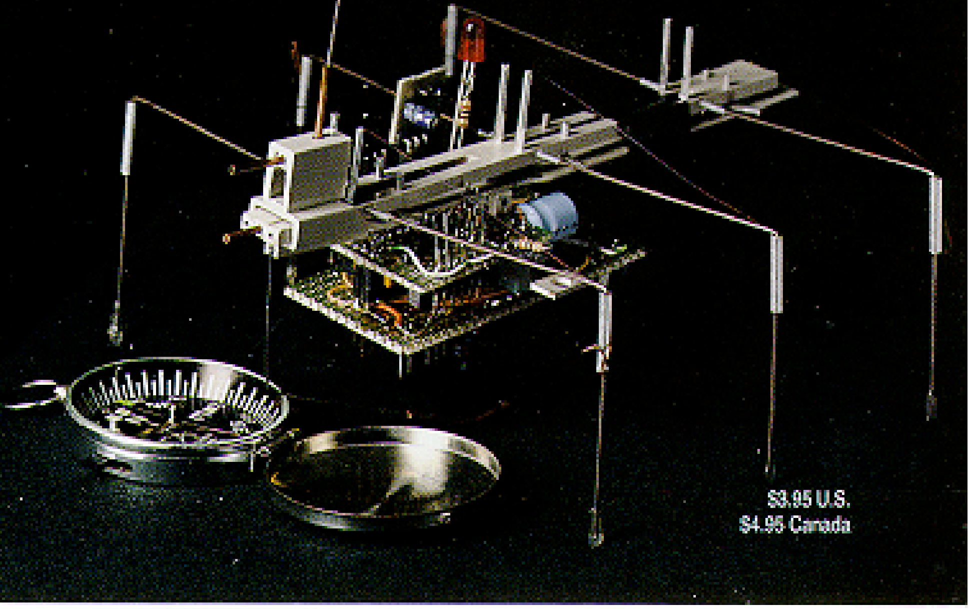
ROBOTICS

Navigation Systems for Small Robots

Electromagnetic Immunity and Susceptibility Testing

HomeR—A Vacuum-Cleaning Robot

EPC: What's CompactPCI?



$3.95 U.S.
$4.95 Canada

TASK MANAGER

For Want of a Nail

I wonder—after his defeat at Waterloo, did Napoleon look down (or up) from his eternal home and wish he'd listened to his blacksmith that morning?

You know how it is. When you're more than busy with your own job, even though you work side by side with others, you really don't know all the details of their job. Yes, if you had to fill in, you probably could do it. But then, when stuff hits the fan and you really have to do their job, you realize how many details you hadn't thought of or how many things were just in their head.

That happened to me this month. Our art director Lisa Ferry, after seven years of working at *INK*, moved on to another publishing house midmonth. This, of course, left Janice, Elizabeth, and me to somehow get this issue to press on time. Since I had the most experience with getting a document ready for the printer, I was elected graphic artist of the month.

My biggest shock was the incredible number of niggling details that go into placing advertisements. The sheer number of advertisers, each with their own individual requirements, submission methods, and—unfortunately—deadlines!, made taking over this job a challenge.

In addition, we just got a new PowerPC in graphics. Its faster speed enabled me to improve the resolution of scans. Hopefully, with this change, we'll be able to catch any errors earlier in the process.

Fortunately for me, our new graphic artist KC Zienka has arrived, bringing with her a background that includes a mix of hands-on chip production (she worked at Intel) and several years of graphics. We look forward to the changes she will bring.

Oh yeah! Content—I've been so busy with the production aspects of this issue, I almost forgot the editorial.

Ingo Cyliax starts this robotics issue off by looking at navigation schemes for robots. That's his Stiquito (*INK* 73) checking out the compass on the front cover. Next, Keith Doty introduces us to Talrik, Jr., a mobile autonomous robot that can be used as a development platform. Wrapping up the features is Part 3 of Bill Payne's primer on networking. He focuses this month on interconnecting devices.

Mike Baylis launches *Embedded PC*'s editorial by bringing us up to speed on CompactPCI. In PC/104 Quarter, Frank Jenkins, our second-place contest winner for last year's PC/104 contest, lets us in on his special project—a home vacuum-cleaning robot. Fred finishes things off with a tale about creating a BIOS in somewhat more than seven days.

In our columns, Joe DiBartolomeo explores immunity and susceptibility issues in EMI testing for both the U.S. and Europe. Jeff finishes his article on Precision Navigation's 2× compass by making a digital compass he can take on the trail. And, Tom ties up our editorial with a look at Atmel's 8-bit AVR RISC processors.

editor@circellar.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
Lisa Ferry
KC Zienka

PRODUCTION STAFF
John Gorsky
James Soussounis

CONTRIBUTOR:
Pellervo Kaskinen

ASSOCIATE PUBLISHER
Sue Hodge

CIRCULATION MANAGER
Rose Mansella

CIRCULATION CONSULTANT
John S. Treworgy

BUSINESS MANAGER
Jeannette Walters

ADVERTISING COORDINATOR
Dan Gorsky

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. **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 © 1997 Barbara Swenson
PRINTED IN THE UNITED STATES

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

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@circellar.com**. World Wide Web: **http://www.circellar.com/**

All programs and schematics in *Circuit Cellar INK*® have been carefully reviewed to ensure their performance is in accordance with the specifications described, and programs are posted on the Circuit Cellar BBS for electronic transfer by subscribers.

Circuit Cellar INK® makes no warranties and assumes no responsibility or liability of any kind for errors in these programs or schematics or for the consequences of any such errors. Furthermore, because of possible variation in the quality and condition of materials and workmanship of reader-assembled projects, *Circuit Cellar INK*® disclaims any responsiblity for the safe and proper function of reader-assembled projects based upon or from plans, descriptions, or information published in *Circuit Cellar INK*®.

12 **Robot Navigation Schemes**
by Ingo Cyliax

20 **Talrik, Jr.**
A Mobile Programmable, Autonomous Robot
by Keith Doty

28 **A Networking Primer**
Part 3: Interconnecting Devices
by Bill Payne

62 **MicroSeries**
Standards for Electromagnetic Compliance Testing
Part 3: Immunity and Susceptibility
Joe DiBartolomeo

70 **From the Bench**
You Can Take It With You
Finding Your Way, Electronically
Jeff Bachiochi

76 **Silicon Update**
Not Your AVRage MCU
Tom Cantrell

EMBEDDED PC

See Pages 35–61 for our Special Section

INSIDE ISSUE 81

2 **Task Manager**
Ken Davidson
For Want of a Nail

6 **Reader I/O**
Letters to the Editor

8 **New Product News**
edited by Harv Weiner

Priority Interrupt 96
Steve Ciarcia
It's Win-Win *and* $10,000

Advertiser Index 65

READER I/O

BLAST OFF!

Do-While Jones' series on GPS ("The Global Positioning System," *INK 77* and 78) came right on time.

I want to launch a rocket, using an M-class motor, and return it as a glider near its starting point. I'm trying several devices, including a GPS, in the rocket. I may buy a GPS unit from Navtech GPS Supply and use the "GPS 31 TracPak" (www.navtechgps.com/pcmcia.htm) and KFTool 2.51 software. Any advice?

Dan French
danf@connectsoft.com

If your rocket returns to the launch point, you don't need to convert to geodetic coordinates. But, if you launch your rocket from an arbitrary point and want it to fly to 35°N latitude, 100°W longitude, that's another matter. You can probably work in GPS's geocentric coordinates. Just remember your starting coordinates.

Your autopilot may need to know which way is up, so just rotate the vector back to the launch point into an x-y-up coordinate system (x and y don't need to align with East and North). Since it probably doesn't matter if Up is off a couple degrees, don't worry about Earth's curvature over the distance covered by your rocket. Your rotation matrix probably doesn't need to be precise.

Do- While Jones
do_while@ridgecrest.ca.us

NOT ONLY X MARKS THE SPOT

I read with great interest Do-While Jones' "The Global Positioning System, Part 1: Guiding Stars" *(INK* 77). I've worked with GPS for 11 years and am glad to see *INK* cover this technology. While most of the article was informative and correct, I found some errors.

The article states, ".receiving -65 dB signals isn't trivial.. .." That's true. However, the received GPS signals are at -160 to -166 dBW. They're below the noise floor. Check the NAVSTAR GPS Joint Program Office Web site for verification (www.laafb. af.mil/MC/CZ/homepage/segments).

I'd also like to clarify the "Warm-Up Time" section. Nothing requires warmup. During a cold start (i.e., first-time powerup with no valid information), the receiver can require up to 15 min. to provide a position solution. After this, it should maintain the last known position via a low-power time source and satellite information (called almanac and epherimedes) on back-up power.

PCs use the same technique to maintain time and CMOS set-up data. Provided there's information from the last time the receiver was on, the receiver should be able to reach a position solution in less than 2 min. During this acquisition time, the receiver doesn't have to be stationary.

GPS doesn't need to operate on a car when the ignition is off. The receiver can maintain the required information with back-up power, drawing only a milliamp or two. Since most modern cars draw 30-60 mA off the battery when the ignition is off, this isn't significant. If the receiver is running, however, the 100–300-mA current draw is very substantial compared to the drain from the rest of the car.

Steven **Quella**
saquella@msn.com

A MINISERIES, PLEASE

I'm hoping Ingo Cyliax's "Video Timecode Fundamentals" *(INK 77)* is Part 1 of many. I'd like to see more detail on implementing an entire system. I'm putting together a functioning system and need more details.

Keep up the good work. Every issue of *INK* stretches the old "squishyware."

David Bley
d.bley@genie.com

Contacting Circuit Cellar

We at Circuit *Cellar INK* encourage communication between our readers and our staff, *so we* have made every effort to make contacting us easy. We prefer electronic communications, but feel free to use any of the following:

Mail: Letters to the Editor may be sent to: Editor, Circuit Cellar INK, 4 Park St., Vernon, CT 06066.
Phone: Direct all subscription inquiries to (800) 269-6301. Contact our editorial off ices at (860) 875-2199.
Fax: All faxes may be sent to (860) 871-0411.
BBS: All of our editors and regular authors frequent the Circuit Cellar BBS and are available to answer questions. Call (860) 871-l 988 with your modem (300-l 4.4k bps, 8N1).
Internet: Letters to the editor may be sent to **editor@circellar. corn**. Send new subscription orders, renewals, and address changes to **subscribe@circellar.com**. Be sure to include your complete mailing address and return E-mail address in all correspondence. Author E-mail addresses (when available) may be found at the end of each article. For more information, send E-mail to **info@circellar.com**.
WWW: Point your browser to **www.circellar.com**.

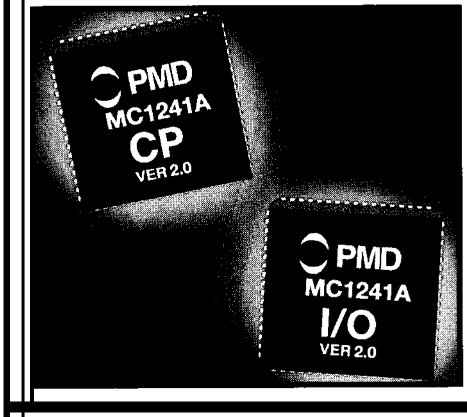
NEW PRODUCT NEWS

Edited by Harv Weiner

MOTION-CONTROL CHIPSET

The **MC1241A** is a dedicated motion processor that functions as a complete chip-based stepper-motor controller. Its drive method lets each phase of a stepper motor be individually controlled with a microstepping waveform. This configuration is ideal for stepper-based applications that require smooth, high-accuracy motion. The MC1241A is available in one- or two-axis configurations.

Packaged in a two-IC chipset, this device performs trajectory generation and microstepping signal generation. The chipset outputs PWM- or DAC-compatible motor command signals that directly drive the stepper motor's windings, eliminating the need for external microstepping circuitry. The microstepping waveforms generated by the chipset provide 64 microsteps per full step. User-programmable registers enable the amplitude and frequency of these waveforms to be precisely controlled. Both two- and three-phase stepper motors are supported.



The MC1241A also provides inputs for quadrature encoder feedback. Each axis maintains the encoder position to a 32-bit resolution. A special feature of the chipset is that if an encoder is connected to the motor, it automatically detects a motor-stall condition during motion. If an encoder is not connected to the motor, it drives the motor using a traditional open-loop approach.

Other standard features include four user-selectable profiling modes, as well as S-curve, trapezoidal, velocity contouring, and electronic gearing. All profile control registers are 32 bits. Additional control modes are provided for automatic position breakpoints, host interrupt generation, and multiaxis synchronization.

The two-axis chipset sells for $65 in quantity.

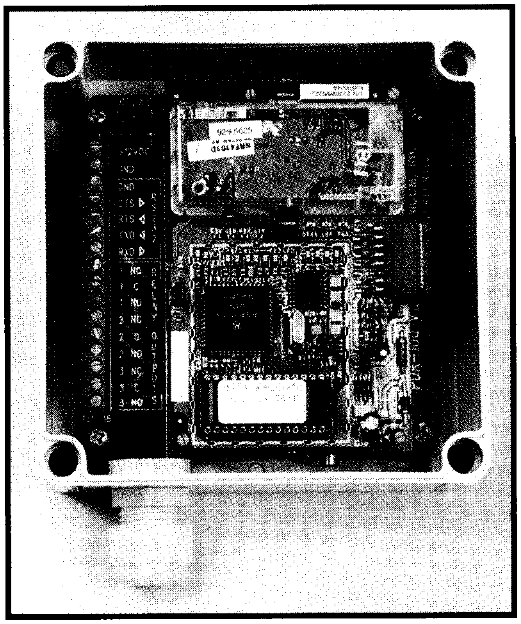
Performance Motion Devices, Inc.
97 Lowell Rd.
Concord, MA 01742
(508) 369-3302
Fax: (508) 369-3819

#500

PAGING DATA RECEIVER

The **PDR-100** enables industry-standard paging transmissions to operate remote relays and deliver ASCII RS-232 messages to a remote site. Applications include electric utility load control, customer notification, capacitor bank switching, traffic control, remote HVAC control, lighting, and signs.

The unit can receive numeric and alphanumeric pages. The string of digits received in a numeric page is decoded and interpreted as commands to operate three onboard relays. The ASCII-character string received in an alphanumeric page is output through an RS-232 serial port, which can drive a printer or other RS-232 device.



The PDR-100 uses Motorola's Bravo Plus receiver technology to receive and decode broadcasts over an existing paging infrastructure. It is available for VHF (138-174 MHZ), UHF (406–5 12 MHz), and 900-MHz (929-932 MHz) frequencies. The PDR-100 operates with existing paging systems or with one of the many third-party paging-service suppliers. The unit's design enables a single paging account to address units for relay control either individually or as a group. The paging interface is industry-standard POCSAG 512, 1200, or 2400 (numeric and alphanumeric with 7-bit ASCII).

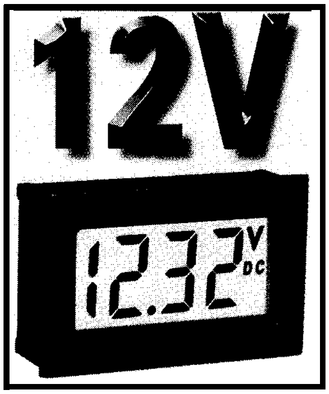
The PDR-100 is housed in a 5.12" x 5.12" x 3" weatherproof enclosure with a tamper-seal cover. Its relays have a contact rating of 8 A at 115VAC.

Technicom, Inc.
20 Washington Ln. SE
Concord, NC 28025
(704) 788-8944
Fax: (704) 782-l 122

#501

NEW PRODUCT NEWS

DC VOLTAGE MONITOR



Datel's **DMS-20LCD-DCM** is a low-cost, self-contained, self-powered, DC voltage monitor. Two models are available-one for 12-VDC nominal operation (+6.5-18-V range, 0.01-V resolution), and the other for 24-VDC (+17–40-V range, 0.1 -V resolution). Applications include DC bus-voltage monitoring, automotive batteries, battery chargers, and solar generators.

The monitor uses a precision ADC and ultra-stable passive components to gain excellent performance over the 0 to +60°C operating temperature range. Applying power to the two rear terminals is all that's required for operation, and the unit draws only 2 mA from the monitored source. Reverse polarity protection is standard on both models. The 0.37" high LCD display with built-in VDC annunciator can easily be read from 12′. The entire unit, including its display and SMT electronics, is housed in an ABS plastic package measuring 1.38" x 0.88" x 0.66".

Pricing starts at $25 in single quantities.

Datel, Inc.
11 Cabot Blvd.
Mansfield, MA 02048
(508) 339-3000
Fax: (508) 339-6356

#502

64-CHANNEL A/D SYSTEM

The **DSPA64/DSPHLF** system from Symmetric Research is a complete, PC-based system for A/D acquisition and processing. The DSPA64 card is external and features a high-resolution 16-bit ADC and a 64-channel programmable multiplexer array. Sampling at aggregate rates up to 138 kHz, all inputs are differential and buffered through a low-noise precision instrumentation amplifier. In addition, overall gain and amplitude limits can be user set by resistors, with nominal values of 1 .O and f2.75 V. A 16-conductor ribbon cable transfers the data in serial form to the DSPHLF card inside the PC.

The DSPHLF card, which features a 50-MHz AT&T DSP32C floating-point DSP, processes and buffers up to 1 MB of incoming data without using any PC time. The DSP32C double buffering enables large blocks of data to continuously be saved to the hard disk with no data loss. Because the PC is completing only disk saves, PC time is available for displaying graphics or running a windowed environment.

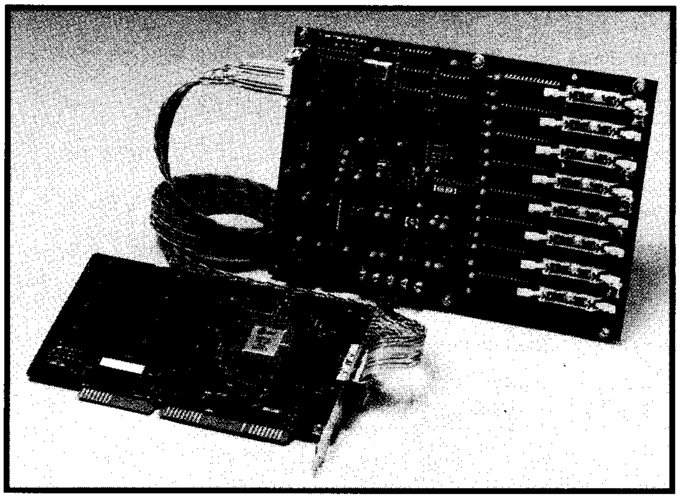
Incoming data is processed up front by the DSP32C, so you can apply digital filters to the incoming data in real time. Data can be oversampled then smoothed to increase the system's overall effective resolution. This powerful feature is further enhanced because DSPHLF programs are saved in onboard SRAM and can be changed anytime by the user. This capability enables you to modify filtering coefficients and other parameters as necessary in real time.

The system offers a complete development environment which provides all the tools necessary to develop custom code and applications, including an assembler and monitor debugger. A full-featured data-acquisition kernel and display program lets users specify selected channels, control acquisition rates, continuously save data, and optionally run a real-time display. Full source code for the acquisition program, as well as many introductory example programs, is included. Full circuit diagrams offer hardware specifics.

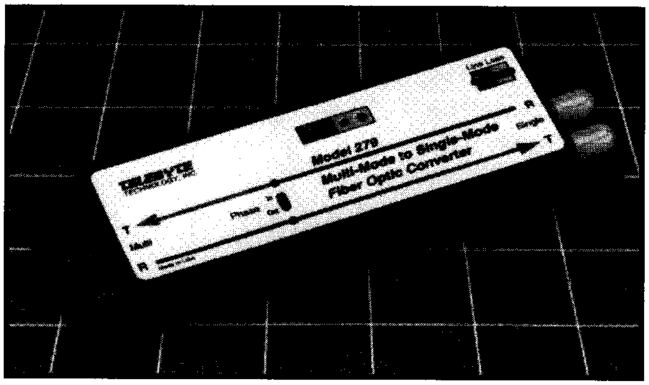
The complete package, including software and power supply, is priced at $2500.

Symmetric Research
15 Central Way, Ste. 9
Kirkland, WA 98033
(206) 828-6560
Fax: (206) 827-3721

#503



NEW PRODUCT NEWS



FIBER-OPTIC CONVERTER

The **Model 279** single-mode-to-multimode fiber-optic converter provides long-distance transmission of fiber-optic signals. A unique feature enables its use in systems that have nulls in their frequency response, as is often the case in master/slave polling networks. Typically, multimode transmission services distances of -2 km. For extended distances (i.e., up to 20 km), single-mode fiber is used. For applications where multimode equipment is used but only single-mode fiber exists, a pair of Model 279s performs the media translation from multimode to single-mode, enabling the use of single-mode fiber.

The Model 279 offers full-duplex conversion for 850-nm multimode signals to 1300-nm single-mode signals. All fiber ports are implemented with ST connectors. A line-loss switch compensates for single-mode cable loss, allowing the unit to accommodate single-mode cable losses of 2, 5, 10, and 15 dB. These losses approximate cable lengths of 3, 7.5, 15, and 20 km, respectively. The selected line loss is displayed on one of four LEDs.

When used as a media converter, the phase of the fiber-optic signals may not be the same. The Model 279 includes a phase-reversing switch for situations where the signal phases of the single-mode and multimode are different. In addition, transmit and receive LEDs display the activity of data passing through the unit.

The Model 279 measures 7” x 3″ x 1” and can mount in any position. Power is supplied by a wall-mounted adapter. The Model 279 sells for $775.

**Telebyte Technology, Inc.
270 Pulaski Rd.
Greenlawn, NY 11740-l 616
(516) 423-3232
Fax: (516) 385-8184
telebyteusa.com**

#504

DCMOTORCONTROLLER

The UC3645 and UC3646 are bipolar integrated circuits designed to drive full-wave brushless DC motors without position sensors. Bidirectional control increases design flexibility, while sensorless commutation reduces component counts and cost. The devices are ideal for micro motor control under 24 W (e.g., copiers, laser printers, fax machines, hard disks, and tape drives).

The UC3645 and UC3646 are designed for use in full-wave drive of three-phase motors. Two 1.8-A drivers are active at one time in each of the six output states. As one sources current, the other sinks it. The internal logic determines the ideal time to commutate the motor based on the EMF signal evaluation of all outputs generating a motor position signal. The same signal also provides speed information via the FG output pin (tach output].

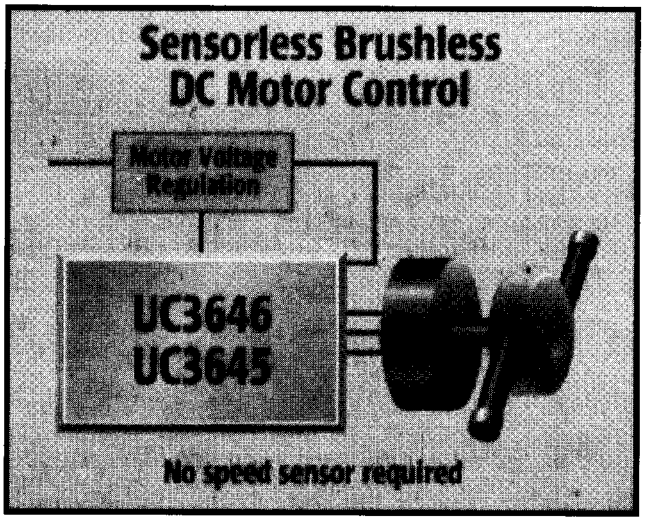
Because the motor load is highly inductive, the outputs incorporate flyback diodes. Current limiting and thermal protection are provided, as well as soft start and programmable commutation delay. Internal start-up and timing oscillators create a minimum commutation frequency and detect reverse rotation since EMF signal is not available during startup.

The UC3645 has a transconductance amplifier for driving a linear regulation transistor for low-noise motor voltage control. The UC3646 uses a PWM comparator to drive a switching regulator transistor for highly efficient motor voltage control.

The UC3645 and UC3646 are priced at $4.94 in 1000-piece quantities.

**Unitrode Corp.
7 Continental Blvd.
Merrimack, NH 03054-4334
(603) 424-2410
Fax: (603) 424-3460**

#505



FEATURES

12 Robot Navigation Schemes

20 Talrik, Jr.

28 A Networking Primer

Robot Navigation Schemes

FEATURE ARTICLE

Ingo Cyliax

If you're an autonomous mobile robot, ya gotta know where you're gonna go. Ingo uses the dead-reckoning method for his robots via a Basic Stamp 2, a Vector 2× compass, and an LCD.

Oor mobile robots to be useful, they have to know where they are.

In this article, I describe several useful techniques for navigating mobile robots, including dead-reckoning and beacon-based systems. The challenge in robotics, as with all engineering, is to arrive at an optimal cost-effective solution that does the job. But that's difficult, especially with navigation.

At first, the solution might seem to be slapping a GPS receiver on the robot. However, if you're dealing with small inexpensive robots like Stiquitos or Servobots (see "Modular Robot Controllers" in *INK* 73), it's not quite that easy.

To illustrate my ideas, I show you a small navigation system based on components available from mail-order sources. The system uses a neat way to compute accurate trig functions, which is described in detail in the sidebar about CORDIC. You can integrate this system into your next mobile-robot project.

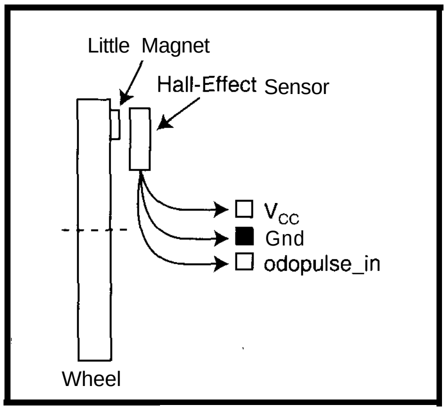


Figure l-/n this *simple odometer for a wheeled robot, the magnet and sensor come* from a 3.5" *floppy* disk *drive.*

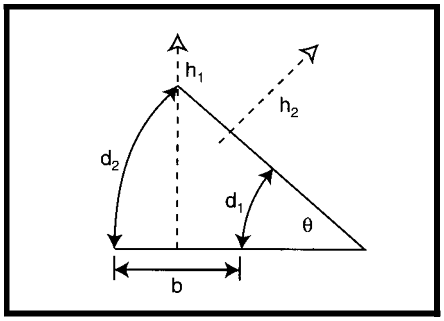


Figure 2-For this differential drive heading sensor, both wheels *have an odometer with enough resolution to send fhe difference in distance traveled* ($d_1 - d_2$) *by each wheel in a curve.*

DEAD-RECKONING

Dead-reckoning is probably the oldest navigation system there is. In dead-reckoning, you track your current position by noting how far you've traveled on a specific heading.

For centuries, sailors used a magnetic compass to note their heading and a combination of sand / water clocks and knotted ropes to measure time and speed. Of course, with ocean currents, this method was never that accurate.

Ships often missed their destination or never made it home. Perhaps that's why it's called dead-reckoning. With the invention of the sextant and chronograph, things got much better-but more about that later.

In robotics, we can also use dead-reckoning. Measuring distance is easy with wheeled robots and some kind of odometer. By putting an encoder on a wheel and measuring the number of revolutions, it's simple to calculate the distance traveled if you know how big the wheel is:

$$\text{distance} = 2\pi r \times \text{revolutions}$$

Figure 1 shows a simple odometer that provides a pulse each time the wheel completes a rotation. The Hall-effect sensor comes from a floppy disk drive.

By mounting more magnets, I can increase resolution by increasing the number of pulses per revolution. For example, eight magnets let me measure distance down to one-eighth of the circumference, simply by counting pulses. The more pulses, the better the resolution.

With a position encoder, I can measure the absolute position of the wheel and, more importantly, the direction of travel. If I use stepper motors or legs, measuring the distance is a breeze. I just count the steps and multiply by the distance traveled with each step.

Now, how do I find the current heading? Let's look at some techniques that can be easily implemented on a robot.

One way to find the heading is to use one odometer per wheel, assuming there are two or more. Remember to note the difference in distance traveled by each wheel since, when the robot turns, one wheel travels a farther than the other.

Look at Figure 2 to see this effect. To calculate a change in heading from this, use simple trigonometry:

$$\theta = \frac{d_1 - d_2}{b}$$

where θ is the angle in radians, *b* is the distance between the wheels, and *d*, and d_2 are the distances traveled by the right and left wheels, respectively.

To track the absolute heading, accumulate the change in each turn:

$$\text{heading}' = \text{heading} + \theta$$

Note, that θ goes negative when the left wheel travels a greater distance than the right wheel. Also, the absolute heading is positive in a counterclockwise direction. This contrasts with a compass heading, which is positive in the clockwise direction (i.e., north = 0", east = 90°, south = 180°, and west = 270"). Therefore, the new absolute heading after the turn in Figure 2 is smaller than it was before.

I now have a vector which is the average distance of half of $(d, + d_2)$ at a certain angle. To track the position, I accumulate the x and y components of this vector:

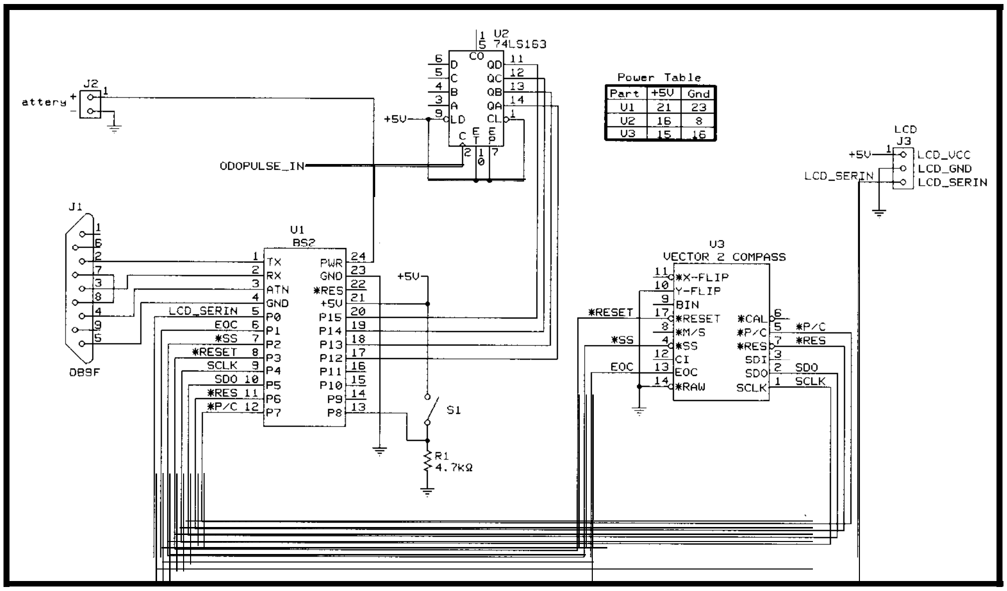
$$x' = x + \cos(\text{heading}) \times \frac{d_1 + d_2}{2}$$

$$y' = y + \sin(\text{heading}) \times \frac{d_1 + d_2}{2}$$

Besides the differential wheel-heading indicator, an electronic compass directly measures absolute heading. The earth's magnetic field is about 0.5 G and can be visualized by imagining a huge magnetic dipole with poles roughly at the north and south poles.

They aren't really at the poles, and the deviation is called the declination. This fact is important when using compasses for global navigation, but not for local navigation.

Three kinds of electronic compasses are available-actually, one isn't electronic in the strictest sense. An electronic compass uses a magnetometer to measure the earth's field, while another compass commonly referred to



Part	+5V	Gnd
U1	21	23
U2	16	8
U3	15	16

Figure 3—*This dead-reckoning navigation system uses a Parallax Basic Stamp 2 and a PNI electronic compass module. The connection to the serial LCD module is via three wires—lcd_vcc, lcd_gnd, and lcd_serin.*

CORDIC-The Swiss Army Knife for Computing Math Functions

CORDIC [Coordinate Rotation Digital Computer) is a method for computing elementary functions using minimal hardware (e.g., shift and add). It's typically used when functions need to be implemented directly in hard. ware.

Initially, CORDIC was hardware developed for real-time high-precision navigational computations in the 1950s. Since then, this technique has been integrated into almost all scientific calculators.

CORDIC works by rotating the coordinate system through constant angles until the angle reduces to zero. The angle offsets are selected so that the operations on x and y are only shifts and adds. Let's first look at the math and then an example.

1'11 start with some coordinates (x,y) which I want to rotate by angle a. The new coordinates (x',y') are defined by:

$$x' = x\cos(a) - y\sin(a)$$
$$y' = y\cos(a) + x\sin(a)$$

I rewrite these to get a tangent of the angle:

$$\frac{x'}{\cos(a)} = x - y \times \tan(a)$$
$$\frac{y'}{\cos(a)} = y + x \times \tan(a)$$

If I break the angle into smaller and smaller pieces so the tangents of these pieces are always powers of 2 and they still add up to the total angle, I can write:

$$x(i+1) = K(i)\ x\ (x(i) - {}^{y}\!/_{2^i})$$
$$y(i+1) = K(i)\ x\ (y(i) + {}^{x}\!/_{2^i})$$

where the angle for each step is:

$$A(i) = \operatorname{atan}\left(\frac{1}{2^i}\right)$$

I won't worry about the ***K(i)*** for now because there's an easy way to deal with it. With these iterative equations, I can design an algorithm that, given an angle in a, will reduce this angle to zero.

At each step, it also increments or decrements the x and y coordinate register by the appropriate value (i.e., shifted values of x and y), thus keeping track of the coordinates while rotating:

```
for i = 0 to N
    dx = X / 2^i
    dy = Y / 2^i
    da = atan(1/2^i)
    if Z >= 0 then
        X=X  - dy
        A = A - da
        Y = Y + dx
    else
        X = X + dy
        A = A + da
        Y=Y  - dx
    endif
next
```

In a real program, I'd precalculate the atan(I/2') values and store them in a lookup table. The divide by 2' should end up as a simple shift instruction on most architectures.

So, how do I calculate the sine and cosine with this? I use this algorithm with initial values and let it calculate the answer. For example, I initialize the angle as 30" and iterate by 8 steps (equivalent to 8 bits of precision).

Remember the ***K(i)*** constants and how they were missing in the last algorithm? By multiplying all the ***K(i)s*** together, I get what's called the aggregate constant:

$$K = \cos\left(\operatorname{atan}\left[\frac{1}{2^0}\right]\right) \times \cos\left(\operatorname{atan}\left[\frac{1}{2^1}\right]\right) \times \ldots \times \cos\left(\operatorname{atan}\left[\frac{1}{2^n}\right]\right)$$

which turns out to be 0.607. It's the same constant regardless of the precision. You can just truncate it to the number of bits you need. I use it to initialize the x register and turn the crank, as you can see in Table i.

i	x	*y*	*a*	atan(1/2^i)
0	0.607	0.000	30.000	45.000
1	0.607	0.607	-15.000	26.565
2				
3	0.835 0.910	0.303 0.531	-2.471 11.565	14.036 7.125
4	0.901	0.427	4.654	3.576
5	0.874	0.483	1.077	1.790
6	0.859	0.510	-0.712	0.895
7	0.867	0.497	0.183	0.448
8	0.863	0.504	–0.265	0.224

Table i—*Calculating the sine and cosine of 30° to* **8 bits** *of accuracy using CORDIC fakes eight steps. The result is in the x(cos) and y(sin) columns in step 8.*

The answers appear in register x (cos) and y (sin). My trusty old HP41C got sin(30) = 0.500 and cos(30) = 0.866. Of course, $1/2^8$ is 1/256, which ends up being 0.0039. So, I can't really hope for more accuracy with 8-bit precision.

CORDIC can do much more. You can also calculate the arctan by initializing the x and y registers, setting a to zero and driving y to zero, with the results:

$$x = \sqrt{x^2 + y^2}$$
$$y = 0$$
$$a = \operatorname{atan}\left(\frac{y}{x}\right)$$

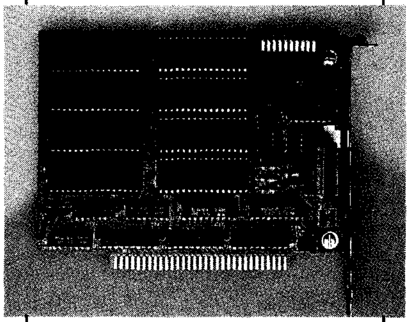
Having the vector magnitude of x and y in x can be handy, don't you think?

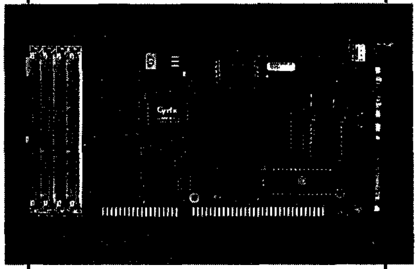
Some clever people also figured out a way to use CORDIC to calculate other functions such as exponential functions (using a table of $\ln(1+2^i)$) and hyperbolic trig functions (using atanh[$1/2^i$] tables).

Check out the references for this article for applications and refinements of this technique.

If you need high-precision trig functions with a small look-up table (n entries) and good performance, give CORDIC a try. I can even implement a 12-bit CORDIC routine on a Parallax BASIC Stamp 2.

You can get the C code used to calculate Table i and an example of the Stamp II CORDIC code at <ftp.cs.indiana.edu/pub/goo/RobNav/scordic.c> and <ftp.cs.indiana.edu/pub/goo/RobNav/cordic1.bs2>.





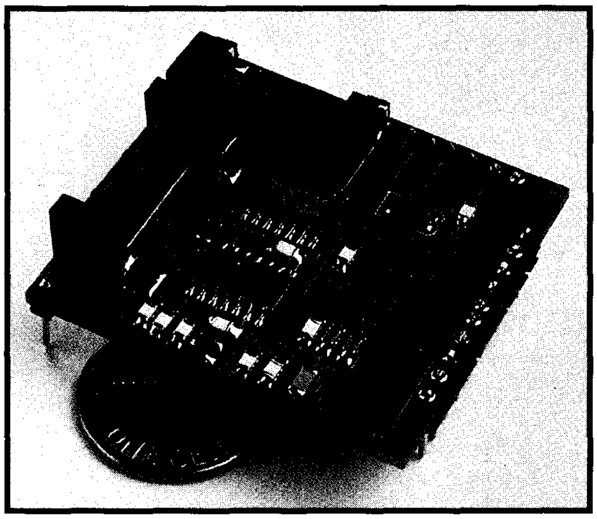


Photo 1 —*This Precision Navigation V2x compass module measures 1.50″ × 1.30″. You can also see the two sensor coils.*

as electronic is just a magnetic compass with a position encoder detecting which direction the needle points. Such compasses are a bit bulky for robotics, but they're simple.

The most common electronic compass is the flux-gate compass. It uses a flux-gate-based magnetometer to measure the magnetic flux of the earth directly. Two sensors measure the x and y components of the field at the current heading.

To calculate the heading, take the arctangent of the ratio:

$$\text{heading} = \operatorname{atan}\left(\frac{y}{x}\right)$$

which is really neat, since the actual magnitudes of the fields cancel out in the fraction.

A Hall-effect compass uses a Hall-effect transistor instead of flux-gate sensors, and it's simpler, smaller, and more robust than a flux-gate compass. It's completely solid state and doesn't require coils. Like all good things in life, Hall-effect-based compasses are more expensive.

The earth's magnetic field isn't parallel to the earth's surface except at the equator. For a compass to work properly, it must be aligned parallel to the earth's surface to measure the x and y components of the field accurately.

Alignment is achieved by using a gimbal mount or by putting the compass in a bubble of oil so gravity levels the compass and the oil dampens vibrations. Some compasses have a third z-axis sensor to measure the field when mounted in any position.

Another complication are magnetic fields introduced by EM emissions. Since these emissions are normally generated by AC, they can be filtered out electronically or by the viscous dampening used in physical compasses. This explains why compasses usually have a low sampling rate (i.e., **5-10 Hz).**

If the vehicle with the compass includes any ferroelectric materials that alter magnetic flux lines, you need to compensate by doing a hard-iron calibration. Calibration is also necessary on robots that may include the static magnetic fields typically found in permanent magnet motors.

Environmental (nonvehicular) magnetic fields may also need to be compensated for. Since they are static, a map-based look-up table can be used.

Inertial navigation systems, which are based on gyroscopes, don't suit robotics applications. Small units tend to drift too much, and low-drift units are too expensive and bulky.

Rate gyros augment other navigational systems to provide some rotational stability with a faster response time than a compass. Small and light rate gyros compensate for the torque present in radio-controlled model helicopters.

DEAD-RECKONING PROJECT

You can build a simple dead-reckoning system with readily available components. My system consists of a Parallax Basic Stamp 2 and a Vector 2x electronic compass made by Precision Navigation Inc. (shown in Photo **1).** Both the Stamp and the compass are also available from Jameco and JDR. Figure 3 shows the setup.

Odometer pulses are buffered using a counter ('163) to ensure that nothing is missed. Interfacing the compass module is straightforward by using a

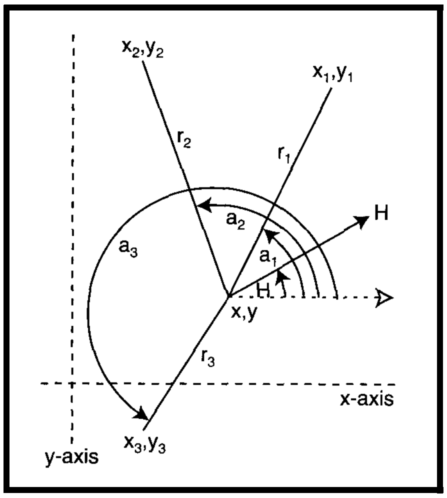


Figure 4—*This* **vector** *diagram shows* **the** *relationships between the vehicle at an unknown location* **(x,y)** *with a* **heading of(H)** *and three known beacons at* (x_1,y_1), (x_2,y_2), *and* (x_3,y_3). **The** *angles* a_1, a_2, *and* a_3 *would be measured in a* **triangulation system.** *The distances* r_1, r_2, *and* r_3 *would be known in a trilateration system.*

synchronous serial protocol with SCLK, SDO, and *SS.

When I want to take a reading, I pulse the *P/C (poll) line and wait until the conversion is done, which is signaled by the EOC line. The data is now valid and can be read serially.

Finally, a serial LCD module records the position and current heading. A single serial line (lcd_serin) interfaces with the LCD module.

On the software side, things aren't quite so simple, but they're still manageable. The main loop continually polls the odometer by reading the high nibble of the 16-bit input register (IND) and the state of the compass's EOC line (IN1). It then dispatches to the appropriate routine for processing:

```
main:
  if IN1 = 1 then proc_compass
  if IND <> last_odo then
    proc_odo
  gosub display
  got0 main
```

To process the current odometer reading, I use a CORDIC-based routine to compute the sin and cos components of the current vector (heading and distance) and add them to the current position.

The CORDIC routine scales the distance by the aggregate constant in register x and the angle (i.e., the current heading) in *z*. The scaled vector components will be in *x*(cos) and *y*(sin) and are added to the current position:

```
proc_odo:
  X = K * ((IND - last_odo) &
    $f)
  Y = 0
  Z = heading
  gosub circ
  pos_x = pos_x + x
  pos_y = pos_y + Y
  got0 main
```

To process the compass, read the x and y field measurements and calculate the heading by taking the arctangent and storing the result in the variable heading, as you see in Listing 1. The subroutine display formats and displays the current values for heading, pos_x, and pos_y on the LCD.

The only other thing to note about dead-reckoning is that systematic errors (i.e., resolution in the encoder and uncertainty of the exact heading) accumulate. To overcome this, a typical dead-reckoning system needs calibrating occasionally by aligning the current position with an actual position.

BEACON-BASED NAVIGATION

Beacons are locations with known coordinates which emit signals (e.g., radio waves or light) to be received by the vehicle trying to find its location. Measuring the distance to these beacons is called trilateration. Other types of systems measure the angle to the beacon, in which case it's called triangulation.

Generally, we end up solving for x and y (and maybe *H*, the heading) in the following system of equations, which go with Figure 4:

$$x + r_1 \cos(H + a_1) - x_1 = 0$$
$$y + r_1 \sin(H + a,) - y_1 = 0$$
$$x + r_2 \cos(H + a,) - x_2 = 0$$
$$y + r_2 \sin(H + a_2) - y_2 = 0$$
$$x + r_3 \cos(H + a_3) - x_3 = 0$$
$$y + r_3 \sin(H + a_3) - y_3 = 0$$

where x_1,y_1 through x_3,y_3 are known locations of beacons, a_1–a_3 are relative angle to beacons, r_1–r_3 are the distance to beacons, *His* the heading, and *x,y* is my location.

Today, the most common navigational systems-GPS, Loran, Omega, and VOR-use trilateration by measuring the time of flight (TOF) or phase relationships of radio signals from the beacon to the vehicle. Even though these systems really use TOF and carrier-phase relationships and are trilateration systems, they often provide heading information to a user.

While these systems work well for finding airports and harbors, it's fairly hard to measure distances with fine resolution. Radio waves propagate in the order of 1' per nanosecond, which calls for very accurate clocks.

Of course, differential GPS (DGPS) can accurately obtain position even with selective availability turned on, but the equipment is too expensive for low-end robotics. It also requires a second stationary GPS receiver to communicate with the robot.

Most of the low-cost trilateration systems for robotics rely on ultrasonic beacons. Sound waves travel at a rate of -900 per second, giving plenty of time for TOF measurements. Laser range-finding can measure the distance to laser targets.

Examples of real-life triangulation systems include lighthouses and celes-

Listing 1--Reading *the x and* **y magnitudes from** *the compass module is done* *with the Stamp 2's* *shiftin* **instruction.** *The heading is* **calculated by** *taking* **the arctangent** *using* **the** *CORDIC technique.*

```
proc_compass:
  pause 10
  low SS_                      ' assert select
  pause 10
  shiftin SDO,SCLK,0,[X\16]    ' read sensor values
  shiftin SDO,SCLK,0,[Y\16]
  high SCLK
  high SS_
  pulsout P_,5000
  Z = 0
  gosub inv_circ               ' calculate arctan
  heading = Z                  ' the answer
  got0 main
```

tial navigation, which is still used as a back-up system to man-made beacon-based systems. For celestial navigation, a sextant measures the inclination of celestial objects above the horizon as well as a chronograph (a very accurate clock) for transit measurements.

Luckily, most triangulation systems available to the robot designer are based on lasers which scan for targets (e.g., ID tags or retroreflectors). Some hybrid systems give range information in addition to angles.

SENSOR FUSION

You now know about several navigation schemes. Most have strengths and weaknesses. In navigation, we're mostly worried about accuracy and having backups.

Imagine a ship that relies on GPS, but the GPS receiver dies. After all, it's a fairly complex system and relies on data (the ephemeris) downloaded from the satellite.

A backup for this might be the Omega navigation system, which is a VLF (very low frequency] radio navigation system using phase-carrier comparisons to give vectors. This system isn't available everywhere in the world, and even where it is available, it has variable accuracy depending on your location and how many transmitters you can receive.

You could also fall back on dead-reckoning. Ships keep logs about how fast and long they've traveled on a particular heading. A sextant and chronograph can be used to get a fix on the current location to update the estimate that dead-reckoning gives.

In nautical navigation, several systems are used. Each system has an accuracy that enables the navigator to assign a certainty of how reliable the information is. Systems with higher accuracy are used when available.

In robotics, we do this with sensor fusion. We can estimate our current location by assigning weights to each measurement which implies a certain "correctness" value. We then average all the measurements and arrive at a best guess and a measurement on how reliable this guess might be.

If I used GPS, for example, I'd assign a correctness factor based on how many GPS satellites are in view, how up-to-date the element data is, and perhaps what the distance is to each satellite.

While GPS is operating, it has a much higher weight than an odometer, which suffers from systematic errors, and an electronic compass, which has low accuracy compared to GPS. However, when no GPS satellites are available, the weight for the dead-reckoning sensors is higher and thus more correct.

WHERE TO GO FROM HERE

While there's a lot of information on this topic, it merely indicates that this problem is not so easy to solve.

At the University of Indiana, we haven't found a cost-effective solution for doing navigation in our colony robots besides using a video camera to find blinking LEDs. However, the small size of the compass used in my project looks attractive for our bigger walking robots, especially since we can measure distance fairly accurately.

A good reference on robot navigation systems is Johann Borenstein's report put out by the University of Michigan. It describes and compares almost all commercially available and research-based navigation systems and schemes that can be used by robots. ❑

Ingo ***Cyliax works as 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*** *soft-***ware and hardware development with Derivation Systems, a San Diego based formal synthesis company. You may reach** *Ingo* ***at*** *cyliax@EZComm.com.*

SOFTWARE

All software mentioned in this article is available at <ftp.cs.indiana.edu/pub/goo/RoNav>.

REFERENCES

Texts:

J. Borenstein, *Navigating Mobile Robots: Sensors and Techniques,* A. K. Peters, Ltd., Wellesley, MA, 1996, and <ftp.eecs.umich.edu/people/johannb/readme.txt>.

C. McComb, ***Robot Builder's Bonanza,*** TAB Books, Blue Ridge Summit, PA, 1987.

J.E. Volder, "The CORDIC Trigonometric Computing Technique," ***IRE Trans. Electronic Computing, 8, 330334, 1959.***

Web sites:

www.taygeta.com/cordic_refs.html

www.cs.indiana.edu/robotics/world.html

www.utexas.edu/depts/grg/gcraft/notes/gps/gps.html

SOURCES

V2× compass module
Precision Navigation, Inc.
1235 Pear Ave., Ste. 111
Mountain View, CA 94043
(415) 962-8777
Fax: (415) 962-8776
www.pcweb.com/pni

Basic Stamp 2
Parallax, Inc.
3805 Atherton Rd., Ste. 102
Rocklin CA 95765
(916) 624-8333
Fax: (916) 624-8003
info@parallaxinc.com
www.parallaxinc.com

Serial LCD Module
Scott Edwards Electronics
P.O. Box 160
Sierra Vista, AZ 85636
(520) 459-4802
Fax: (520) 459-0623
72037.2612@compuserve.com

Compass, Basic Stamp, LCD module
JDR Microdevices
1850 S. 10th St.
San Jose, CA 95 112
(408) 494-1400
Fax: (408) 494-1420
www.jdr.com

Jameco
1355 Shoreway Rd.
Belmont, CA 94002
(415) 592-8097
Fax: (415) 592-2503
info@jameco.com
www.jameco.com

IRS

401 Very Useful
402 Moderately Useful
403 Not Useful

Talrik, Jr.

A Mobile Programmable, Autonomous Robot

If you're in a catch-22 between designing a robot and having a test platform, then meet TJ—a small robot running on a compact single-chip computer. It provides an affordable, autonomous testing option.

FEATURE ARTICLE

Keith Doty

For nearly a decade, researchers at engineering and computer-science schools worldwide have taken a realistic approach to the design and development of autonomous mobile robots. The increasing expectation and demand for robotic systems to autonomously perform complex tasks in manufacturing, construction, transportation, and consumer services are driving this research.

Applications for autonomous mobile robots include diverse products like lawn mowers, vacuum cleaners, industrial and nuclear cleanup, military warriors, scouts, and saboteurs, as well as construction, underwater search, and transportation vehicles.

Some companies offer small mobile robots priced from $1000 to more than $20,000. A few start-up firms offer robots for $500 to $1000, with performance characteristics competitive with some of the more expensive models. The lower-cost models carry much smaller payloads, but their size is 100–1000 times smaller as well.

The complexity and high cost of current robot platforms prevent many from exploring and applying machine intelligence, neuronets, reinforced learning, and fuzzy logic to robots. Advances will be rapid, however, when the industry devises an inexpensive but sufficiently complex robot that supports behavior programming, learning, and manipulation capability.

START-UP COSTS

Even a minimum functioning autonomous mobile robot requires multiple sensors of various types, one or more microcontrollers, a power source, software support, and application programs. Until recently, the cost for all these features exceeded the robot's utility, except in research applications.

To develop applications, a computer-controlled robot needs software support and a programming language. Freeware compilers, simulators, and assemblers for low-cost microcontrollers reduce entry prices considerably. However, for long-term reliable support, the robot developer must eventually turn to commercial software.

In this article, I introduce you to a low-cost ($100–150), open-architecture, autonomous mobile robot. It is my hope that widespread use of such robots will generate small industries in sensors and application packages that will operate on a real, simple, easy to build and use, autonomous-mobile-robot platform.

MEET TALRIK, JR.

The design of Talrik, Jr. (alias "TJ") derives much from its parent-Talrik 1, a much larger robot with more capability and higher cost.

As you can see in Photo 1, TJ stands about 4" high, with an upper circular plate 6.5" in diameter. The platform consists of 5-ply, aircraft birch plywood, but other light, strong materials are easily substituted.

A minimal TJ sensor suite consists of two IR emitters with two IR detectors and three front bumper switches. Two 2.75" rubber-tire model-airplane wheels and a rear nylon skid provide support.

The wheels mount directly on standard model-airplane servos' output shafts. The axis aligns with the upper plate's diameter so TJ can turn in place by differential control of the motors.

A built-in recharge circuit and power plug offers a 6-h trickle charge by an AC adapter of 12 VDC and 200 mA. TJ can recharge during long hours of testing and debugging, keeping batteries fresh and ready to go for floor testing. Of course, TJ's wheels must be elevated above the bench top during this time!

A Motorola MC68HC11E2 processor on a 2" x 2" PCB controls TJ's DC motors and sensors and executes behavior programs. The 'E2 provides 256 bytes of RAM and 2 KB of EEPROM, which is sufficient space to implement a variety of sophisticated behaviors.

Motor-control software generates pulse width modulation for the two DC motors on PB7 and PB6 of port B (see Figure 1). The DC motors are standard 42 oz.-in. servos modified to always feedback the servo center position to the servo control circuitry.

Any pulse-width command between 1-2 ms on the port-B output pins indicates a set point that the servo can't achieve unless it's the center position. This position corresponds to a control pulse width setting of -1.5 ms. A set point greater than 1.5 ms causes the motors to turn forward. A setting below that causes the motor to reverse.

The PWM period varies **18-20 ms.** Differential control of the motors provides complete maneuverability. TJ can turn 180" in place.

The MC68HC11E2 provides eight channels of 8-bit ADC [port E) for sensory inputs. Port B furnishes eight digital outputs, and port C can be programmed for either inputs or outputs.

TJ has two forward- and one backward-looking IR emitter to illuminate the scene with 40-kHz modulated, 940-nm IR. PBO drives all three emitters in 2.5ms bursts (see the series LED circuit in Figure 1).

Two forward-looking, 40-kHz Sharp GP1U58Y digital IR detectors complete the IR proximity-detection system. This system handles obstacle avoidance, wall following, and beacon detection [1].

The two front IR-detector outputs feed into analog inputs PE6 and PE7. An optional rear IR detector driving analog input PE2 enables TJ to detect IR from other TJs (or predators!).

Adding a bump sensor on the upper plate provides collision detection.

SINGLE-CHIP COMPUTER CIRCUIT

When a project doesn't demand extensive computer capability or memory, a small, compact microcontroller system often proves useful. The Mekatronix MSCCl **1** single-chip computer, which incorporates an MC68HC11E2 as the onboard processor, is ideal for TJ.

Transferring code and data between the MSCC 11 E2 and a PC requires the Mekatronix bidirectional serial communications board (MB2325). The 2.4" x 2.4" MSCC11 is a completely functional controller that's useful for a wide variety of embedded applications.

The MSCC 1 1E2 provides 2 KB of EEPROM, more than enough to program TJ to do incredible stuff. As Figure 2 shows, it features eight 3-wire inputs (i.e., 5 V, ground, analog signal) on port E via connectors J44–J5 1, eight 3-wire powered digital outputs on port B via connectors J9–J16, and eight 3-wire powered bidirectional digital signals on port C via connectors J1–J8.

TJ employs the unregulated voltage power rail to drive the wheel servos and IR LEDs attached to port B. The regulated voltage rail always drives the microcontroller and the eight powered digital and analog inputs attached to port E. Up to eight 3-wire powered analog sensor connectors can attach directly to port E.

A 6-pin male header enables the MSCCl 1E2 to communicate serially with other MSCC 11 s or PCs via a 6-wire cable to the MB2325.

A serial communications port, supported by Motorola's freeware PC-BUG1 1 program, lets you download and upload code and data into TJ's EEPROM on the MC68HC11E2. The Mekatronix MB2325 provides the voltage conversion from logic levels to the RS-232C requirements.

RS-232C COMMUNICATIONS

Serial communication of data and code between a PC and an embedded microprocessor application requires RS-232C voltages to be converted to logic voltage levels and vice versa. Usually, this problem is solved by placing the conversion circuitry on the microprocessor application circuit board.

The embedded application itself typically doesn't require an RS-232C communication port except to download application programs and data or to upload data. Hence, such RS-232C voltage-conversion circuitry unnecessarily occupies valuable board space.

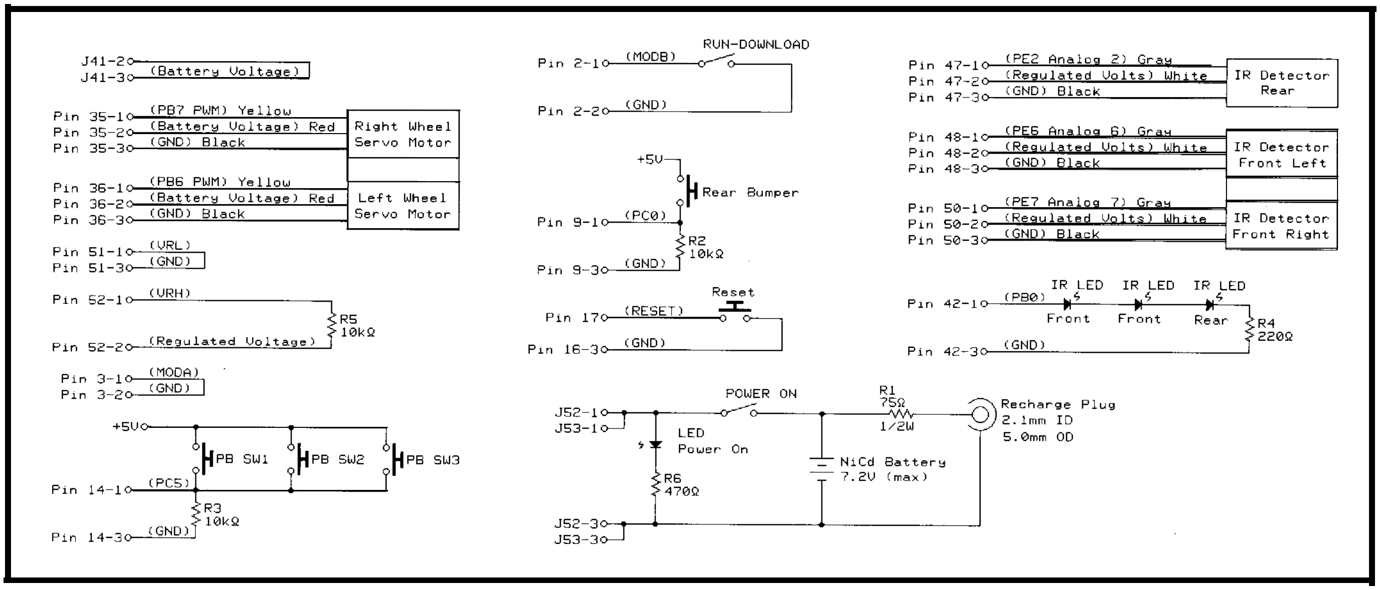


Figure 1—Peripheral circuitry connects to the MSCC11 to sensualize, empower, and actuate TJ. Note carefully how each feature connects to a particular male header. For example, the right IR defector connects to the header that brings out pin 50 of the processor, while the left motor connects to the header at processor pin 36, Port_B6. The power circuitry connecfs across pins 1 and 3 of Jumper 52 on the side of the PCB.

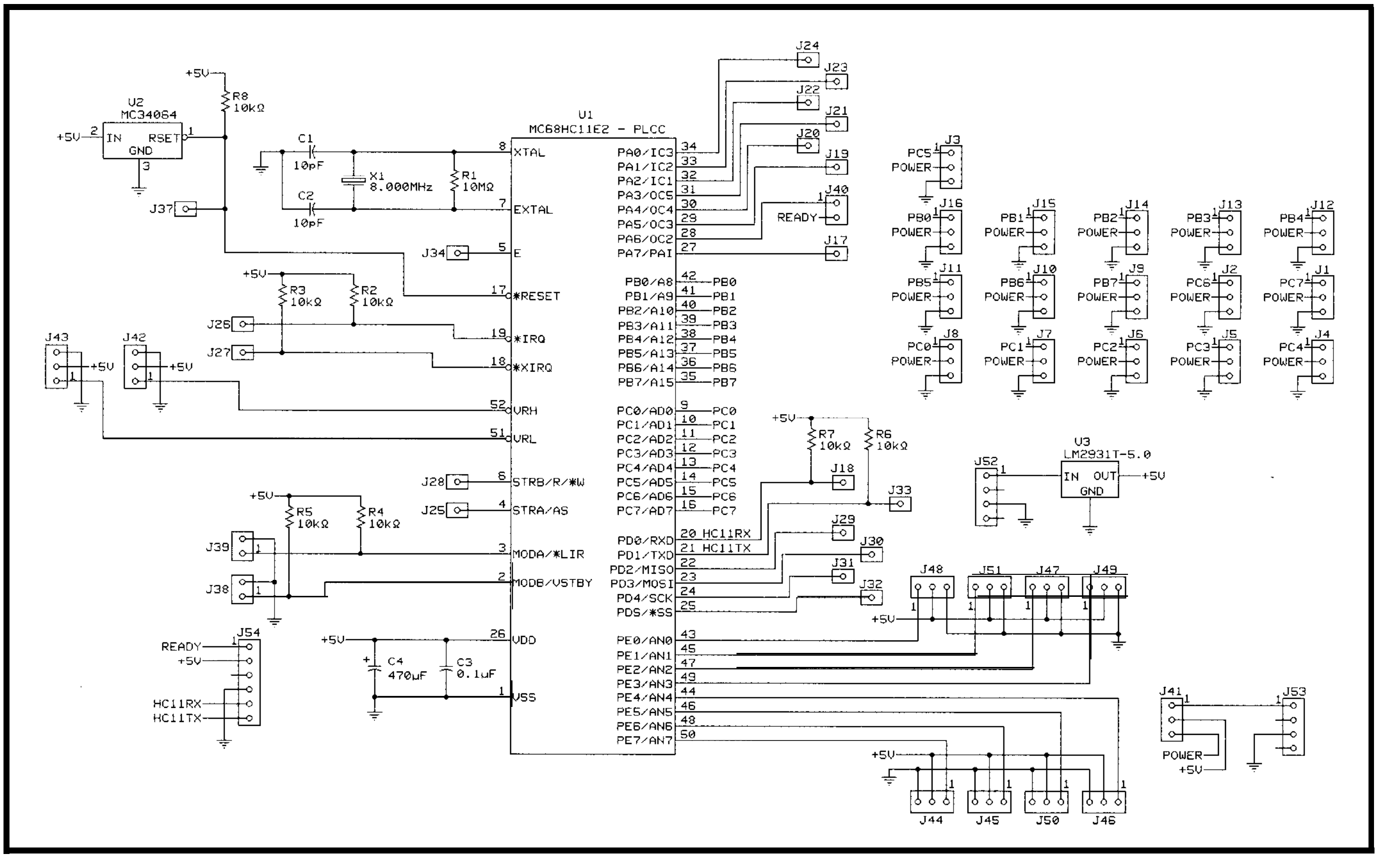


Figure 2—*The MSCC11E2* **circuit** *consists of* **an** *MC68HC11E2* **microcontroller surrounded** *by numerous male* **headers, designated by** *CON1, CON2, and* **CON3 with one, two, and** *three pins, respectively. These male* **headers provide easy access to** *the processor's powerful digital and analog I/O features. They also enable you to expand* **your** *TJ's sensor and actuator capabilities.*

By **offloading** the conversion hardware from the mobile robot, you can reduce power consumption and the cost per robot. Figure 3 depicts the communication-board circuit.

The MC 145407 charge pump converts the actual voltages. Header J3 provides a standard DB-25 connector for RS-232C communications, and header J2 connects to a Mekatronix 6-pin, logic-level, asynchronous serial communications cable.

MECHANICAL STRUCTURE

Photo 1 illustrates a particular TJ embodiment. The tabs on the side pieces make cutting the parts out more difficult. They add more strength to the joints, but they aren't essential.

Of course, you can design a completely different platform since the motors, microcomputer, and sensor circuits work on any body of comparable size. Geometric layout of the IR emitters and detectors, however, is critical, but feel free to experiment.

The circular plate mounts on the side plates like a reverse automobile engine hood. The rounded rectangular slots are wire conduits. The side plate supports the servos, one on each side.

The servos slide into the large rectangular opening in the side's center. Four small cross slats hold the side plates rigidly apart and provide a case for the six AA battery pack above the nylon skid.

Two slots on the top plate slip onto the "goose" necks in the front perpendicular to the floor. Holding the plate firmly against the vertical ends of the side pieces, the plate slowly rotates 90° to the rear as you release the pressure holding the plate vertically.

Two slots in the rear of the plate slide over the tabs with circular holes.

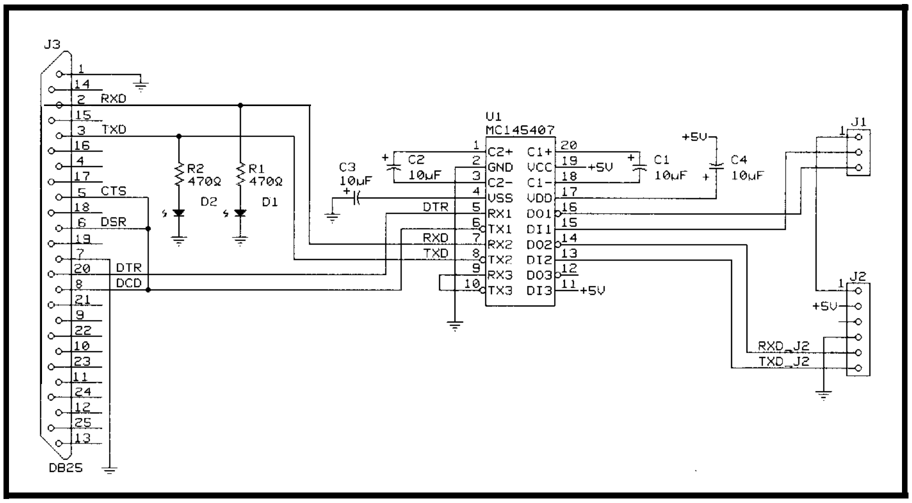


Figure 3—*With the MB2325, the MC145407 charge converts RS-232C standard voltage levels (±12 V) to logic levels (ground to 5 V) signals and vice versa. The 25-pin D connector gives the RS-232C levels necessary to communicate with your PC, while the 6-pin cable* **connector** *provides the logic levels for the MC68HC11E2's SCI port. The diodes' visual confirmation of communication makes them invaluable for debugging.*

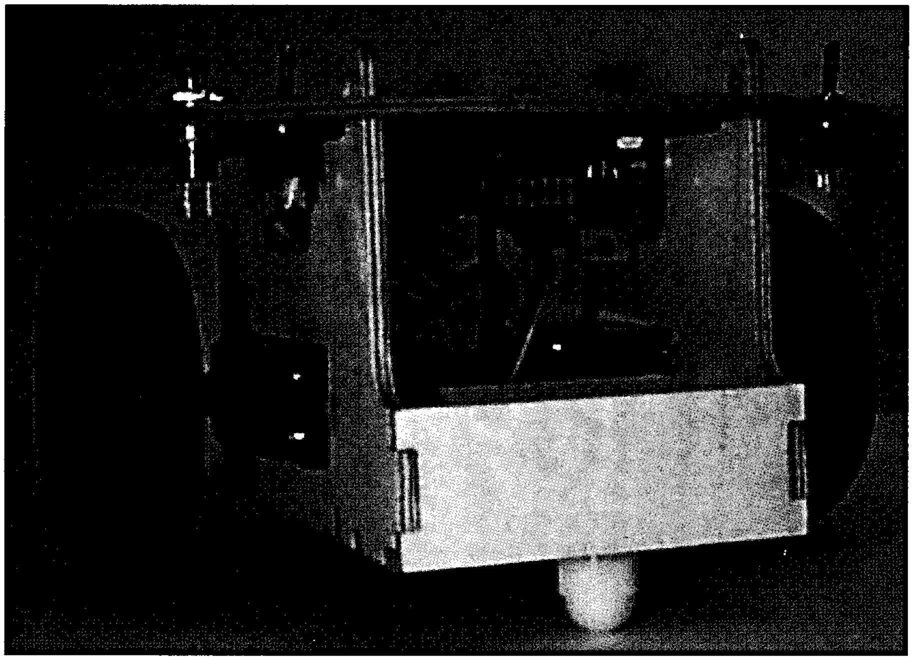


Photo I--This rear view *of a partially assembled TJ clearly illustrates the body assembly and the mounting of servos, wheels, rear nylon* **skid,** *and* **switches.** *The MSCC11 attaches* **underneath the** *plate* **with four** *4/40* **screws. The** *battery* **box** *above the skid provides* **stability and** *roadability when* **loaded with six** *AA NiCd batteries.*

A **0.25"** dowel slipped through the tab holes locks the top plate into place. A simple hinge, another cross plate in front, and a rear latch can eliminate this complex structure.

MOTORS AND SWITCHES

Any standard servo can be modified to create a DC gearhead motor. After removing the back-plate screws, take off the gear-box cover. Mount a servo horn on the output shaft and rotate the servo to the center of its range.

You can do it manually, or for more precision, you can write a program to do it automatically for you. In the rest of the procedure, avoid rotating the output shaft from its center position.

Remove the output gear and cut the plastic tab off the output gear. Take the potentiometer tab out of the output gear so it will not turn the shaft potentiometer.

Remount the gear and reassemble the servo. This almost ruins the servo as a servo, but instead you have a DC gearhead motor with electronic control!

The 3-pin female connector of the NovaSoft/Mekatronix MS410 servos or Aristo-Craft 03-410 Tracker Servo slips onto the MSCCl 1's male header without modification.

TJ possesses three switches mounted on the top plate in the rear-off/on, download/run, and reset. In the download position, the download/run switch forces the processor in special bootstrap mode on reset.

When the processor is in special bootstrap mode, you can download programs. In run mode, the processor changes on reset to single-chip mode and executes the downloaded program.

In addition to the control switches, three bumper switches mount on the front edge of the plate and one on the back. The MSCCl 1E2 board mounts underneath the plate in four mounting holes.

WIRING SENSORS

For a complete TJ assembly, all the switches, discrete components, and sensors must be wired to the MSCC-11 E2 as in Figure 1.

The IR emitters (IRE) and detectors (IRD) should be mounted on the top and bottom, respectively, of the top plate.

One IRD can be attached with velcro underneath the left front of the top plate, just outside the left side plate. The other IRD can be mounted on the right. For the best sensitivity in detecting objects, the IRDs should be splayed out from straightforward.

The front bumper switches are wired in parallel. The back bumper is wired separately to permit TJ to differentiate a front or back collision.

You can also wire each bumper switch separately and bring them into different pins of port C. This approach enables TJ to determine which bumper switch or switches have closed, providing a tactile view of objects about the robot.

PROGRAM DEVELOPMENT

The next task is to program behaviors. TJ can do a surprising number of actions-avoid collisions, follow walls, trace out geometric patterns an the floor, and so forth. You can add more sensors, but there's plenty to do with just the IR and bumper.

For example, a reinforcement learning program lets TJ learn how to avoid obstacles using the bump sensors as negative reinforcement. Only your imagination and 2 KB of EEPROM limit what TJ can do!

BASIC, C, and Forth compilers and assemblers are all available for the MC68HC11. If you're familiar with the MC68HC 1 1E2 assembly language, you can program the servos and the sensors from the information provided by the circuit diagrams.

Listing 1 presents a sample servo driver code in Image Craft C (ICC1 1) with embedded assembly code for efficiency.

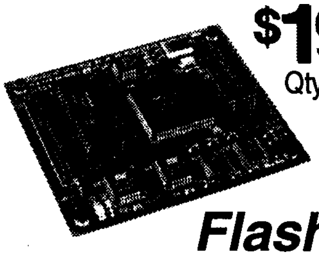
The servo driver uses OC2 to time the pulse widths of both servos, explaining the complication in selecting thedifferent signal-states. The servo outputs drive PB6 and PB7 at the program-generated duty cycle.

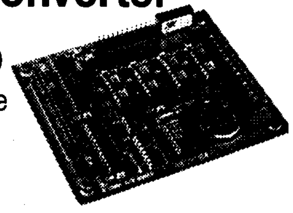
The wheels don't move when the duty cycle equals 3000 cycles **(1.5 ms).** For the right wheel, full forward equals 4000 cycles and full reverse is 2000 cycles. The left-wheel values have the opposite values.

PCBUG11

After writing a TJ program on your favorite editor and compiling or assembling the code to Motorola S19 format, you'll want to test it. Here's where Motorola's freeware PCBUG1 **1** and the MB2325 serial communications board come into play.

Mount TJ on a stand next to your PC so the wheels don't touch the table





Listing I--This *code uses* **OC2** *interrupts to control the timing for both servo motors. PO R TB provides the* ***I/O*** *pins for the* ***PWM*** *signals generated by the code. It is designed to operated a* ***TJ*** *using a sing/e-chip* ***computer board (e.g., MSCC1 1E2).***

```
#include <hcll.h>
#include <mil.h>
#define PERIOD 40000        /* 40,000 cycles = 20 ms at 2 MHz */
#define HALFPERIOD 20000
#pragma interrupt-handler servo-hand
void servo_hand();
unsigned width[2], current-width;
char signal-state;
char servomask[2];
/* current_width required in case width[] is changed while signal
     is being produced. Otherwise, shaking in servos results.*/
void init_servos(void)      /* Initalizes the servos */
{
   INTR_OFFO;
  CLEAR_BIT(TCTL1,0xC0);    /* Interrupt will not affect OC2 pin */
  width[0] = width[1] = 0; /* Set PWMs to 0 first */
  TOC2 = 0:                 /* First OC2 int occurs at TCNT=O */
  current-width = 0:
  signal-state = 0;         /* Initial state of signals */
  PORTB = PORTB & 0x3F;     /* Motors start turned off */
  servomask[0] = 0x40;
  servomask[1] = 0x80;
  SET_BIT(TMSK1,0x40);      /* Enable OC2 interrupt */
   INTR_ONO:
}
void servo(int index, unsigned newwidth)
/* Sets indexed servo to newwidth duty-cycle pulse width */
{
/* if (index) width[1] = newwidth; else width[0] = newwidth;*/
  asm("ldab %index \n"      /* Assembler to implement C code */
       "asrb \n"
       "ldy #_width \n"
       "aby \n"
       "ldd %newwidth \n"
       "std 0,Y");
}
void servo-hand 0
{
  char  odd;
  int  index;
  unsigned int pwidth;
/* signal-state = 0 -> servo0 on: signal-state = 1 -> servo0 off
    signal-state = 2 -> servol on: signal-state = 3 -> servol off */
  signal_state &= 0x03:     /* Only use last 2 bits */
  index = signal-state:
  asm("ldaa %index \n"
       "lsra \n"
       "staa %index");
  odd = signal-state & 0x01;
  pwidth = *(width+index);
  if ((pwidth == 0)&&(!(odd))) {
    TOC2 += HALFPERIOD;
    signal-state++:
  }
  else  {
    if (!(odd))
      TOC2 += pwidth:
    else if (odd)
      TOC2 += PERIOD - current-width;
      PORTB ^= *(servomask+index);
      current-width = pwidth;
  }
signal-state++:
/* Signal 0 goes on bit 6 of PortB, and Signal 1 is bit 7 */
CLEAR_FLAG(TFLG1,0x40);    /* Clear OC2I flag */
}
```

top. Plug the 6-wire serial cable into TJ at one end and the MB2325 at the other end.

Connect the MB2325 directly to a serial cable or 25-pin D-connector on the personal computer. Place TJ into download mode by setting the download/run switch appropriately and pressing Reset.

Now, execute PCBUG1 **1,** invoking the E2 version. Follow the instructions for changing EEPROM, and load your program. Your program should specify onboard EEPROM or RAM addresses.

To execute your program, switch to run mode and press Reset. Debug and enjoy!

AUTONOMOUSANDAFFORDABLE

My students and I have demonstrated that, through construction, an autonomous, programmable, mobile robot can be affordable.

An assembled version of TJ sells for $189, and an unassembled kit is available for $129. ■

Many thanks to the students in the Machine Intelligence Laboratory at the University of *Florida who* assisted *in the design of TJ. Erik de la Iglesia, Scott Jantz, and Chris Gomez designed the MSCC11 computer board. The platform evolved from ideas by Scott. Ivan Zapata developed the software. Scott and Ivan also constructed and* tested prototypes.

Keith Doty has been a professor at the University of Florida, researcher, and industrial consultant for over 25 years. His current interests include behavior-based, sensory-driven autonomous mobile robots and applied machine intelligence. You may reach Keith at doty@mil.ufl.edu.

REFERENCES

[1] www.mil.ufl.edu/imdl/handouts

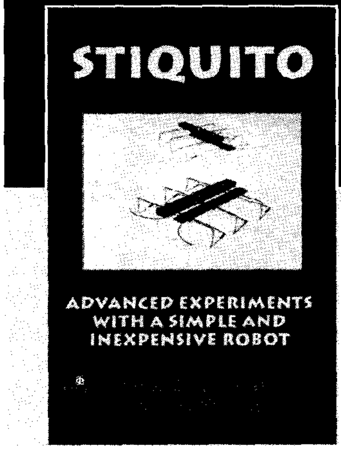
SOURCES

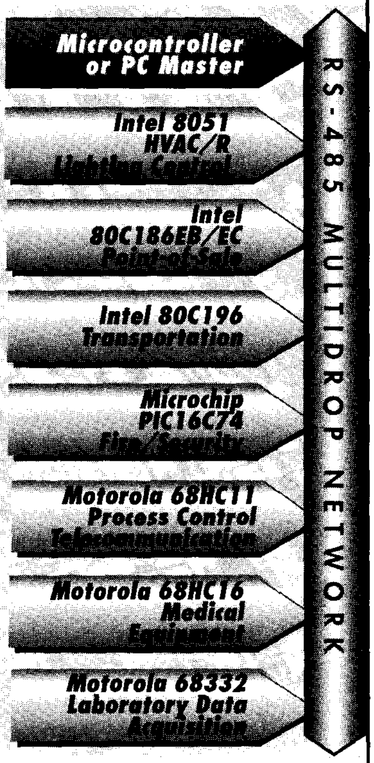
TJ kits and assembled versions
NovaSoft / Mekatronix
4813 NW 19 Pl.
Gainesville, FL 32605
(352) 392-4951
Fax: (352) 392-4976

PCBUG11
Motorola
Fax: (602) 3028157
BBS: (512) 891-3733
crc@crc.sps.mot.com
www.mcu.motsps.com/freeweb/
pub/mcu11/pcbug342.exe

I R S

404 Very Useful
405 Moderately Useful
406 Not Useful




Microcontroller or PC Master
Intel 8051 HVAC/R Lighting Control
Intel 80C186EB/EC Point-of-Sale
Intel 80C196 Transportation
Microchip PIC16C74 Fire/Security
Motorola 68HC11 Process Control Telecommunication
Motorola 68HC16 Medical Equipment
Motorola 68332 Laboratory Data Acquisition
RS-485 MULTIDROP NETWORK

FEATURE ARTICLE

Bill Payne

A Networking Primer

Part 3: Interconnecting Devices

At times, network communication gets as complicated as psycho babble. Bill talks about various network components, explaining the protocols and how to ensure that all the parts can talk to each other.

In this last installment, I cover the definition and operation of devices that interconnect networks. Each of these devices-repeaters, bridges, switches, routers, brouters, and gateways-fills a specific need of interconnecting LANs.

Limitations, such as the physical length of the network, are overcome by these technologies. And, they enable multiple LANs to communicate even though each may use different access methods and protocols.

Driving the need to interconnect networks are interoffice communication, E-mail, and centralized file and print sharing. Centralized LAN management helps in problem determination and resolution.

REPEATERS

Figure la shows a simple repeater. It operates at the Physical layer of the OSI model and overcomes the physical distance limitations of the wiring media. It can also increase the number of devices allowed on a LAN segment.

The repeater is not intelligent. It regenerates the received signal and retransmits it along another segment. It deals only with the raw data bits and the physical aspects of the LAN media.

A repeater can only interconnect LAN segments using the same media-access protocol. If one LAN segment uses the Ethernet media-access protocol and another Token Ring, they cannot be interconnected with a repeater.

In the Ethernet environment, repeaters must follow the 5-4-3 rule, depicted in Figure 2. The rule states that a LAN can comprise a maximum of five segments using a maximum of four repeaters. Also, only three of these segments can be populated with devices.

The two nonpopulated segments typically extend the network's physical length. These so-called link segments are found in places such as the wiring between floors in a building and between wiring closets.

In 10BaseT Ethernet environments, every hub acts as an active multiport repeater. The interconnection of these hubs must also follow the 5-4-3 rule.

By contrast, in a Token Ring network, every connected device is an active repeater, regenerating received signals. The 5-4-3 rule doesn't apply.

The downside to using repeaters in an Ethernet environment is that all signals are propagated to all segments. As new devices are added to a segment, the total network traffic increases.

The limitations of the usable bandwidth of your media-access method must be understood. As I pointed out in Part 2, Ethernet-usable bandwidth shouldn't exceed 45% utilization. Token Ring-usable bandwidth shouldn't exceed 80% utilization.

BRIDGES

Bridges operate at the Data Link and Physical layers in the OSI model and interconnect LANs. They bridge data between two independent LANs as shown in Figure 1 b.

Bridges filter data passed across them. Each frame's Media Access Control (MAC) address is checked to determine whether it should be forwarded or not. Frames with the same MAC address as the LAN they're operating on are ignored by the bridge.

In this way, a bridge isolates each LAN from collisions occurring on other LANs. Such isolation creates what is referred to as a "collision domain."

In a proper bridge setup, only 20% of the LAN traffic forwards. The remaining 80% should be local.

Bridges only forward frames containing user data. Frames used for tasks such as network management aren't forwarded. Each forwarded frame must have a valid checksum and not be addressed to the bridge itself.

Bridges are rated by the number of frames they can forward per second [i.e., their filtering or forwarding rate). There are three basic types of bridges-transparent, source routing, and source routing transparent {SRT}.

TRANSPARENT BRIDGES

A transparent bridge is defined by the IEEE in the 802.1D specification. This true plug-and-play device can be used by any protocol adhering to the IEEE 802 specifications.

On powerup, transparent bridges forward all frames received on each segment. As a frame is received, the source address is stored in a bridge-internal table. The bridge thus learns the address for the segment where the frame was received, which is why it's sometimes called a learning bridge.

All frames on this known segment not having the same address for both their source and destination are forwarded. The addresses of each segment are stored in a filtering database internal to the bridge. The database uses a flat-addressing scheme, so every device has a separate address entry.

Transparent bridges can operate in one of five possible states-disabled, blocking, listening, learning, and forwarding. Each state is defined by the IEEE in the 802.1D specification.

A disabled bridge doesn't forward frames or learn. This state is entered and exited via management commands sent to the bridge.

Blocking bridges also do not forward frames or learn. The bridge continues to participate in all spanning tree protocol operations.

When listening, a bridge enters the learning state. All bridge ports are active, but no evaluation of the frame MAC addresses occurs. This transitional state occurs when the bridge is brought from the blocking or disabled state into the frame-forwarding state.

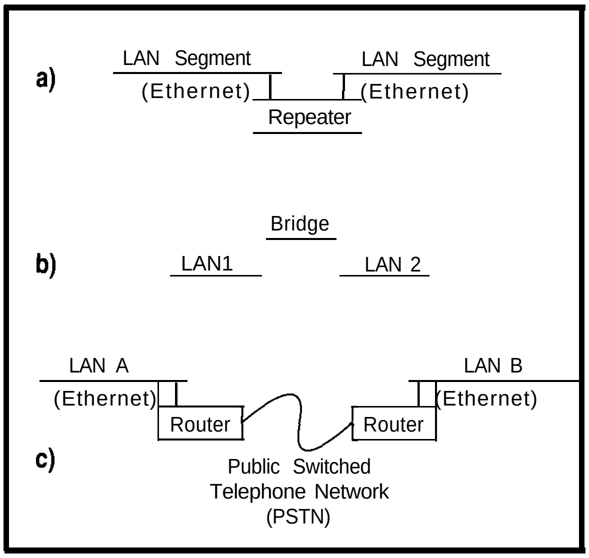


Figure la--Repeater. deal *only with the raw data bits and the physical aspects of fhe media. b-Bridges operate at both the Data Link and Physical layers in the OSI model. c—Routers deliver data packets across an internetwork via a telephone inferconnect, more correctly known as a Public Switched Telephone Network (PSTN).*

Once a bridge is in the learning state, it prepares to forward frames. The MAC address of each frame received is added to the filtering database. This state is entered through operation of the spanning tree protocol.

In the forwarding state, the bridge actively participates in frame-forwarding. Each bridge port is still learning and adding new entries as they occur to the filtering database.

SPANNING TREE PROTOCOL

This protocol is a bridge-hierarchy protocol (see the IEEE 802.1D specification for details] that lets bridges communicate with other bridges on a LAN, enabling the network to detect when bridge or segment failures occur. In this event, the network dynamically reconfigures the routes and bypasses the failed segment or bridge.

Primarily, this protocol organizes routes between redundant bridges to eliminate bridging loops. Redundant bridge paths in a transparent bridge environment can be fatal.

Bridging loops (see Figure 3) are due to the primary and redundant bridges continually updating their filtering databases. Because of propagation delays on the media, one bridge receives the frame packet before the other one.

So, after the primary bridge receives a frame packet, it updates its filtering database to point to the frame packet's source. It then forwards the packet to the redundant bridge.

The redundant bridge sees the frame packet and assumes it originated on the same side of the bridge it was received on. It then takes the frame packet and forwards it back to the originating LAN segment. This process continues indefinitely.

The spanning tree protocol overcomes this infinite loop by selecting one redundant bridge as the designated bridge and the other as the backup.

The protocol uses a special frame packet, the Bridge Protocol Data Unit (BPDU), to communicate between bridges. The BPDU exchanges enable the network to dynamically reconfigure after a failure.

To assist in reconfiguration, each bridge has a unique eight-byte ID number. The ID's first two most significant bytes are assigned by the network administrator. The last six bytes are the manufacturer's assigned MAC address for the bridge-internal port adapter.

The spanning tree protocol uses this ID to select the designated and backup bridges. On powerup, each bridge transmits a BPDU with its unique ID on all ports.

If the bridge receives a BPDU with a lower bridge ID than its own, it stops transmitting its own BPDU and starts forwarding the BPDU with the lower bridge ID. The BPDUs are transmitted at 2-s intervals. All bridges can respond to their own specific address as well as a bridge's assigned multicast address.

Topology changes when an administrator issues a change command or through a segment or bridge failure. During a topology change, all bridges stop forwarding frame packets to prevent temporary loops.

SOURCE-ROUTING BRIDGES

Source routing is an IBM specification that relates to data transmission in an IBM SNA environment. It is typically found only in Token Ring networks as depicted in Figure 4.

The source-routing bridge doesn't maintain a filtering database like the transparent bridge. Instead, each device on the network maintains its own dynamic table of routes.

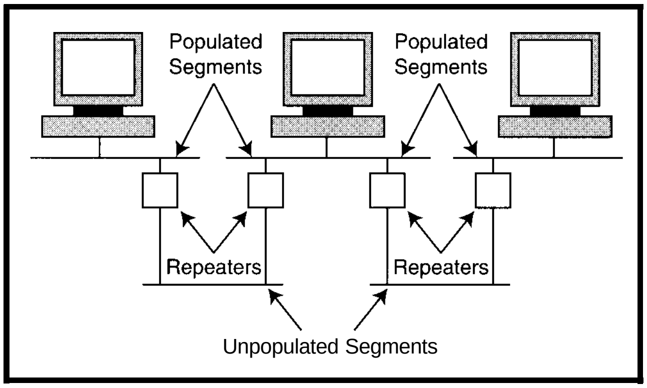
When data needs to be transmitted to another device, the transmitting device performs a route-determination operation. This operation, which builds the route table internal to the device, consists of the transmitting device sending a hello packet to the destination device.

The transmitting device takes the route information from the first reply back from the destination device and adds it to the route table. All other replies from the destination device are ignored.

IBM refers to this process as an exchange identification (XID) packet within the Logical Link Control (LLC) sublayer. As I explained in Part 2, the LLC sublayer within the Data layer of the OSI model handles error control, flow control, framing, and MAC addressing.

A problem inherent in the route-determination process is that it generates massive amounts of network traffic. To resolve this, IBM developed two states for a source-route bridge.

Figure 2—*The* Ethernet 5-4-3 *rule states that fhere can be no more than five physical segments in the LAN, a maximum of four repeaters, and only fhree of the five segments can be populated with nodes.*



In the all-routes-broadcast state, all bridges forward all packets. But, this system has problems on a LAN composed of multiple Token Rings interconnected by multiple bridges.

For example, suppose two Token Rings are connected via two bridges (a common scenario in a network where redundancy is used for fault tolerance). The destination ring receives two frames for each frame generating on the source ring. The amount of traffic grows exponentially as more rings interconnect and as more devices enter each ring.

In the single-route-broadcast state, on the other hand, only the bridge forwards packets. The redundant bridges remain in the all-routes-broadcast state but ignore all data packets with the single-route-broadcast bit set. For this system to work, every device on the ring must be configured to transmit all route-determination packets as single-route broadcasts.

SOURCE-ROUTE TRANSPARENT BRIDGES

The source-route transparent (SRT) bridge incorporates the features of both transparent and source-route bridges.

It acts as a source-routing bridge for frames carrying source-routing information. When it receives a frame which doesn't have source-routing information, it acts as a transparent bridge with a filtering database.

The SRT bridge is usually found in mixed environments with IBM SNA devices and nonSNA devices, but presently, no standards exists for it. The various manufacturers of these bridges implement them as they see fit.

SWITCHES

Switches, a newer LAN technology, are a hybrid between a repeater and a bridge.

A classic bridge operates by bringing each received frame packet into cache memory. After the address information is processed by the bridge CPU, the frame packet is either discarded or forwarded. This process adds a considerable amount of latency to each forwarded message.

A switch doesn't cache the entire message. Instead, it only processes the MAC address header on the frame packet. No further processing is done.

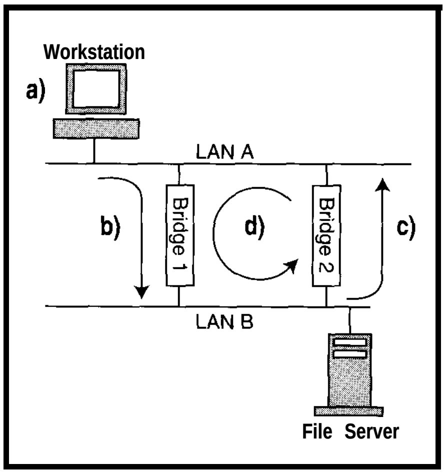


Figure 3a—*In a bridging loop, the workstation sends a frame packet to fhe file server on LAN 5. b-Bridge 1 receives the frame packet on LAN A and forwards if to LAN B. c-Bridge 2 receives the frame packet on LAN B and forwards it fo LAN A. d—This cycle repeats indefinitely. The spanning free protocol prevents fhis infinite loop by defining one bridge as the designated bridge and the other as the backup.*

Switches therefore give a much higher packet-per-second throughput. They still provide the basic traffic filtering needed on a multisegment LAN but at a much higher throughput.

ROUTERS

Routers deliver data packets across a internetwork independent of the communications media (see Figure lc). They work up to the Network layer of the OSI model.

Routers use software addresses that are hierarchical as opposed to the flat-addressing scheme used in bridges. The flat addresses used in a bridge define only a single hardware address, whereas hierarchical software addresses describe the entire network.

Routers can convert a data packet from one protocol to another. This feature is useful if two LANs need to be connected but are in different geographic areas.

A technique known as tunneling is sometimes used to connect two LANs across an analog telephone line. The originating LAN protocol data packet is encapsulated in the transport protocol and sent across to the other router. On receipt, the transport protocol is stripped from the encapsulated data packet, leaving the original data packet.

Routers provide two types of delivery services-connection-oriented and connectionless- or packet-oriented.

Connection-oriented services guarantee packet delivery. The existing X.25 Public Switched Telephone Network (PSTN) provides the most common connection-oriented services.

With this type of service, a router must establish a connection with a remote router before transmitting data packets. This connection becomes fixed until it's no longer needed.

All transmitted data packets arrive at the receiving end in the transmitted order. The originating router informs the receiving router of the maximum packet size used in the communications session.

An analogy to this type of service is a telephone call. You can't begin communicating with the other party until they answer the phone. And, when the other party answers, communication begins only if you both speak the same language.

Connectionless or packet services do not guarantee delivery of any data packets. Frame relay is the most common connectionless or packet service.

Each data packet is routed dynamically, based on the network address. Transmitted packets arrive at the receiver in no specific order. At the receiving end, they are reassembled based on a sequence numbering scheme.

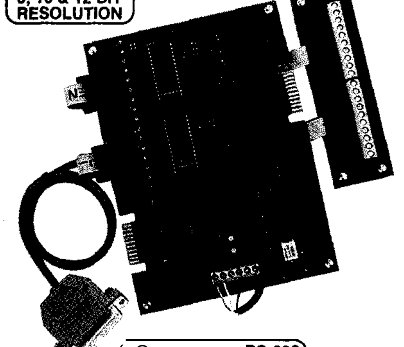
Bandwidth is allocated dynamically to all users. If network traffic is high, packet services degrade equally to all users. No one is denied network access.

ROUTING PROTOCOLS

Routers use very specific protocols to communicate with each other. They discover routes, update routing information, send alerts about traffic congestion, and advertise the cost of each path in hop counts. (A hop refers to the passing of a packet from one router to another.)

Routing protocols should not be confused with the OSI network-layer protocols. Instead, they are based on one of two distributed routing algorithms known as the distance-vector and link-state algorithms.

The distance-vector algorithm calculates routes based on information gathered from neighboring routers. Due to its simplicity, this algorithm



was the first one implemented.

But, its biggest disadvantage is the amount of time it takes for all routers in the network to exchange information (i.e., the convergence time).

The link-state algorithm calculates routes based on information actively learned about the network. These routers build their route tables by discovering their neighbors.

Once the router has discovered its neighbors, this information is sent to all neighboring routers in a special link-state packet (LSP). Each of the neighboring routers then forwards the LSP to all the other routers in the network.

Therefore, all the routers in the network can converge much more quickly after a topology change. The Open Shortest Path First (OSPF) routing protocol is an example of a link-state algorithm protocol.

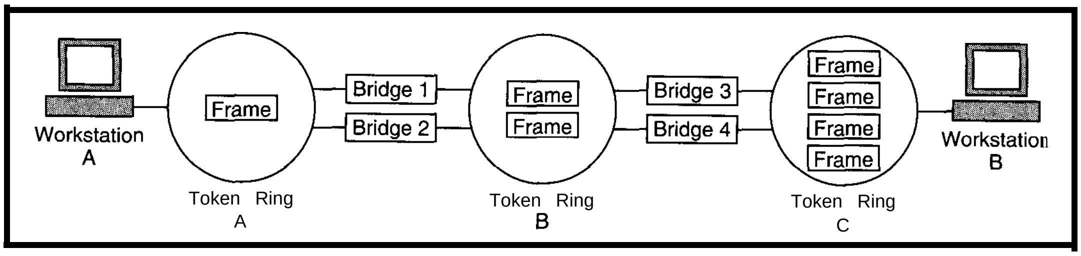


Figure 4—Problems arise when multiple Token Rings are connected with redundant bridges. Workstation A sends a frame packet to workstation B on Token Ring A. Both bridges 1 and 2 place copies of the original frame packet onto Token Ring B. Bridges 3 and 4 place copies of replicated frames onto Token Ring C. Workstation B responds to the original frame packet as well as the three copies. IBM resolves this problem via two possible router states—all routes broadcast and single route broadcast.

BROUTERS

A brouter is an intermediate device which combines the functionality of a transparent bridge with a router. Protocols which cannot be routed (e.g., IBM's SNA) are bridged to interconnecting LANs. Protocols which can be routed are passed through the router.

GATEWAYS

Gateways are software-based programs that translate between incompatible protocols. They function at all layers of the OSI model.

A good example of a gateway is one which converts cc:Mail to Simple Mail Transfer Protocol (SMTP), providing a connection to the Internet. Gateways are usually very CPU and memory intensive.

YOU'RE CONNECTED

This series of articles should give you a basic but complete overview of the principles involved in networking and intemetworking. The information I've presented only scratches the surface of the technologies involved. Each area could fill a separate book.

I trust I've given you enough information about LAN operating principles that I've removed the veil of black magic that most people associate with LANs. ❑

Bill Payne has many years' experience as digital design engineer. He is also a Novell Master CNE, Novell Master GroupWise CNE, and a Novell CNS. You may reach Bill at bpayne@kramerkent.com.

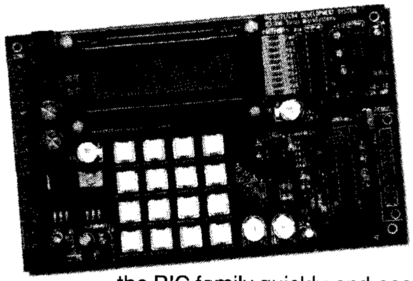
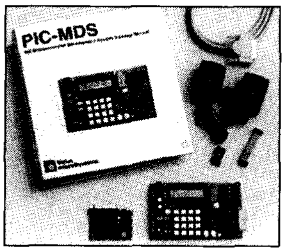
SOURCES

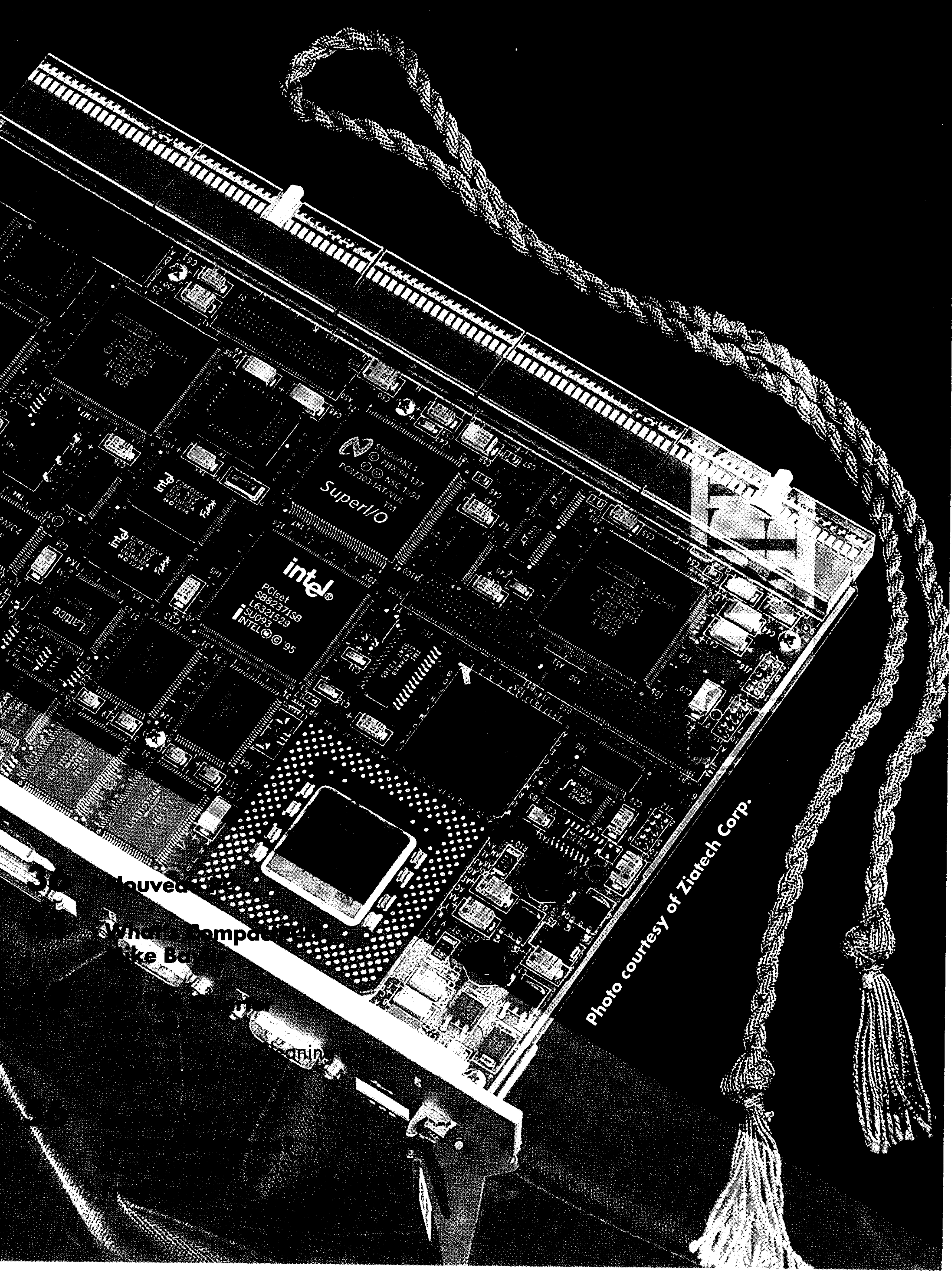
IBM cabling standards
IBM Corp.
1000 NW 51 St.
Boca Raton, FL 33432
(407) 443-2000
Fax: (407) 443-4533

Novell
Novell, Inc.
1555 N. Technology Way
Orem, UT 84057-2399
(801) 222-6000
Fax: (801) 861-3933
www.novell.com

I R S

407 Very Useful
408 Moderately Useful
409 Not Useful



3[illegible] Nouveau [illegible]

[illegible] What's CompactPCI
Mike Ba[illegible]

[illegible]

[illegible] leaning [illegible]

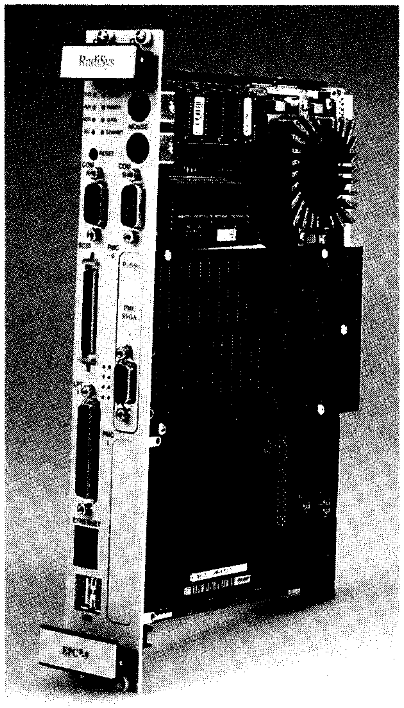
3[illegible]

Photo courtesy of Ziatech Corp.

EMBEDDED PC WITH VME BUS

The EPC-9 is a dual-slot PC/AT-compatible board with a Pentium processor that's designed for high-performance embedded applications. It is a full-featured VME-bus board with onboard Ethernet and SCSI II, IDE hard drive, USB, and two PMC slots for PCI-bus expansion. Its modular processor design provides for a field upgrade to a Pentium Pro processor.

The EPC-9 features four 72-pin SODIMM memory sockets, 256-KB secondary cache, an autosensing 1 O/I OOBaseT Ethernet controller with front-panel RJ-45 connector, and a fast SCSI II controller with front-panel connector. I/O is accomplished via two USB ports, one parallel and two serial ports, mouse and keyboard controllers, and an EXM-bus expansion.

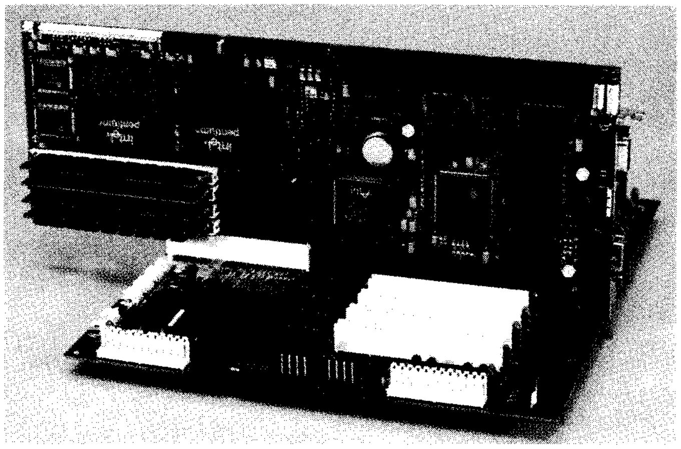


An optional 2.5" IDE 1.44-GB hard drive fits on the two-slot EPC-9, as does an optional SVGA PMC module. By offering video in a low-cost PMC form factor, the EPC-9 provides an optimal video solution that can beeasilyupgraded. Should an application not require video, two PMC slots are available for other functions.

The EPC-9 is available with a 100-,133-, 166-, or 200-MHz Pentium-processor-based module which is upgradable to a Pentium Pro module. The EPC-9 with a 133-MHz Pentium processor and 8-MB DRAM sells for $3527 in quantities of 100.

RadiSys Corp.
5445 NE Dawson Creek Dr.
Hillsboro, OR 97124
(503) **615-1100**
Fax: (503) 615-l 150
www.radisys.com

#510



DUAL **PENTIUM** SBC

The 2108 single-board computer combines dual-Pentium processing powerwith a variety of high-performance features. High-level graphics is achieved via the Chips and Technologies65548 CRT/Flat-Panel Controller with bit-block transfer and GUI accelerator featuresorvia the Cirrus 5434/ 5446 CRT controller with 2 MB ofvideo memory. The flat-panel interface features a PC video-input connector, while the CRT interface has a feature connector. Both graphics controllers are on the Local bus, providing optimal graphics performance for imaging and multimedia applications. Resolutions up to 1280 x 1024 pixels are supported with a full 1 MB of video RAM.

The 2 108 uses the Triton II 82340HX chipset, which supports USB and infrared I/O. Memory is supported with parityorerrorcorrection to5 12 MB of 32-bit-wide EDO DRAM. Up to 5 12 KB of L2 synchronous cache is also onboard.

A RISC-based Adaptec AHA 2940 Ultra Fast SCSI interface provides data transfers up to 40 MBps in synchronous or asynchronous modes. Combined via a PCI bus is Intel's 1 O/I OOBaseT LAN controller with support drivers for a number of operating systems.

The 2 108 includes two full 16C550 serial ports, printer port with ECP support, and floppy and enhanced IDE support as part of the standard I/O package. Embedded PC and system BIOS features include extensive power management, temperature sensing for processorclock speed reduction, no fan for speeds up to 133 MHz, optional fast boot, and boot without keyboard. A full PICMG interface facilitates many off-the-shelf PCI or ISA products.

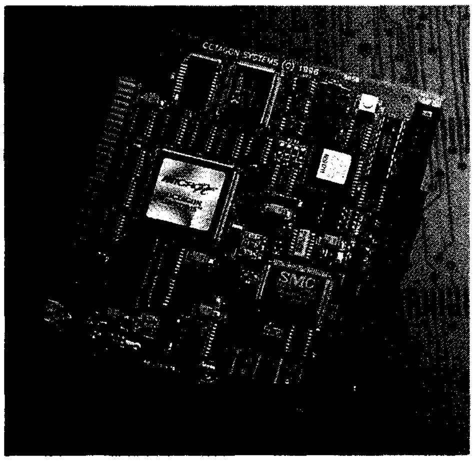
Pricing for some models starts at $995 in single quantities.

TME
5149 Bradco Blvd.
Mississauga, ON
Canada **L4W 2A6**
(905) 625-3203
Fax: (905) 625-3717
sales@tme-inc.com
www.tme-inc.com

#511

SINGLE BOARD COMPUTER

The new PC Microcontroller series of single-card industrial computers combines industry-standard PC architecture with industrial-class I/O and an extensive suite of embedded software. The first two products in the family-the 6050 and **6040**—have a robust feature set thot includes digital, analog, and serial I/O. These cards are ideal for a wide variety of applications, including machine control, GPS, PLC monitoring, weighing machines, point-of-sale systems, and operator interface.



The 6050 and 6040 microcontrollers both feature two 16C550-compatible serial ports, an enhanced parallel port, two optoisolated interrupts, 24 lines of bit-programmable DIO, keyboard and speaker ports, real-timeclock, watchdog timer, 1 -MB flash, 2-MB DRAM, and an AT battery onboard. The 6040 microcontroller adds 10 lines of 12-bit analog I/O. The flash-file system on both cards lets users save program and data files in the same manner as with a hard disk. The cards can operate stand alone or be expanded via the ISA **bus.**

Several factors contribute to the cards' ruggedness. In addition to the -40 to 85°C temperature range, the input and output liines on the cards are protected. The serial ports can withstand an 8-kV discharge without failure, and the external interrupts are optoisolated to protect the card from high-voltage transients.

The software suite includes diagnostics, networking, DOS 6.22, and CAMBASIC, a fast multitasking control and data-acquisition language. Embedded in flash memory, the PC Microcontroller software suite eliminates laborious tasks such as writing hardware drivers. In addition, the family is compatible with many of the popular real-time operating systems (e.g., QNX), as well as most other PC software tools.

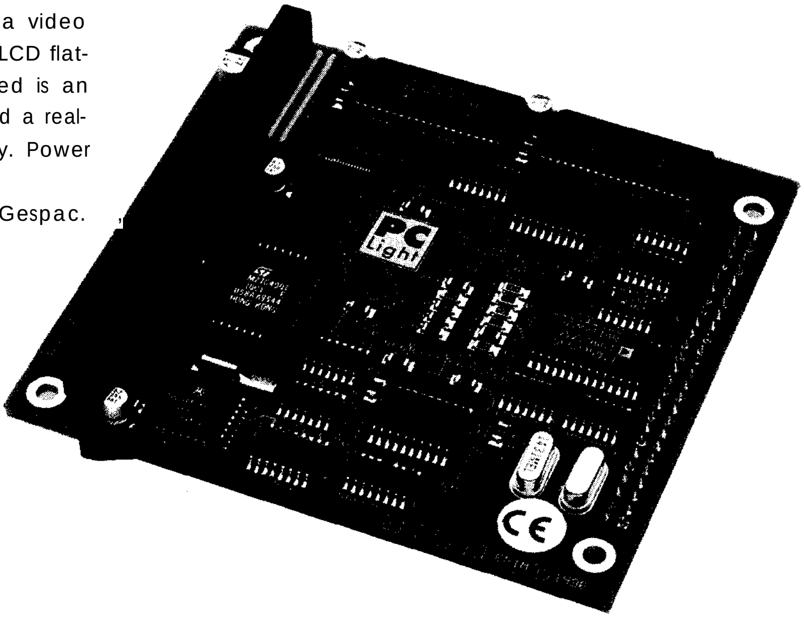
In small quantities, the 6050 and 6040 sell for $374 and $562, respectively.

Octagon Systems
65 10 W 91 st Ave. • Westminster, CO 80030
(303) 430- 1500 • Fax: (303) 426-8 126
#512

ULTRA-COMPACT PC/ 104 CPU

The **PC light 8680V2-104** is an ultracompact PC/104-compatible computer from Groupe Erim. It includes 512-KB or 1 -MB DRAM, 5 12-KB or 1 -MB flash memory (300 KB for a RAM drive with BIOS, DOS, and drivers already installed), a video controller for direct connection to a 320 x 240 (‘/A VGA) LCD flat-panel display, and an RS-232 serial port. Also included is an interface for a 64-key matrix keyboard or XT keyboard and a real-time clock backed up by an external lithium battery. Power requirement for the unit is +5 V at 250 mA max.

The PC Light 8680V2-104 is distributed in the U.S. by Gespac.



Gespac
50 W. Hoover Ave.
Mesa, AZ 852 10
(602) 962-5559
Fax: (602) 962-5750

#513

'x86 SIMULATOR

Systems and Software is shipping a compact disc that contains two versions of the **VisualProbe** 'x86 Simulator. The Trial and Standard Editions are both source and symbolic debuggers for 32-bit protected-mode and 16-bit real- and protected-mode embedded C and assembly applications.

Also included is an extensive series of tutorials and additional information designed to educate the user about the functions, features, and capabilities of using a simulator as a debugger. Their goal is to explain the function of a simulator and how it works.

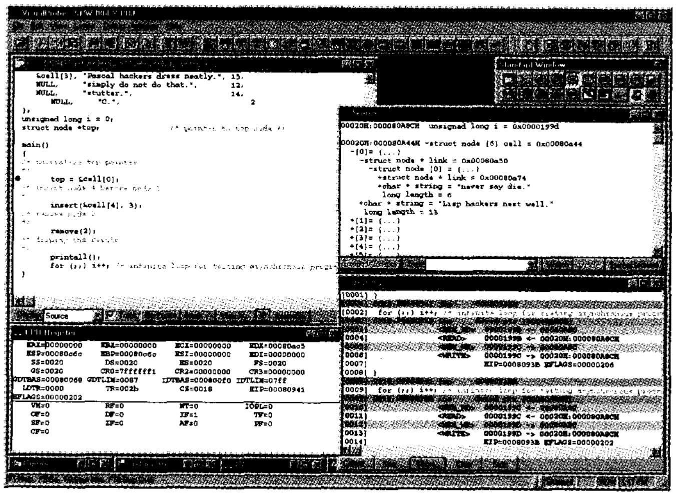
The Trial Edition provides all the capabilities necessary to help you evaluate the VisualProbe Simulator. Also, it enables you to teach about how to use simulators as debugging tools and how their functions can be used in developing embedded applications.

The Standard Edition offers simulation of the '386/'387, '486, and Pentium CPUs, a command window, unlimited trace, and an extensive list of peripheral models, as well as a printed manual. The Trial Edition is a fully functional subset of the Standard Edition. It simulates only the '386/'387 processor, it has no command window, trace is limited to 100 lines, and no printed manual or technical support is provided.

The Trial Edition is currently offered free (though shipping cost may be applied to some requests and to all express-shipment requests). The user may also upgrade to the more capable Standard Edition for $495. When users upgrade their VisualProbe, they can unencrypt the Standard Edition already present on the CD.

Systems and Software, Inc.
18012 Cowan Ave., Ste. 100
Irvine, CA 92714
(714) 833-1700 • Fax: (714) 833-1900

#514



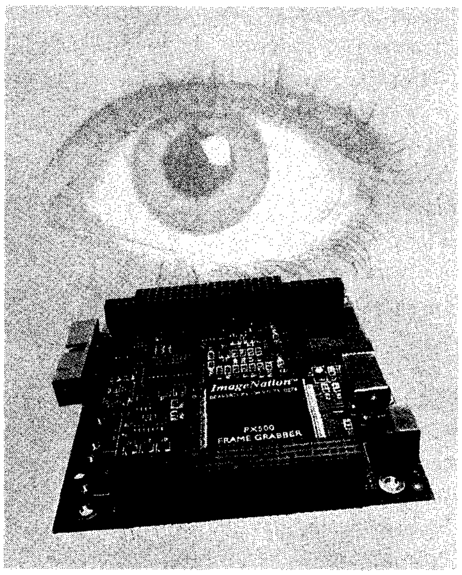
```
    &cell[3], "Pascal hackers dress neatly.", 15,
    NULL,     "simply do not do that.",       12,
    NULL,     "stutter.",                     14,
       NULL,     "C.",                          2
};
unsigned long i = 0;
struct node *top;
main()
{
        top = &cell[0];
        insert(&cell[4], 3);
        remove(2);
        printall();
        for (;;) i++;
}
```

```
00020H:000080A8CH  unsigned long i = 0x0000199d

00020H:000080A44H -struct node [6] cell = 0x00080a44
  -[0]= {...}
    -struct node * link = 0x00080a50
      -struct node [0] = {...}
        +struct node * link = 0x00080a74
        +char * string = "never say die."
         long length = 6
    +char * string = "Lisp hackers nest well."
     long length = 13
  +[1]= {...}
  +[2]= {...}
  +[3]= {...}
  +[4]= {...}
```

```
EAX=00000000    EBX=00000000    ECX=00000000    EDX=00080ac5
ESP=00080c6c    EBP=00080c6c    ESI=00000000    EDI=00000000
 SS=0020         DS=0020         ES=0020         FS=0020
 GS=0020        CR0=7ffffff1    CR2=00000000    CR3=00000000
GDTBAS=00080068 GDTLIM=0087     IDTBAS=000800f0 IDTLIM=07ff
 LDTR=0000       TR=002b         CS=0018        EIP=00080941
EFLAGS=00000202
 VM=0            RF=0            NT=0           IOPL=0
 OF=0            DF=0            IF=1            TF=0
 SF=0            ZF=0            AF=0            PF=0
 CF=0
```

```
[0001] }
[0002] for (;;) i++;
[0004]            <READ>    0000199B <- 00020H:000080A8CH
[0006]            <WRITE>   0000199C -> 00020H:000080A8CH
[0007]                      EIP=0008093B EFLAGS=00000206
[0008] }
[0009] for (;;) i++;
[0011]            <READ>    0000199C <- 00020H:000080A8CH
[0013]            <WRITE>   0000199D -> 00020H:000080A8CH
[0014]                      EIP=0008093B EFLAGS=00000202
```



PC/104-*Plus* FRAME GRABBER

The **PX104-Plus** is a precisevideo-capture module with PCI-bus compatibility in a PC/l 04-*Plus* package. Its high accuracy and low pixel jitter make it ideal for industrial and commercial machine-vision applications. The compact PC/l 04-*Plus* format simplifies integration and allows for a compact, rugged, cost-effective embedded PC-based machine-vision system.

The PX104-Plus **features** 256 gray-scale levels (8 bits) with a maximum pixel jitter of ±3 ns. It also has four multiplexed video inputs with automatic video-format detection and switching, as well as continuous, triggered, or software-initiated frame capture. It includes RS170/NTSC and CCIR/PAL compatibility. Vertical and horizontal cropping and pixel decimation reduce memory and data-transfer demands. Other features include a PCI-bus master design for real-time image capture at rates up to 13 1MBps to system RAM or VGA display, as well as horizontal and vertical sync output for precise synchronization of the frame grabber and camera. Instant resynchronization and dual strobe outputs make working with resettable (asynchronous) cameras easy. Optional support for non-interlaced video from progressive-scan cameras with image lengths up to 1024 lines is also provided.

The PX104-Plus comes with complete software support for 16-bit DOS and Windows 3.1, 32-bit Windows 95, and Windows NT applications, as well as compatibility with Visual Basic and C/C++ compilers from Borland and Microsoft. The unit sells for $895.

Imagenation
P.O. Box 276
Beaverton, OR
97075-0276
(503) 64 l-7408
Fax: (503) 643-2458
www.imagenation.com

#515

Nouveau PC

Mike Baylis

What's CompactPCI?

Moving PCI into the embedded world wasn't easy. *But, not only did CompactPCI* overcome the limitations of the desktop and industrial *PCI, it* also developed a *format* that hosts *3U* and *6U* boards *and* bridges to **ISA,** *VME,* and *STD 32.*

CompactPCI is a new high-performance embedded-bus standard gaining momentum among industrial-computer suppliers and users. Combining Intel PCI signalswith Eurocard packaging, it offers a rugged high-performance alternative to desktop PCI designs.

To better understand CompactPCI's advantages, let's start with a brief history of early Local-bus implementations and specifications, and then move on to more technical discussions of both PCI and CompactPCI. I'll also cover possible applications for this new standard.

IN THE BEGINNING

PC manufacturers were constantly striving to improve the performance of videographics adapters residing on the PC's expansion bus. Early adapter designs had to process low-level commands from the microprocessor.

The first step towards increasing throughput was adding intelligence to the adapter card so it could handle high-level commands and free up the microprocessor from screen-intensive operations. The ISA expansion bus had become a bottleneck as well, limited to 16-bit transfersat 8-l 0 MHz.

The next step was to move the intelligent video adapter from the slower expansion bus to the processor's Local bus and optimize the design to minimize or eliminate the wait states inserted in each bus cycle. For better or worse, the birth of Local-bus designs had begun.

VL BUS

Lack of an existing standard for interconnecting Local-bus devicesand PC architecture led a group of video-adapter manufacturers called the Video Electronic Standards Association (VESA) to come up with the W-bus specification. The first version defined two methods of interfacing to the processor's Local bus.

VL type A described a direct-connect scheme. The device connects directly to the processor's bus structure, requiring no additional system board logic.

VL type B defined a buffered approach. The Local bus was electrically isolated from the processor bus by a buffer/driver, which appeared as one load to the processor bus.

Both approaches achieved a maximum data-transfer rate of 132 MBps at 33 MHz during burst reads (cache-line fill operations) and 66 MBps on 32-bit write transfers.

So, why didn't the VESA spec become the prevalent standard?

First, it was rather shortsighted, being designed around the Intel '486. Next-generation processors needed the bus interface logic redesigned. VL bus didn't have clear electrical-design guidelines to ensure bus-design integrity.

As well, the VESA VL V. 1 .O spec called for automatic system-configuration support. But, it didn't define the location or format of Local-bus device-configuration registers.

ENTER PCI BUS

Aiming to provide a clearly defined and longer-term solution, Intel released the Peripheral Component Interconnect (PCI) specification in June 1992.

PCI uses a workstation approach for interfacing a Local-bus device to the processor bus. It combines the processor's level-2 cache controller with a bridge that acts as an interface between the processor, main memory, and the Local bus.

With the Local-bus interface independent of the processor's bus, the Local bus becomes processor independent. When a new processor becomes available, only the bridge chip needs replacing.

The workstation approach also enables the processor to access information from its cache, while the cache controller allows a PCI-bus master access to main memory or other PCI devices on the Local bus.

Intel thought it best to make the PCI spec an open standard. They helped form the PCI Special Interest Group, which defined the standard PCI spec as well as revisions for supporting 64-bit extensions and 3.3-V technology. Let's look at some of the other advantages PCI offers over VL bus.

On the PCI bus, all read and write operations are burst transfers lasting as long as the target can receive or send data. At 33 MHz, a maximum data transfer rate is 132 MBps using 32-bit transfers and 264 MBps using 64-bit transfers. PCI V.2.1, released in 1995, supported 66-MHz bus speeds, doubling the maximum data transfer rate to a whopping 528 MBps.

Multiple bus masters are supported under PCI, with each master being connected to an arbiter via a pair of bus request (*REQ) and grant (*GNT) signals. The arbiter is usually integrated into the host-to-PCI bridge chip.

Figure **1:** To support ***autoconfiauration on powerup, the Intel specification*** recommends a ***256-byte*** configuration register ***space,*** the first 64 registers ***being mandatory.***

Byte 3	Byte 2	Byte 1	Byte 0	Doubleword Number (in Decimal)
[illegible]		[illegible]		00
[illegible]		[illegible]		01
Class Code			[illegible]	02
BIST	[illegible]	Latency Timer	Cache Line Size	03
Base Address 0				04
Base Address 1				05
Base Address 2				06
Base Address 3				07
Base Address 4				08
Base Address 5				09
CardBus CIS Pointer				10
Subsystem ID		Subsystem Vendor ID		11
Expansion ROM Base Address				12
Reserved				13
Reserved				14
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	15

▨ Required Configuration Registers

Arbitration is allowed while the current bus master performs a data transfer by removing *GNT from this master and issuing it to the next owner of the bus. This hidden arbitration doesn't waste bus time in arbitrating cycles.

Unlike VL bus, PCI clearly defines configuration address space to support autodetection and configuration of PCI peripheral cards. Configuration software detects PCI devices on the bus at powerup, typically through PCI BIOS calls, and then accesses the configuration address space of each device to determine its requirements and to assign it unique memory and I/O regions.

A PCI device's configuration address space consists of a base set of 64 registers subdivided into several groups:

- Vendor ID, Device ID, and Revision ID—aid in vendor identification
- Class Code-identifies its basic function (e.g., mass storage, network, video, etc.)
- Command and Status-control how it responds to and performs PCI accesses
- Header Type-describes the format of its configuration header

Figure 1 illustrates a typical PCI-device configuration header.

Unlike other bus architectures, PCI doesn't use terminating resistors at the end of the bus, so the signal wavefront can reflect back down the bus. Weaker output drivers drive the signal line to half the desired logic state. The resulting signal wave is doubled when reflected back, registering a logic high at each device along the way.

This reflective wave-switching technology reduces driver size as well as surge current, but it makes capacitive loading on the bus an important concern. PCI chipsets can drive up to 10 loads, and with components, connectors, and cards each considered one load, this equates to three slots.

INDUSTRIAL CONCERNS

PCI's increased performance capabilities drew the attention of industrial-PC manufacturers. However, passive-backplane architectures are favored over motherboard-based designs in industrial applications mainly due to improved reliability and ease of component replacement.

To address this, the PCI Industrial Computer Manufacturers Group (PICMG) consortium was formed in 1994 by I-Bus, Teknor Industrial Computers, Texas Micro, and Trenton Technology. PICMG defined a passive-backplane PCI standard where a PCI connector was added to the back of

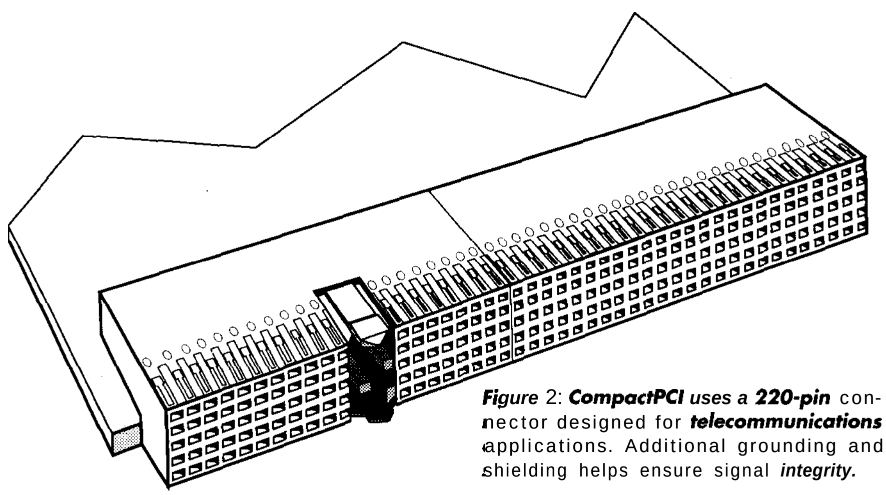


Figure 2: ***CompactPCI uses a 220-pin*** connector designed for ***telecommunications*** applications. Additional grounding and shielding helps ensure signal ***integrity.***

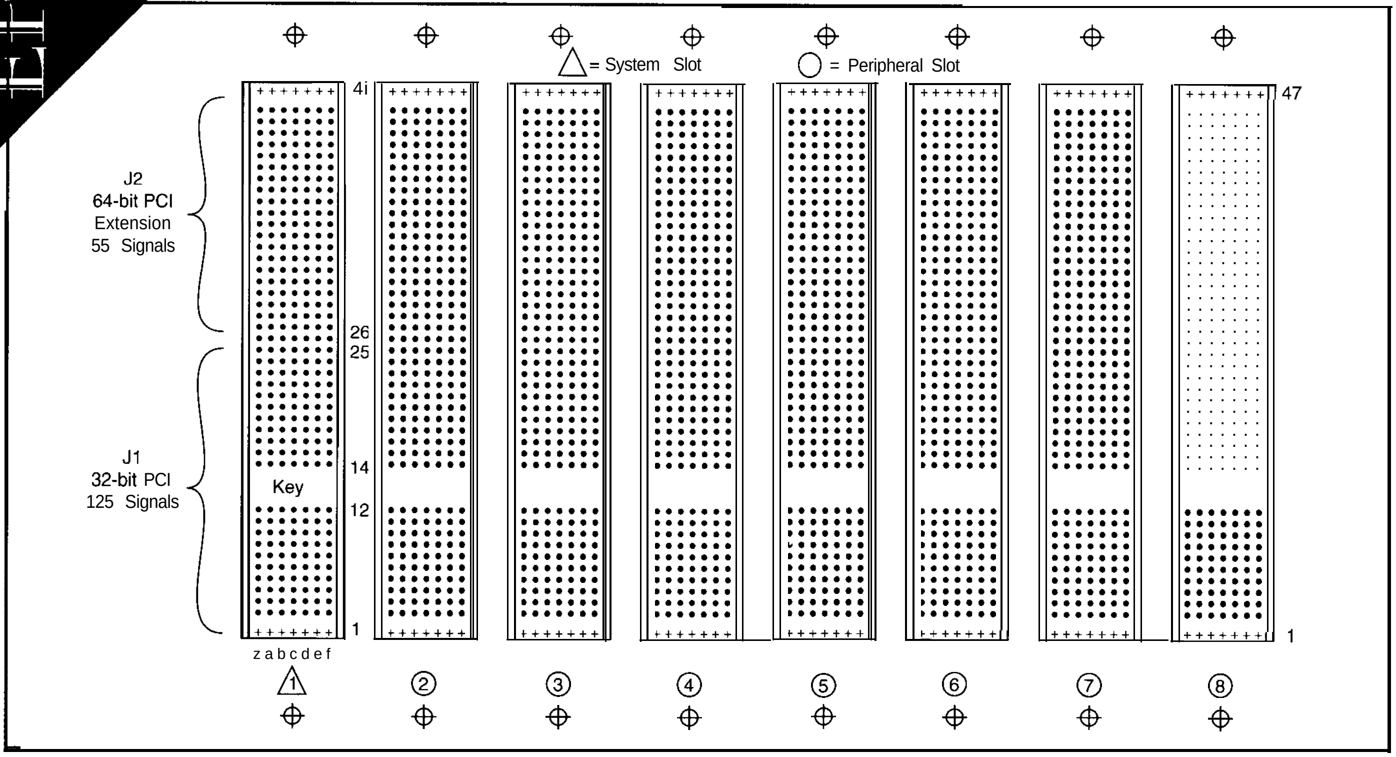


Figure 3: A CompactPCI backplane consists of one system slot located *at* either end and ***up*** to seven peripheral slots. *pin rows* ***1-25(J1)*** support ***32-bit*** *PCI and* the keying mechanism, while rows 2647 ***(J2)*** ***support*** M-bit ***PCI extensions, user-defined signals, and pins reserved for future*** expansion.

passive-backplane ISA boards to maintain backwards compatibility.

Thisapproach solved some maintenance issues, but it didn't solve connector reliability or vibration concerns. Also, industrial PCs typically need to support six or more slotstoaccommodatespecialized I/Ocards and standard peripherals-this was well beyond PCI's four-slot limitation. Repeater cards with PCI bridges support additional slots but at a performance penalty.

VME and Multibus manufacturers also adopted PCI support through a defined mezzanine approach called PCI Mezzanine Card (PMC). These cards mount near the front of VME boards using a pin and socket connection. The I/O signals pass through the front panel.

Unfortunately, PMC modulesdon'tstack, and their form factor limits the number of modules that can be mounted on a VME board to two. linking PMC modules over the VME backplane introduces timing difficulties since VME transfers data at40 MBps whereas PCI's rate is 132 MBps. Also, PCI can't directly support VME read-modify-write transactions.

CompactPCI EMERGES

In 1994, Ziatech Corporation was investigating implementing PCI as a mezzanine standard to its STD 32 Pentium processor boards. The form factor of existing PCI mezzaninecards like the PMC is incompatible with the STD 32 form factor, and as I mentioned, the cards aren't stackable.

From this development effort emerged the concept of a well-defined and open passive-backplane PCI standard that uses commercially available PCI chipsets, supports more than four slots, adopts a Eurocard format and packaging options, and allows for future hot-swap capability of peripheral cards.

After meeting with other industrial computer manufacturers including Teknor, Pro-log, I-Bus, and Ampro, and laterwith PICMG's blessing, Ziatech worked with other companies on the first draft of this new spec. It was presented to PICMG in early 1995 and became CompactPCI.

Now, let's take a close look at how CompactPCI overcame the limitations of both desktop and industrial PCI implementations.

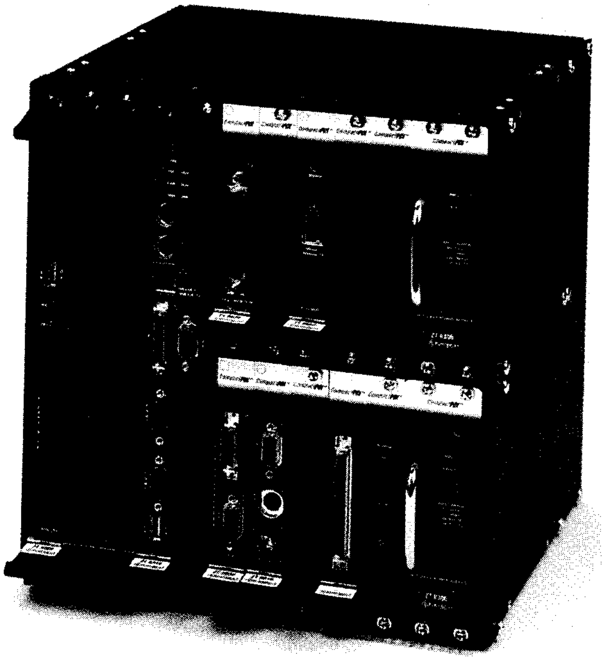


Photo 1: The Ziotech ZT 55 10 6U CompactPCI Pentium board supports up to 14 CompactPCI peripheral cards on two separate CompactPCI buses without bridging.

NUTS AND BOLTS

CompactPCI is an implementation of Intel PCI electrical signals on a Eurocard format with a rugged gas-tight pin-and-socket connector. As shown in Figure 2, the chosen connector is a shielded, 2-mm-pitch, 7-row connector currently manufactured by AMP, Framatone, and ERNI and designed for telecommunication and backplane applications.

Additional features including high ground-to-signal ratio, shielding for EMI/RFI protection, a positive keying mechanism, and a rear pin option for through-the-backplane I/O connection.

CompactPCI defines a 7-column x 47-row pin array divided into two groups. Pins I-25 on one connector support 32-bit PCI and connector keying, while pins 26-47 on a second connector support 64-bit PCI transfers with some pins reserved for future enhancements. The Functional subdivision of these pins is shown in Figure 3.

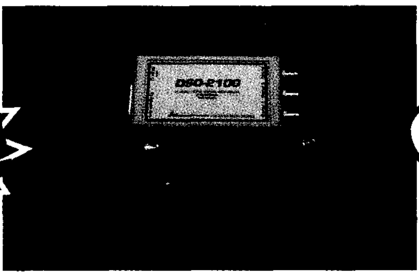
The CompactPCI backplane consists of one system slot that provides arbitration, clock distribution, and reset functions for other adapters. And, it has up to seven peripheral slots that can accommodate simple adapters, intelligent slaves, or PCI-bus masters. Note that the system slot can be located at either end of the backplane.

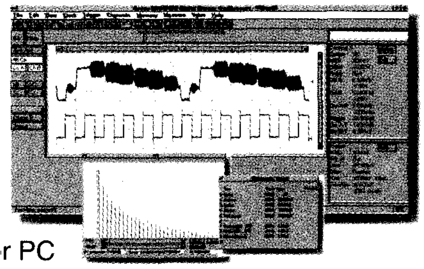
The greatest challenge CompactPCI faced was overcoming PCI's three peripheral-slot limitation.

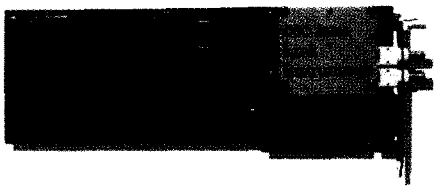
The connector chosen for CompactPCI cards was a key factor towards increasing slot count. Remember, PCI chipsets can drive 10 capacitive loads. Desktop PCI card-edge connectors typically have a capacitive load of 12 pF, whereas the 2-mm hard metric connector chosen for CompactPCI has a capacitive load of 2 pF per pin.

PCI signals were carefully mapped onto the 2-mm connector's pins to take advantage of its extra ground pins and column coupling. And, 10-Ω stub terminating resistors were added to all bused PCI signals on each adapter board to distribute termination and minimize EMI/RFI.

A technical subcommittee headed by Ziatech cooperated with AMP Interconnection Systems to conduct an extensive simulation of Intel PCI signaling through this connector, across a passive backplane, and onto peripheral cards using commercially available PCI chipsets. Lightly, moderately, and fully loaded eight-slot







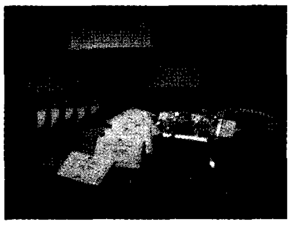
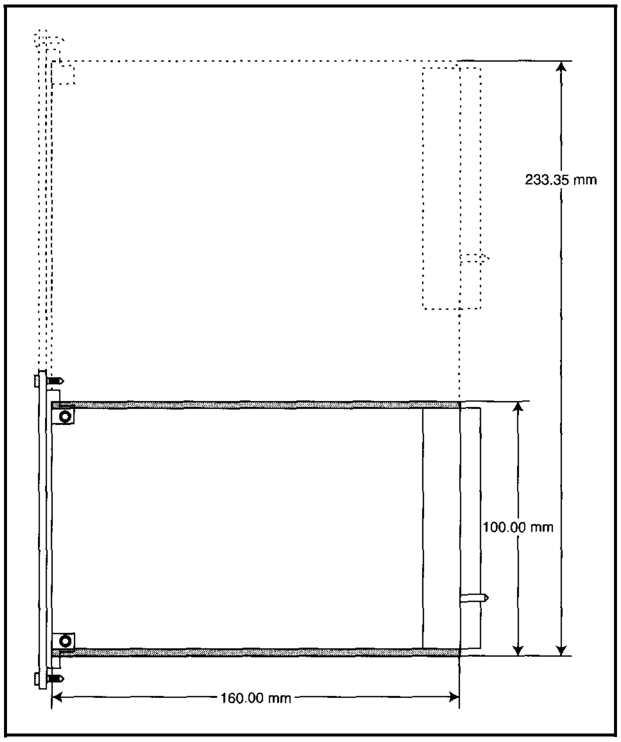


Figure 4: *Both* ***3U and 6U Compact-PCI*** boards have the same ***connec***tor, with 6U boards supporting an additional ***2-mm*** connector for user-defined signals.



backplanes were modeled to determine the best-, nominal-, and worst-case buffer technologies allowed by the PCI spec.

In the fully loaded backplane model using the weakest drivers, all PCI signals'worst-case settling times fell within acceptable limits. In the lightly loaded backplane model using the strongest drivers, the unloaded connectors presented a long unterminated stub that was easily addressed by adding diode termination at the far end of the trace.

Some additional signals defined by CompactPCI to enhance system operation are push-button reset (*PRST), power-supply status (* DEG, * FAL), system slot identification (*SYSEN), and legacy IDE interrupt support. However, these additional signals don't affect the existing PCI signals.

CompactPCI'sEurocard format enables users to take advantage of already existing Eurocard enclosures designed for VME systems. And, manufacturers can design core 3U and optional 6U format boards that will coexist in the same system as shown in Photo 1. Both form factors have the same 2-mm connector, with the 6U format supporting an area for a user-defined second connector as you see in Figure 4.

You may have noted similarities between CompactPCI and VME in both performance and packaging, and you're probably wondering whether the two bus structures will compete or coexist.

So, let's discuss their differences.

CompactPCI OR VME?

Although both bus structures support 3U and 6U formats, VME uses a 2.54-mm connector scheme versus CompactPCI's 2-mm connector. The two buses use entirely different data-transfer methods. And, VME has an asynchronous scheme, while CompactPCI uses a synchronous clock.

As well, VME and CompactPCI perform byte transactions differently. VME uses Big Endian and CompactPCI, Little Endian. Also, VME lacks the plug-and-play capabilities found in PCI systems.

From a performance standpoint, CompactPCI looks like the winner, supporting data transfers up to 264 MBps at 64 bits. By contrast, VME supports 80MBps, using VME64 extensions.

Despite these differences, it's likely that the two buses won'tcompete head to head. Indeed, many major VME manufacturers have joined PICMG, are endorsing CompactPCI, and are announcing products. One VME manufacturer, Force Computers, has developed a 6U CompactPCI card implementing CompactPCI signals on one connector and VME64 extension signals on another.

POTENTIAL APPLICATIONS

So, where does this new standard fit in?

Early inquiries into CompactPCI have come from companies involved in telecommunications applications.

Besides its extremely high bandwidth, CompactPCI offers Eurocard packaging and a connector widely used by the telecommunications industry for both its modularity and reliability. The additional connector on 6U cards can bridge to other buses such as ISA, VME, and STD 32.

Vision applications are also well-suited to CompactPCI. In the past, ISA-based frame-grabber boards were expensive because they needed onboard processors, large amounts of RAM, and special software to capture and process images.

Since a CompactPCI bus-master framegrabber board can grab images and transfer them to the CPU board's main memory at 132 MBps, it no longer needs its own buffer memory. The host CPU can then use less expensive desktop image-processing libraries.

The avionics industry can also take advantage of CompactPCI's performance and small, rugged form factor. Given the space limitationsofaircraft, a 3U CompactPCI system is an excellent choice over a 6U VME. And, it offers better performance.

CompactPCI has taken the performance capabilities of Intel PCI, overcome its slot limitations, and packaged it into a small, rugged form factorwell-suited for industrial applications requiring high performance. It's also an open, well-defined, and increasingly accepted standard among industrial computer manufacturers. EPC

Mike *Baylis* **is an applications *engineer for* Ziatech, a manufacturer *of small* embed-dedcomputers *for STD 32 and CompactPCI Bus. You may reach him at* mike_baylis@ ziatech.com.**

SOURCES
CompactPCI
Ziatech Corp.
1050 Southwood Dr.
San Luis Obispo, CA 93401
(805) 541-0488
Fax: (805) 541-5088

REFERENCES

J. Child. "PCI Becomes Entrenched in Embedded Boards," Computer Design, 35:9, 1996.

J. Child. "PCI and VME Dare to Shore," *Computer Design,* **35:5, 1996.**

R. Davidson and P. Nash. "Early Adopters of New 3U CompactPCI Standard Embrace Its Small Size and Speed," *RTC,* **4:3, 1996.**

PCI Special Interest Group, *PC/* **Fundamentals, Intel, 1993.**

R. Quinnell. "The Mighty Morphin' PCI Bus," *EDN,* **9, 1996.**

T. Shonley and D. Anderson, PC/ System Architecture, 3rd ed., Mindshore, Inc., 1995.

Ziatech, *CompactPCI Short Form* **Specification, 1996.**

Ziotech, *CompactPCI Simulation Overview, 1996.*

IRS

4 10 Very Useful
41 1 Moderately Useful
4 12 Not Useful

"wanna be famous?" WINNER PC/104 Design Contest

PC/104 QUARTER

Frank Jenkins

HomeR

A Home Vacuum-Cleaning Robot

Want a robot that vacuums your house while you're not home to be bothered? Sounds great, right? Frank combines boards **from** *Ampro* and *ComputerBoards* with a three-level *C++* program to bring that fantasy closer to reality.

When I started this project over five years ago, I had three primary goals. I wanted real-world (not lab), hands-on experience with an autonomous mobile robot.

I also wanted to evaluate various types of mechanical parts, electronic systems, sensors, and software algorithms. And, I wanted to promote the concept of autonomous robots as practical devices rather than experimental curiosities.

Engineering projects need well-defined specifications to guide development decisions. I decided that a worthwhile, but seemingly not-too-difficult, robotics task was to perform household vacuum cleaning.

This robot wouldn't vacuum randomly like a swimming-pool cleaner. Rather, it would automatically map out the house, plan vacuuming paths, and proceed to clean on a regular schedule.

HomeR, shown in Photo 1, was designed as a home appliance rather than a robot. As such, it needed to meet the average consumer's expectations for reliability, ease of use, and safety.

It had to operate properly for many months without repair or adjustment. Users shouldn't need backgrounds in mechanics, electronics, or software. The device shouldn't damage (or be damaged by) household objects. And most critically, it had to operate safely around pets and small children.

I wanted a no-compromises robot-one that would come as close as possible to a consumer-ready product while remaining within my modest budget. Rather than buying surplus parts and then building a robot, I designed the robot first and then intended to buy, modify, or build exactly what I needed.

Inevitably, I compromised. Often, the time and cost of using or modifying an off-

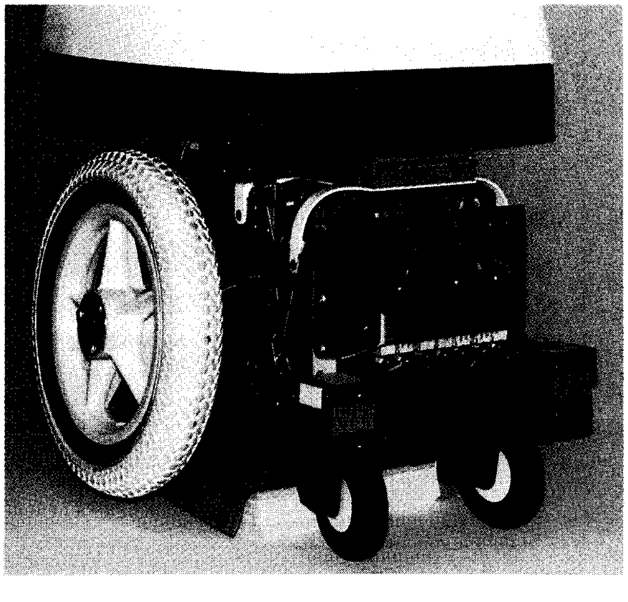


Photo I: In this rear view of ***HomeR*** (lower shell removed), *you can see* ***the*** *sensor ring and lower part of* ***the vacuum*** canister. The electronics bay with its circuit *breakers* and switches is mounted at the rear of the chassis. One of the main drive wheels and both rear castors can also be seen. This picture is about two years old. The latest configuration ***looks slightly different.***

the-shelf product were substantially less than designing and building parts.

HomeR, since its inception, has undergone two major design revisions in both its mechanical and electronic configurations. About the only original parts are the drive motors and the wheels.

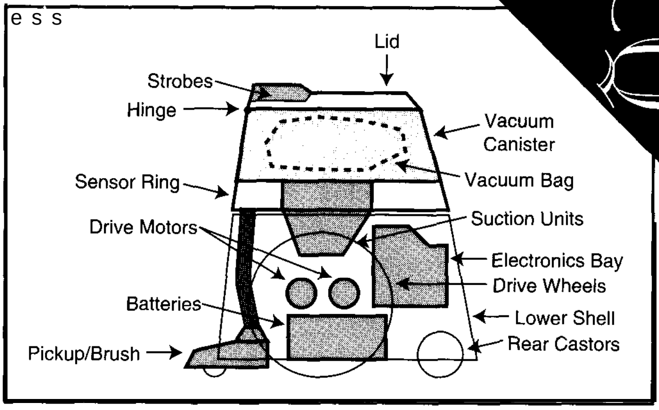


Figure 1: The vacuum cleaner is integmted into the ***overall*** design, not ***just*** added on.

MECHANICAL AND ELECTRICAL

Figure 1 shows HomeR's mechanical layout. The most visually notable feature is the large main-drive wheels, which give good traction on uneven surfaces and a light footprint on carpets. Two smaller costoring wheels are used at the robot's rear corners.

The main chassis was designed to be modular and look like die-molded plastic. It's actually made of thin marine-grade plywood, which approximates the weight, strength, and stiffness of molded plastic and permits easy modification.

The lower shell is constructed as a fiberglass-foam sandwich composite. The middle sensorring is a complex structure made from fiberglass-balsa and-foam sandwiches, plywood, and acrylic plastic.

The vacuum canister is built from plywood, fiberglass, and foam. The lid uses a fiberglass-balsa sandwich base with a custom-molded Kevlar top.

The canister was designed around a standard filter bag from a large Kenmore canister vacuum. The bag is easily replaced by lifting up the front-hinged top lid.

The vacuum system combines two suction motors derived from Black and Decker 12-V car-vats. The retractable, front-mounted pick-up unit (with a rotary brush) is a heavily modified Black and Decker battery-powered floor vacuum. The pick-up/brush unit is electrically retractable (with a mechanical up-lock mechanism) and has an automatic height-adjustment feature.

The two main drive motors are Brevel servo-type gear motors (85: 1 reduction). They nominally use about 2 A of power at 12 V, but during startup, they can consume up to 8 A.

The suction retract/extend motors are two identical Maxon motors with a two-stage 15: **1** belt reduction and a combined peakoperating currentof less than 2 A. The two suction motors are the major power draw, using about 6 A at 12 V with no load and more than 10 A under some conditions. The beater-brush motor draws -2-3 A, depending on the floor surface.

HomeR only has DC brush-type motors. To minimize electrical noise, the motor cases are grounded with 0.1-µF capacitors between the brush connectors and the case.

Three battery systems power the motors and the electronics. A 12-V, 5-Ah battery powers only the suction and brush motors. Another 12-V, S-Ah battery powers the wheel drive motors, and a 9.6-V, 2.5Ah battery powers the electronics.

I chose NiCd battery packs for their high power density, low internal resistance, and deep discharge capability. The batteries are wired independently, using optocouplers and relays for complete isolation from one another. Each battery pack is protected with an internal fuse, and the electrical systems contain an additional seven circuit breakers.

For safety, two nonsynchronized strobe lights mounted inside the canister lid are activated when the wheel motors are enabled. Ribbon cables integrated into the lid hinge provide power and data wires for the lid-mounted electronics.

The electronics and the motor circuits power on via two push buttons, accessible through a movable panel on the **back of** the lower shell. In an emergency, the robot can be powered off **by** simply hitting the panel.

The original projected weight of the robot was 25 lbs., but it's now over 40 lbs.

ELECTRONICS AND SENSORS

Figure 2 shows a basic diagram of HomeR's electronics. The primary processor board is an Ampro LB/386SX-25 with 4 MB of RAM. I designed the basic robot around the LittleBoard form-factor because it was the smallest 'x86 processor board available at the time. (The smaller Core-Module hadn't yet been announced.) A color VGA card piggybacked onto the main board enables a video monitor to be connected.

A 120-Mb 2.5" hard drive connected to the IDE port is used for booting, program storage, and data logging. If necessary, a keyboard and/or an external floppy drive can be connected for data transfer and debugging code. Typically, though, large code is done on another computer and uploaded through the parallel printer port.

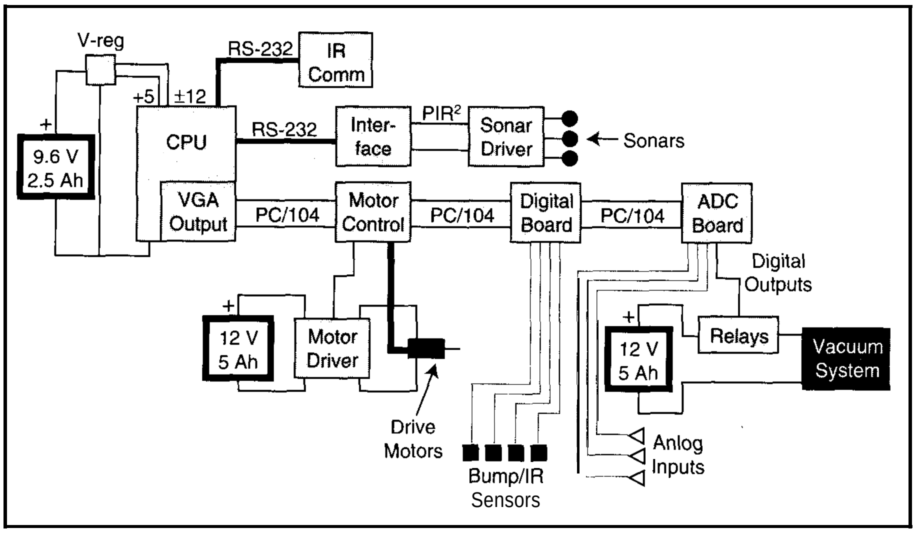
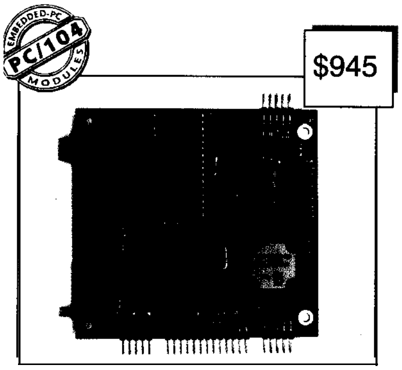
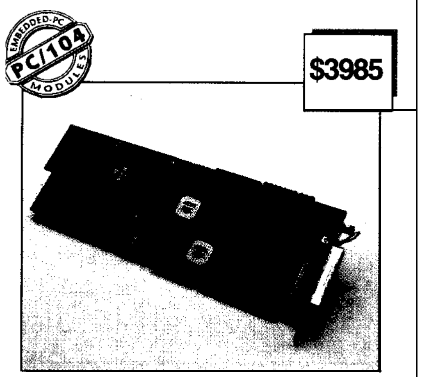


Figure 2: ***Both the PC/I 04 bus and the RS-232 ports are used for data acquisition and control.*** Three battery systems isolate the electronics, motor driver, and vacuum-system ***power.***





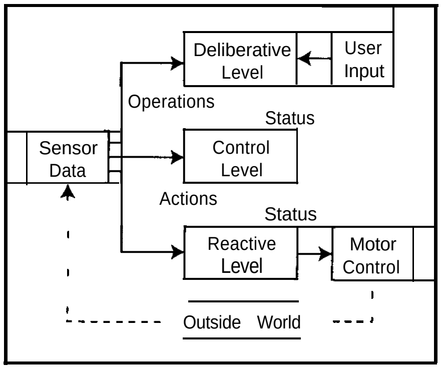


***Figure** 3: The Reactive level performs simple, **low-level control. The** Controlkvelestablishes the priorities for the Reactive level, and the* Deliberative level *does the general planning. All levels can access sensor data.*

The CPU connects through the PC/I 04 bus to a Mesa Electronics 4127 LM629-based servo-motor controller board that's used only for the main drive motors. The motors use Hewlett-Packard HEDS-5500 quadrature optical motor encoders (1440 cpr), which feed back into the controller.

The motor-controller board connects through optocouplers to the motor encoders and a custom-built NMOS H-bridge PWM motor driver with IR2 1 1 OMOS driver chips. Thus, once the CPU determines each motor's trajectory, it sets register values in the 629s, starts the motors, and is then free to perform other tasks.

Also connected on the PC/I 04 bus is a ComputerBoards PC 104-DI48 48-pin digital input board. It's based on 74LS logic chips but uses standard 8255 data-transfer protocols. The digital inputs sense the state of the 22 microswitches and 8 IR sensors.

ComputerBoards' PC 104-DAS08 8-input 12-bit analog board senses motor currents and battery voltages. It also has 4 digital outputs to control the vacuum motors through relaysand the pickup retraction/extension system.

The extender/retractor motors have no encoders. Instead, they use four microswitches to sense whether the pickup/brush is in the up-lock or down-floating position.

HomeR's bumper senses and absorbs collisions. The lower shell is shock-mounted to the chassis and has eight microswitches to sense collisions from any direction.

The sensor ring and vacuum canister are also shock-mounted to the chassis, with eight more microswitches detecting displacement in any direction. The front pick-up/brush unit has two wrap-around bumpers that can activate two more microswitches.

Eight modulated-light infrared sensors are mounted around the sensor ring. The sensor pairs each use an LM567 tone decoder to modulate outgoing light and detect reflected light.

The IR sensing uses a triangular beam path to indicate when an object is within -4". No attempt is made to determine analog distance using the reflected light intensity. The output of each '567 is a binary detect/nodetect sent to the digital input board.

HomeR can have up to 26 sonar distance detectors, although only 8 are in use currently. A total of 24 square Polaroid 7000-series transducers are mounted in the sensor ring. Two round transducers are also mounted to the front of the lower shell.

While the sensors are intended primarily for obstacle detection, they have limited mapping capability. The CPU is connected via RS-232 to a Processing Innovations Robotics Research 68HC1 1 board that communicates to a sonar driving/sensing board over their PIR^2 bus. The 'HC1 1 handles all sonar control. The CPU requests distance readings as needed.

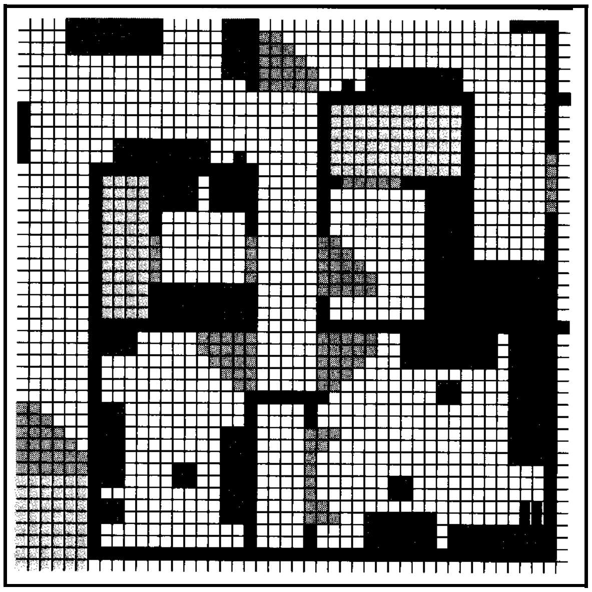


*Figure 4: In **HomeR's two-dimensional world,** floor space is broken up into equal-sized cells. Each cell is assigned a value that indicates whether the floor space is empty, has some obstruction, or is off limits. The shading represents the different cell values.*

Table 1: Some of these Control-level actions might be generated by **the Deliberative-level** *operation* "Position-at-second-door-on-lefi. *"* The normal actions are designed to **perform** the task, assuming no problems. Contingency actions are available if *problems occur.*

Operation
"Position-at-second-door-on-left"
Normal Actions
align-parallel-left-wall
follow-left-wall
move-forward-medium-speed
follow-left-wall
center-to-left-opening
rotate-left-90
Contingency Actions
avoid-obstacle
recover-bumper-hit
search-for-opening-left
end-of-corridor
request-fail-replan

User communication to the robot is via an infrared hand-held remotecontrol keypad. The receiver module and electronics (based on the Forte Infrared Communicator) located inside the canister lid connect to an RS-232 port on the CPU board. Communication from the robot is through a vocabulary of distinctive R2D2-like beeps.

While the motors are driven directly from 12-V batteries, most of the electronics use 5-V power from two 3-A PowerTrends 78HT205HC ISRs. An IPS NMH-0512S 2-W DC-to-DC voltage converter generates the ± 12 V for the analog input board.

Almost all the electronics are mounted in a single bay at the back. It also contains the powerconversion electronics, the circuit breakers, and two small cooling fans. This bay removes as a single unit to facilitate circuit-board development and trouble shooting.

SOFTWARE LANGUAGE AND OS

The software is written in Borland C++ and uses ROV_C robot programming functions. ROV_C is a set of high-level C++ functions(suchosMoveBase,GetSonar, etc.) that give basic function control and monitoring without having to deal with the low-level characteristics of the robot's hardware.

The software currently runs under MS-DOS V.6.2, which is widely available, predictable, and low cost. However, DOS has none of the multitasking capabilities that would suit this real-time application.

Concurrent operation is handled by careful software design and a lot of polling. The lack of multitasking is somewhat alleviated by independent processors for basic motor control and sonar ranging, but the OS is clearly a weak point in the software.

SOFTWARE ARCHITECTURE

HomeR's overall program structure is a variation on what's rapidly becoming the standard robotics AI software architecture. As shown in Figure 3, this multilevel system comprises Deliberative, Control, and Reactive levels.

At the top, the Deliberative level decides the robot's goals. It maintains a map and has access to path planners to determine how to move through the robot's world.

Using the system clock/calendar, the robot's current location and action, and possibly inputs from the user, the software determines the robot's main operating mode. Based on the mode, a set of ordered operations are defined and sent to the Control level.

The Control level mediates between the smart Deliberative and dumb Reactive levels. After receiving the operations list, this level determines the robotaction sequence.

For each action, the program uses certain sensor inputs to create a prioritized list of skills, each with associated parameters. An action's skill list is submitted to the Reactive level. While the action is processed, the Control level monitors both the sensors and the Reactive-level operation to determine when to submit the next action.

The Reactive level simply does what it's told. It continually reevaluates the prioritized skill list. Based on certain sensor inputs, it controls the robot's motors and/or the sound output.

All three levels operate concurrently, but their expected response times vary greatly. The Reactive level performs its I/O responses within milliseconds, so the robot can react rapidly to sensor changes (e.g., a bumper hit). To generate smooth transitions between activities, the Control level requires typical response times of a few tenths of a second.

Most of the Deliberative level's process- and timeconsuming activities (e.g., planning) occur while the robot is inactive. But, if the robot has to deal with an unexpected problem or plan change, it may remain motionless while it "thinks."

In this system, control between levels is downward and only to the next level below.

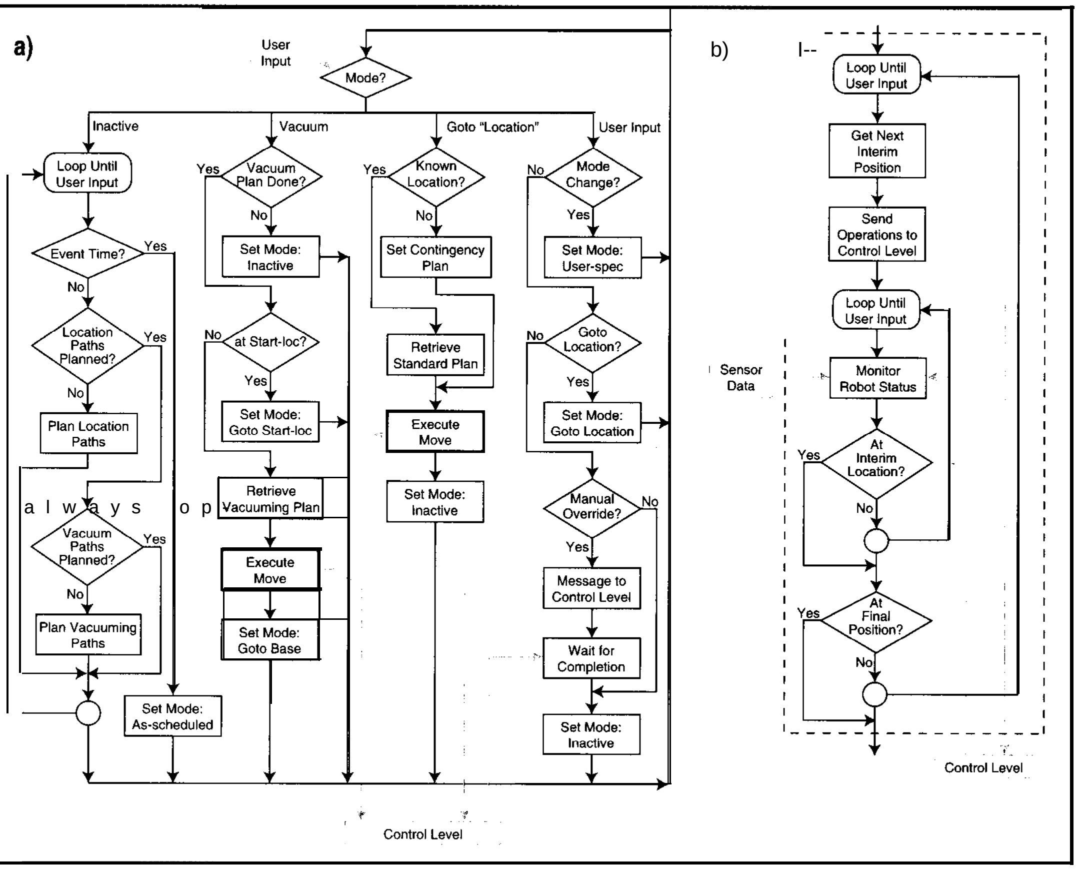


Figure 5a: The ***Deliberative-*** level program always operates in one of several modes. ***The particular*** mode depends on **several factors,** and ***the*** robot can se/f-switch between modes. b: In the execute-move box, the *robot* sends movement operations to the Control level and waits until the *move is* completed. The Control level can signal if ***the*** robot is unable to complete the action.

All levels can access sensor data, but user input is only into the Deliberative level. Also, only the Reactive level controls the robot's actions.

The top two levels each receive status signals from the next lower level. However, as indicated in Figure 3, the control loops are closed primarily through the robot's environment (as interpreted by the sensors). Each level responds to what happens, and not just what's supposed to happen.

A key software component is a map of the robot's working environment. Using the map, the program can search for and evaluate optimum paths for vacuuming or between two points.

This 2D topological map subdivides HomeR's environment into 4" cells. Each cell is assigned a value indicating if the floor space is occupied or free, although other characteristics are also encoded.

In the map section shown in Figure 4, the shading indicates each cell's assigned value. Currently, the map is manually entered and maintained.

As you'll see, each of the three levels performs specific functions, but it is the interaction between the levels (and the environment) that ultimately determines how well HomeR performs.

DELIBERATIVE LEVEL

Figure 5a details the Deliberative-level operation. Within each operating mode, the robot can take several actions, including self-switching to another mode.

In the inactive mode, the robot is typicallywaiting for a scheduled action time or for remote-control input. During this time, the robot can also do path planning or other time-consuming background tasks.

In the vacuum mode, HomeR verifies its starting location and whether it has a path plan. It then executes the movements necessary to vacuum the accessible floor space.

In the goto-location mode, it uses the internal map to navigate between locations. If the start and end positions are standard, the program can call up a predefined plan. Otherwise, it may create a contingency plan.

During theexecute-move processshown in Figure 5b, the Deliberative level continually breaks the robot's path into subpaths, which are then sent to the Control level. While the robot moves, the Deliberative level monitors both the robot's position and the Control-level operation. If a problem occurs (e.g., the robot is "lost" or batteries are low), the Deliberative level may activate an appropriate contingency plan.

The calibrate mode (not shown) uses the remote control and sound output to check whether all external sensors are operating correctly.

Of course, HomeR is ultimately controlled by a human with a remote control. The user can then change the mode or goal location, or manually override to directly control the robot's movement.

Note, however, that regardless of the remote command, the Control level may prevent the robot from following direct orders (e.g., ramming into a wall) if it would damage the robot. While this doesn't follow Dr. Asimov's Second Law of Robotics, I don't want HomeR to damage itself because of an inept human.

Consider this scenario. HomeR is waiting at its home location, which will eventually include an automatic charger. It's programmed to begin vacuuming a certain room at a specified time.

At that time, HomeR wakes up and goes into vacuuming mode. If it's not at the start location, it moves there and reverts to vacuuming mode. When the task is done, it switches to goto-base mode and returns home.

If the robot is in the middleofvacuuming and the userwants itto return to the charger,

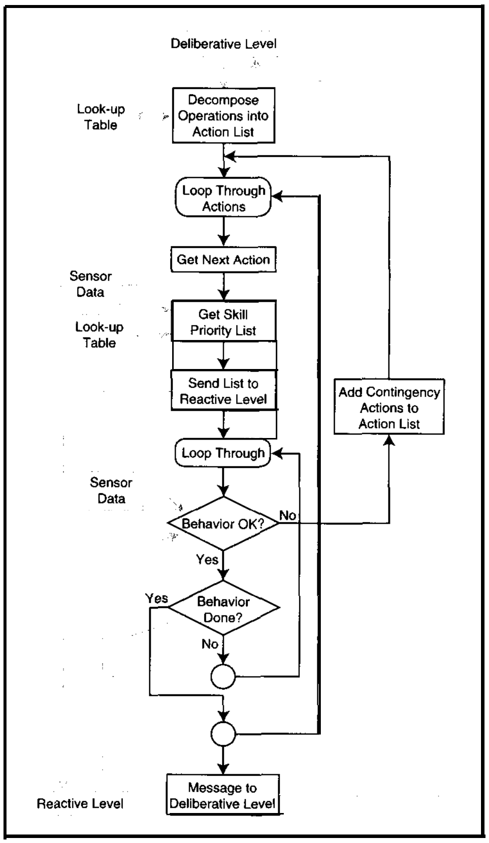


Figure 6: In Control-level program flow, Deliberative-level operations are broken down into a series of ***actions.*** Each action is then converted into a ***prioritized*** skill list that's sent to the Reactive level. If a problem occurs, the Control level may add its own actions to the fist to attempt to resolve the problem.

theusersimplyenters ReturnToBase on the remote control. Since the robot was unexpectedly interrupted, it must stop and custom plan a path back to the charger.

CONTROL LEVEL

The Control level takes an ordered set of operations from the Deliberative level and converts them into prioritized skill sets for the Reactive level. In practice, this task is much easier to state than implement.

As Figure 6 details, the operations sent from the Deliberative level are first decomposed into an action list. Currently, HomeR uses a look-up table approach where a given operation causes a certain set of actions (and contingency actions) to be retrieved.

Some operations map one-to-one to an action (e.g., Immediate-Stop), but in general, each operation generates multiple actions and contingency **actions.** Table 1 shows how a complex operation might expand to a sequence of normal and contingency actions.

As the program loops through each action, it uses another look-up table to generate a priority list of Reactive-level skills. This listcontainsoneor more primary skills for accomplishing the task as well as some contingency skills.

Using a proprietary method, sensor data defines the specific parameters for each skill in the priority list. The finished list is sent to the Reactive level for processing.

While the Reactive-level skills are running, the actual results are continually compared with intended actions. When an action completes, it is removed from the action list, and the code loops to process the next action.

Suppose the Deliberative level wants the robot to go down a hallway and position at the second door on the left. The Control level decomposes the operation as shown in Table 1.

HomeR first aligns itself with the corridor and follows the left wall to the first door, where it switches to deadreckoning until it reacquires the wall. When the robot reaches the second door, it centers with reference to the doorway and turns left.

The program repeatedly checks that the active skill is a primary skill. If a contingency skill is activated or the sensors detect a problem, the Control level may call on an interim contingency action.

It's entirely possible for the robot to request a new contingency action while it's processing an existing contingency. For example, if it sees an obstacle in the hallway, it must first get around it. However, if an unexpected bumper hit occurs while avoiding the obstacle, another contingency action may be necessary before continuing the obstacle avoidance.

Generally, the Control level attempts to complete the requested operations. However, as a last resort (e.g., the robot cannot get around the obstacle), the Control level requests a new plan from the Deliberative level.

REACTIVE LEVEL

This level contains a library of skill functions, each of which evaluates some

Table 2: In the Reactive-level skill priority list, emergency or other unexpected situations are typically given higher priority than the primary program task.

Prioritv	Skill
High	backup-if-bumper-hit (back-dist, bpl, bpr)
↑	stop-if-bumper-hit (bls2, brs3)
│	stoD-if-ir-detect (ir1, ir2)
│	slow-if-ir-detect (speed, ir3, ir4)
│	stop-if-high-motor-current (max-current)
│	stop-if-battery-low (min-charge)
│	maintain-distance-on-left (dmin, dmax)
↓	move-in-direction (azimuth, speed)
Low	return-to-control-level

sensor data and, if necessary, produces an action based on the sensor data and certain numerical parameter values. Table 2 offers some examples.

Some of HomeR's skills are simple and absolute(e.g., emergency-stop). More sophisticated skills involve continuous monitoring of sensor data (e.g., maintain-constant-distance (sonar#, distance)). Also, the same skill function can be used with different parameter values (e.g.,slow-if-ir-detect (x-speed, irl) vs. slow-if-ir-detect (y-speed, i r2)).

Figure 7 illustrates the Reactive-level operation. This level always contains a prioritized list of active skills. During execution, the program sequences through the prioritized list, and each skill function is evaluated against sensor inputs.

If a function evaluates as true, that skill is activated. Once a skill is activated, the program doesn't check lower priority skills. Rather, it returns to the top of the list to reevaluate higher priority skills.

At first glance, this may resemble Control-level flow, but there's an important difference. When a Control-level action is activated, the program waits until the action completes before going to the next action. When a Reactive-level skill function is activated, the program immediately reevaluates the prioritized list. Then, if any higher-priority skill is activated, it overrides the currently active skill.

For example, the program may be executing move-in-direction (azimuth, speed) when the robot gets too close (i.e., less than distanced,,) to a wall on the right side. This behavior causes the higher priority maintain-distance-on-left(dmin, dmax)to activate instead.

Once the distance is again between d_{min} and d_{max}, move activates. If a specified bumper is hit, stop-if-bumper-hit activates and overrides any currently active skill. Table 2 simply illustrates the concept, but a typical list is much longer.

The Reactive-level operation is similar to Rodney Brooks' Subsumption Architecture. But, unlike the standard Subsumption Architecture, behaviors (skills) do not necessarily time out.

Thus, if something activates stop-if-bumper-hit, the robot remains stopped until the bumper is no longer hit (the obstacle moves) or the Control level changes the priority list. This dependency is crucial-the robot cannot usefully operate using the Reactive level alone.

CONTINGENCIES

A major problem for roboticists is balancing goal achievement against contingency handling. But, it's surprisingly easy to get the robot in situations where contingency handling seems to be its primary purpose.

Designing robots to operate in the real world would be much easier if the world was simple and unchanging. But, the number of ways things can go wrong exceeds the normal modes of operation.

Many things conspire to make a robot's life difficult. Obstacles move, wheels slip, sensors break or give false readings, batteries discharge, and doors open and close. It's easy for a robot to get "lost" or exhibit oscillatory behavior.

Increasing the number of sensors may increase the chance of recognizing problems, but it also ups the probability of a failed sensor. Also, the more capable the robot, the more possibilities there are. Since the same failure in different environments may require different responses, the number of potential contingency actions becomes mind-boggling.

EXPERIENCES

The obvious question is: How well does HomeR work? The answer: By experimental standards, it works OK. By home-appliance standards, it has a long way to go.

The vacuum and motioncontrol systems arevirtually problem free. If I control HomeR via the infrared remote, it appears to work very well.

But, when HomeR runs autonomously, it stops for no apparent reason, oscillates between two actions, runs into things its sensors don't see, and generally exhibits unintended behaviors. Despite careful and conservative electronics design, random glitches occasionally occur.

The hard reality is that anytime HomeR runs autonomously, I'm close by with my finger on the remote Stop button.

FUTURE DEVELOPMENTS

For all practical purposes, HomeR's mechanical design is complete. Although the basic chassis design has evolved considerably, a clean-sheet redesign would certainly be lighter, simpler, and more compact.

However, even by the standards of a simple insect, HomeR is virtually deaf, blind, and tactually insensitive. I want to add technologies such as a flux-gate compass and pyroelectric detectors. Another useful addition would be a sound board to

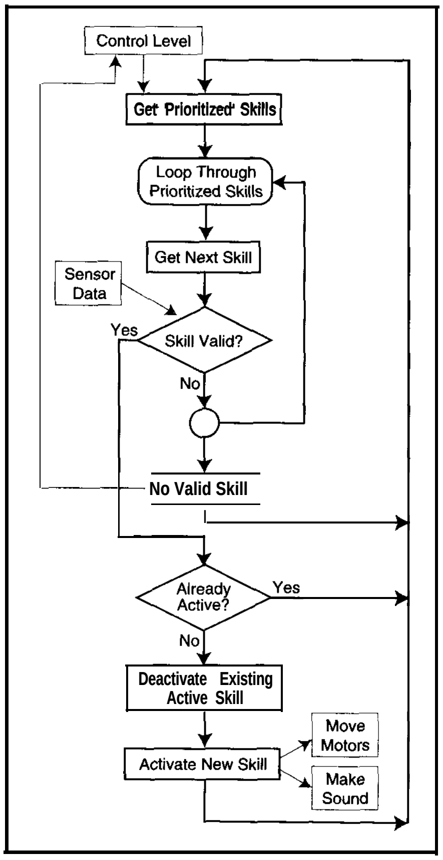


Figure 7: The Reactive-level program sequences through the prioritized skill list provided by the Control level. When a valid skill is *found, it executes* until no longer *valid or until a* higher priority skill becomes valid. *if no valid skill can be found, the Control level is notified to* revise *the* priority *list.*

give HomeR more user-friendly human-like speech.

Software is the **key enabling** technology in autonomous robotics, yet it is by far the least mature. Determining the propertypes, parameters, and sequences of Deliberative-level operations, Control-level actions, and Reactive-level skills is quite difficult. Making sense of sometimes contradictory sensor readings is far from being solved.

From a technical standpoint, a complex project like HomeR is never completed. Rapid advances in hardware technology and constantly **evolving software paradigms seem to make changes obsolete before they can be fully implemented. Furthermore, autonomous mobile robots are so new and rare that there are no generally recognized standards for comparison.**

Though I haven'tfullyachieved my **original goals, I've learned a lot. And with HomeR (and perhaps its successors), I expect to learn even more.** PCQ.EPC

Frank Jenkins **is an electrical** ***engineer with over 15 years'*** **experience in** *high-technology* **consultingandscientificprogramming.** ***/-/is areas of interest and expertise include computers, electronics, artificial intelligence, and robotics. You may reach Frank at*** *fjenkins@ netcom.com.*

SOURCES

LB/386SX-25
Ampro Computers, Inc.
990 Almanor Ave.
Sunnyvale, CA 94086
(408) 522-2 100
Fax: (408) 720-I 305

c++
Borland International
100 Borland Way
Scotts Valley, CA 95066
(408) 43 I-1 000

PC104-DI48 48-pin digital input board,
PC104-DAS08 8-input 12-bit analog board
ComputerBoards, Inc.
125 High St., Ste. 6
Mansfield, MA 02048
(508) 261-I 123
Fax: (508) 261-1094
info@comp-boards.com

HEDS-5500 quadrature optical motor encoders
Hewlett-Packard
Components Group
370 West Trimble Rd.
San Jose, CA 95 13 1
(408) 435-7400

NMH-05 **12S** DC-to-DC converter
International Power Sources
200 Butterfield Dr.
Ashland, MA 0172 1
(508) 88 I-7434

IR2110 MOS driver **IC**
International Rectifier
233 Kansas St.
El Segundo, CA 90245
(3 10) 322-333 1

LM629 servo-motor controller board
Mesa Electronics
4175 lakeside D., Ste. 100
Richmond, CA 94086
(510) 223-9272
Fax: (510) 223-9585

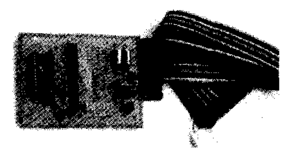
7000-series transducers
Polaroid Corp.
1 19 Windsor St., 2-B
Cambridge, MA 02 139
(6 17) 577-468 1
Fax: (6 17) 577-32 13

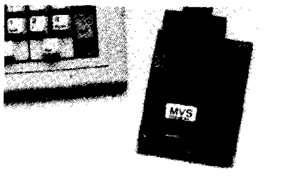
78HT205HC ISRs
PowerTrends
1101 N. Raddant Rd.
Batavia, IL 605 10
(708) 406-0900
Fax: (708) 406-0901

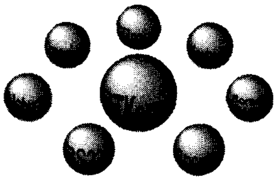
68HC 11, **PIR²** bus
Processing Innovations Robotics Research
10471 S. Brookhurst St.
Anaheim, CA 92804
(714) 535-8161

IRS

413 Very Useful
414 Moderately Useful
415 Not Useful







Applied PCs

Fred Eady

System Problems?

Maybe It's Your BIOS

There's a problem. You know how to solve it. You wanna play God and make a great masterpiece. But, incompatibilities between system components and problem restraints are making your solution formless and void. Fred understands.

Most of you must think that writing for a top-notch magazine like *INK* is a glamorous detail. Well, it is. But sometimes we authors *do* have to work. Like today.

The sun is shining, and the oranges on the backyard tree are ripe. I'm working on my year-round tan, and it's just another sunny day in Circuit Cellar's Florida Room.

To make things even brighter, a big overnight package is sitting on the front porch. Grammar Engine! Smells like a project to me!

While sipping fresh orange juice, I open the box to see what new concoction Scott sent. A PromICE! Grammar's ads always talk about, how pretty this little marvel is. I must admit it'sverycute. It looks well engineered, too.

Ever turn over a puppy or kitten to "see what it was"? Well, I turned over my new PromICE to find that it could emulate 2716 (2 KB) through 29F040 (5 12 KB) devices. There are ports for serial or parallel connections to the host PC and pretty LEDs on the face for Das Blinkinlight types like me.

PromICE is a take-charge tool. Auxiliary signals enable it to reset or interrupt the target and write to its ROM address space. If your application is bit crazy, you can even chain multiple PromICE boxes together.

Digging deeper in the treasure chest, I find a Forth-Systeme '386EX development system complete with the additional serial/DRAM/LED/breadboard card. Am I awake?

I spent the rest of the day reading and absorbing user manuals. The Forth-Systeme '386EX documentation is really skimpy-a mere 0.132". I figured this little board would be either real easy or rock-hard to train. The PromICE user manual stacks in at 0.518". As tech manuals go, it's pretty good reading.

As I studied my new '386EX toy, I found it helpful to have the *Intel'386EX* Embedded *Microprocessor User's* Manual handy for quick cross reference (another 1.46").

DAY TWO

After careful scrutiny of the tiny Forth-Systeme '386EX board, I found that, in addition to the Intel '386EX processor, it contains 256 Kb of zero wait-state SRAM and reset logic in the form of a Maxim MAX707.

Address and bus decoding is provided courtesy of some highly concentrated Lattice ICs. The Forth-Systeme board also brandishes a high-quality 32-pin socket designed to saddle up to 5 12 Kb of boot EPROM or flash.

The '386EX doc mentioned an optional flash file. I was excited. I searched for the 2-or 4-Mb flash-file device, doublechecked the layout diagram, studied the schematic and the '386EX module to the point of identifying most of the miniature surface-mount components, but it just wasn't there.

Having that flash file would have been great for this application, but there was no 56-contact TSOP flash silicon or supporting Lattice part to be found! No flash is a real downer, but at least I have PromICE's power and lots of available RAM.

Before you or I get depressed, realize that Intel's '386EX platform is designed to

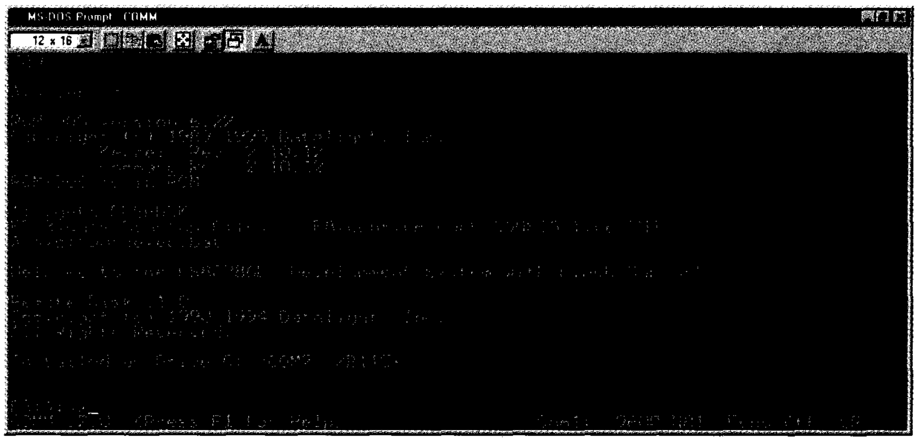


photo **1:** You don't know how I *longed for* this *screen. Note the* Remote Disk installation note.

be fully PC compatible. All the fancy software development packages we use on the big guys can be brought to bear on the development of any Forth-Systeme project—even the ones I cook up!

DAY THREE

The hardware seems pretty straightforward. I have the usual complement of PC-compatible peripherals with the Intel '386EX, ROM and RAM address space, and a couple serial ports.

There's always a no-count, lazy-butt PC around to host. And, if the application or hardware needs to talk to something more stupid, a dumb terminal is within reach.

Now what? I've got this jazzy sports car, but I can't put gas in it. No BIOS. No operating system. No code! Arrgghh!

Once again, I face the engineer's dilemma-buy it or build it?

DAY FOUR

I need a flexible BIOS and OS.

The Forth board lacks a video adapter, so I need to be able to view stuff over the serial port. It doesn't have physical disk-drive capability, so I'll embed some type of bootstrap code and OS in ROM, although it would be great to use "something else's" physical drives.

Wish I had some flash, too, but you can't have everything. If this application puts me in the parallel-port mood, I'd better heat up the soldering iron 'cause there ain't no physical parallel-port interface on the add-on card either.

With all these "ain'ts," maybe **you** think the Forth board is weak. Not so. The entire '386EX part is pinned out to accommodate any possibility, and all standard ISA-bus signals are available. Plus, if you get solder happy, there's plenty of breadboard area.

At this point, I decide to buy. What software do I have on the shop shelf?

AnnaBIOS **looks** good. The documentation is friendly and concise. The code seems very modular with lots of software switches to pull and tug on.

But, I need a particular debugger as well as an 80x86 assembler and C compiler just to get started. I don't see any options or examples for redirecting the console I/O.

The next logical step is to check AnnaBIOS tech support for additional info. I logged on to their Web page and BBS and sent up an SOS. Nada. Looks like a bunch of trial-and-error programming is in my future.

AnnaBIOS is a good package, but it seems better suited for the standard feature-loaded embedded configurations, which in this case, I sorta don't have. I love to code and debug, but not today.

I need a good OS, too. Lo and behold, there's Datalight's ROM-DOS developer's kit. I've used this software before with great success. I remember that a mini-BIOS comes with this package.

Apparently, the Datalight mini-BIOS is basic but will run almost any generic PC-compatible configuration. It's worth a shot.

It also says that the serial port is supported as a console. Perfect! I only need an 80x86 assembler and C compiler (Bill's or Borland's), and there's even an example of how to do it. Plus, it has a ROM disk generator. It's all I need!

DAY FIVE

Up bright and early. Laid in bed all night, and counted bytes roaming on address ranges in the PromICE.

All standard BIOS code resides in a memory area that enables the CPU to reset and hand control to the BIOS routines. Usually, this 64-Kb area starts at F000:0. Datalight's miniBIOS occupies the upper 8 Kb of this space. ROM-DOS slides in at the beginning of the 64-Kb block.

The whole idea is for the BIOS to turn on all the '386EX goodies and pass control to a user routine or, in my case, ROM-DOS. I'm not worried about the application at this point. I want the **BIOS** to tickle the hardware and take ROM-DOS out on the town.

So, I assemble the miniBIOS, and everything seems OK. Being a bithead at heart, I pull up the newly created miniBIOS code in a binary editor to peek at my handiwork.

What's this "Intel '386 Evaluation Board" stuff in the BIOS banner? I'm not using that. Uh-oh! Well, what the heck.

I went ahead and built ROM-DOS and ROMDisk images in hopes that this whole shootin' match would workout. I placed the images into the PromICE at their specified addresses and hit the CPU reset button. Nothing. Nothing on the terminal window. No banner. April fools!

I'm not shy, so I dial upJamie at Datalight tech support. Last time we talked, it was a "pilot error" situation with ROM-DOS. You know, nut loose behind the keyboard. Hopefully, the same would be true now.

She confirmed that my code-module placement was correct and everything should work. Something just wasn't clicking with the Forth board and the miniBIOS.

After some investigation, Jamie informed me that Datalight's engineering staff was currently working on some Forth-Systeme

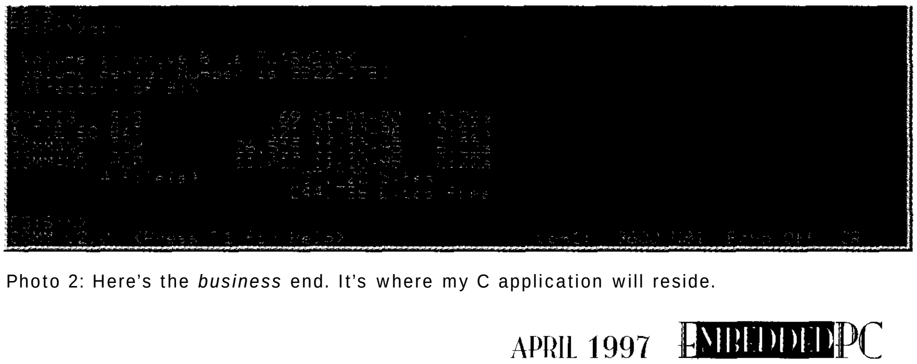


Photo 2: Here's the *business* end. It's where my C application will reside.

miniBIOS code. Great! When would it be available? "In a couple weeks."

It was time for me to do a Bohemian Rhapsody with Freddie and Queen. If you're not familiar with the song, its story line has to do with murder, lost love, and the fantasy of life. In the end, nothing in life really matters to Freddie. (Nothing really matters...hmmm. Not a bad idea about now.)

DAY SIX

If at first you don't succeed, read the manual again. This time, I noticed a BIOS-generating company called General Software claimed to support my Forth-Systeme configuration.

After minor phone tag, I found that General Software's BIOS is incorporated into some Forth-Systeme boards-including mine! Yahoo!

Problem. I couldn'tgetan eval copy. But, the rep told me he'd E-mail the user manual and that the Grammar Engine folks **already** had a workable firmware solution for me.

"Yo, Scott, you got the goods?" Turns out the General Software rep was right.

DAY SEVEN

The Fed Ex delivery person put a rather large, heavy package on the porch. Manuals! That 0.132" '386EX folder just became 3.83" thick-excluding the three high-quality binders!

But, what's this? A 32-pin PROM? Can't be! I'm about manualed out by now, so I take a logical approach to figuring it out as quickly as possible. I "speed" read.

As it turned out, part of the new Forth-Systeme documentation was a reprint of the *Intel '386EX* ***Embedded Microprocessor*** *User's Manual* with the companion '386EX folder doc in German. April fools!

Naturally, itwas the first manual I picked up. You can imagine my panic. At least the technical descriptions and pinouts were in English. Does anybody out there know what an "Entwicklerpaket" is?

Editor's note: Yes, Fred. It's a developer's package.

Another section contains Datalight's ROM-DOS and Card Trick doc. That leaves the General Software embedded **BIOS** and DOS sections. So, I've eliminated (through experience) two binders. There's still that mystery IC and seven diskettes to peruse.

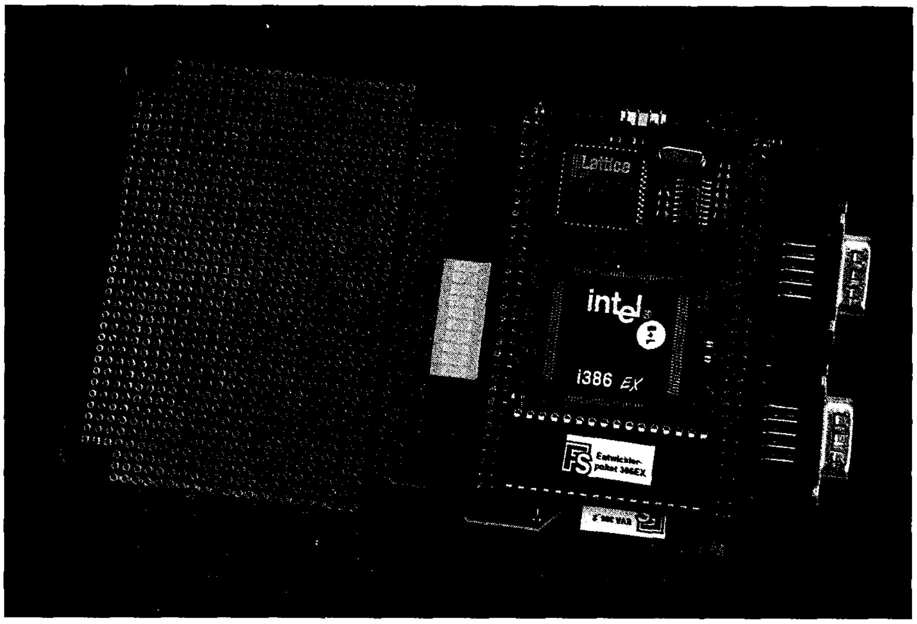


photo 3: The players ***include*** the Forth-Systeme Module (shown here) and ***the PIRM. The*** expansion ***board also*** includes ***Das Blinkinlights.***

IT'S ALIVE!

And on the seventh day, he rested. Whoa! That's another story.

The diskettes were evaluation copies of Paradigm Locate5.0and Debug, Datalight Card Trick and ROM-DOS 6.22, and General Software's Embedded BIOS 3.0 and Embedded DOS 6-XL. Of these, I chose the DOS/Card Trick suite and General Software's BIOS.

To generate a working BIOS, simply read the excellent General Software documentation and configure two header files. The BIOS programmer can concentrate on twoconfigurationfiles,OPTIONS. INCand CONFIG. I NC, instead of changing parms in over 50 BIOS source files.

There are over 170 configuration options that cover a wide variety of system BIOS designs. I could write a couple articles just on building up a BIOS for this project.

For now, in addition to the normal BIOS stuff, the serial console redirection and remote disk-access features make the General Software and Datalight product suite the ticket for my application.

Generating ROM-DOS and ROMDisk is an automated process. Datalight includes a Build program that walks you through assembling a working ROM-DOS/ROM-Disk image. No worries here.

What's left? There's a single sheet of what seem to be instructions with that mystery IC. Hey, I've learned. When in doubt, read.

Connect a dumb terminal to serial port 1. Instead, I used a Windows 95 DOS window running the COMM program that comes with Datalight's Developer Kit. This way, I can show you what happens. Done.

Connect COM2 on the '386EX board to a serial port on the host PC. Instead of a desktop PC, Teknor's VIPer embedded PC

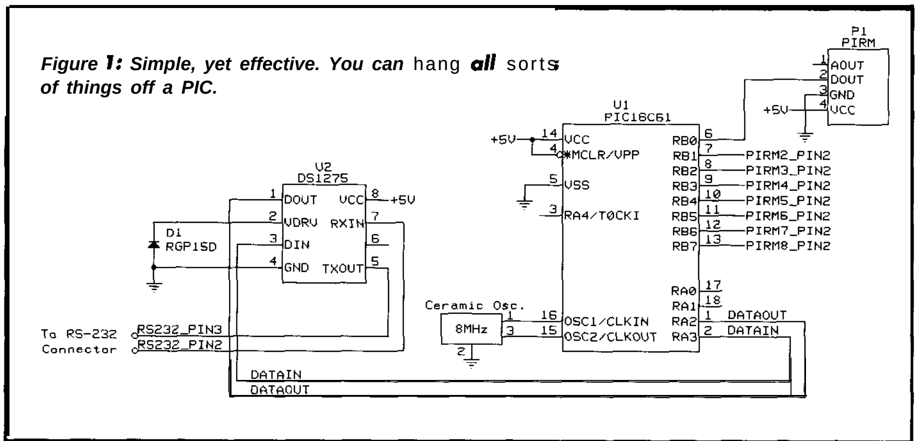


Figure 1: Simple, yet effective. You can hang ***all*** sorts ***of things off a PIC.***

with a full complement of disk peripherals is the host. After all, I'm writing for the *EPC* section. Done.

Run the REMSERV program included on the binary image diskette. Of course, before you do all this, install the included PROM in the EPROM socket on the Forth-Systeme module. (You know, this process brings another song to mind, "Won't get fooled again" by The WHO.)

In goes the EPROM. On goes power. A General Software BIOS banner and ROM-DOS is initializing. Photo 1 appears. If Scott were here, I'd kiss him. Well, maybe not.

Notice the FS:A:\> prompt. This is the ROMDisk image. Photo 2 is a directory listing of the Flash-Disk designated as drive B. Yea, flash at last! Also, there's a drive C that equates to the VIPer's physical hard drive.

REMSERV provides a way to redirect disk-access requests through the VIPer and Forth-Systeme serial ports. I'll use the VIPer C drive to load the Flash-Disk B drive with my application. Thanks to Datalight's Card Trick, I can emulate that flash I was yearning for with some of the 29F040.

Finally, I'm where I want to be. A working General Software BIOS was generated for the '386EX module. A Datalight em bedded DOS was generated and placed in ROM address space. And, some tricky Datalight disk-access code made it all useful.

Just for grins, I copied the PROM contents into a binary image and looked at the whole landscape using PromICE. Beautiful.

PUT IT TO WORK

Yeah, yeah. The future's so bright, I gotta wear shades. Let's get on with it.

Some time ago, I bought some modules that replace standard AC wall switches with an automatic proximity switch and timer. When motion is detected, the light associated with that wall switch activates and automatically turns off after a certain time period. Well, I found some of those sensors in their raw form, and I'm gonna put them to work.

The sensor is an AMP PIRM (Passive Infrared Module) that comes as a complete package including detector, Fresnel lens, and electronics. The detector and lens provide a field of 34 horizontal and 30 lookdown beams. Output is an active-high digital signal activated when motion is detected within 20'.

For you analog heads, there's also an analog output offset at 2.5 V. The PIRM uses a standard +5-V TTL supply drawing 2 mA max, so a battery supply can be used if your application requires it. Photo 3 shows the Forth-Systeme modules.

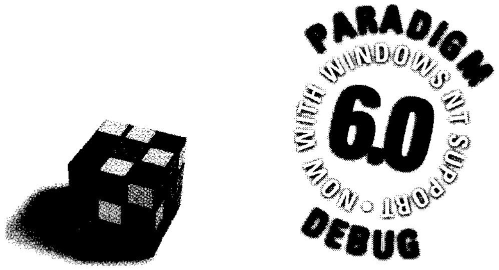
Lots of applications come to mind. Is someone in a secure area of an office or lab? Are they playing in the highdollar parts bin? Is light necessary in this room if it's empty? Do I want my coffee pot to turn on when I enter the kitchen?

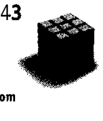
The list is endless, so I'll apply the technology, and you provide the reason.

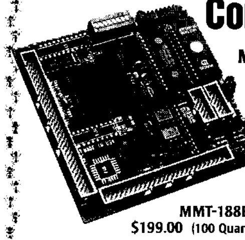
Remember, there's no physical parallel port on my '386EX board. And, I'm not going to add complexity and parts when I can accomplish the task with resident hardware and software.

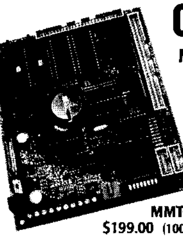
With a working DOS, I can write a driver in C or any other high-level language to interface with the PIRM controller. This controller is a standard micro programmed to interface eight PIRMs to the second '386EX serial port.

Here's how it fits together. The C program monitors the PIRM controller via the









Listing 1: As you see, C is effective in an embedded environment.

```
i/include   <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <bios.h>
l/include  <dos.h>
unsigned char serial-in,
              mask-value,
              sensor-bit,
              sensor_data_in;
unsigned int  com_port,
              sensor-flag,
              status,
              i;

void          get_byte(void);
#define       corn-setup  (_COM_9600| _COM_CHR8| _COM_STOP1
                |_COM_NOPARITY)

#define       COM1
#define       COM2

get_byte()
{
  do{
    if (kbhit())
      exit(0);
      status=inp(0x2FD);
  }
  while ((status & data-ready) != 0x01);
    serial_in=inp(0x2F8);
}

void main0
{
  _bios_serialcom(_COM_INIT,COM2,com_setup);
  com_port=1; //COM2
  //com_port=0; //COM1
  do {
    get_byte();
    for (i=1;i<9;++i)
      printf("                    \n");
    sensor_flag=0;
    mask_value=1;
    for (i=1;i<9;++i){
      sensor_bit=serial_in & mask-value;
      if (sensor-bit != 0){
        ++sensor_flag;
        printf("SENSOR %d IS ACTIVE..\n",i);
      }
      mask_value=mask_value << 1;
    }
    if (sensor_flag==0)
      printf("NO SENSORS ARE ACTIVE..\n");
 }
 while (!kbhit());
 printf("exit\n");
 exit(0);
}
```

'386EX COM2 serial port. The serial data received by the program is an eight-bit word with each bit corresponding to a sensor. If the bit is 1, the sensor is on.

This program can be embellished to include things like room numbers or bin locations per sensor. My program simply tells the user which sensors are active. Since the video is routed to a serial port, these messages appear on the terminal attached to the '386EX COM1 serial port.

The compiled C program is placed on the VIPer C drive. Just make a DOS copy of thefilefrom this drive to the '386EX Bdrive. From there, kick off the C program as you would on a desktop PC. Listing 1 tells all.

To most of us, the PIRM controller is an old friend-a PIC. The type doesn't really matter as long as there are enough I/O pins to accommodate your sensors.

The PIC16C61 I chose for my PIRM controller constantly polls the I/O port attached to the sensor array.

When a sensor detects motion, that particular I/O pin goes high and the PIRM controller sends an eight-bit data packet to the C program. It decodes the bit pattern and displays a user-defined message.

The data packet is an image of the I/O port pins. Need more sensors? Add PIC I/O ports and send more data packets. Figure 1 shows the PIRM-controller schematic.

PAY DIRT

By using some really sweet DOS and BIOS code modules from General Software and Datalight, along with the Forth-Systeme '386EX module and one smart Microchip IC, I've transformed my little '386EX evaluation kit into a useful device.

I could have written some nonDOS-compatible C code and embedded it or done it without any BIOS code. But why?

The headache of embedding applications is erased by standard DOS-compatible tools and code. The Grammar Engine PromICE let me load and test different DOS and BIOS images with minimal effort. Because the Datalight and General Software modules composing the final images are pretested to a point, I didn't need to debug.

This application is just a beginning. The outboard PIRM controller could just as easily be an array of serially interfaced ADCs or another '386EX module running a slave application. By using DOS-compatible hardware and embedding DOS-compatible firmware, I've again proven it doesn't have to be complicated to be embedded. APC.EPC

My thanks to Scott Copeland at Grammar Engine for servicing my spurious interrupts during production of this article. Also, thanks again to Jamie Ferrier at Datalight for her sympathetic ears.

Fred Eady has over 19 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. Fred may be reached at edtp@ddi.digital.net.

SOURCES

AnnaBIOS
AnnaBooks
1 1838 Bernardo Plaza Ct.
San Diego, CA 92 128-2414
(6 19) 673-0870
Fax: (619) 673-I 432
www.annabooks.com

PIRM Module PIRM1 80-200
B.G. Micro, Inc.
P.O. Box 280298
Dallas, TX 75228
(972) 2715546
Fax: (972) 271.2462
www.bgmicro.com

ROM-DOS V.6.22, ROM-Disk
Datalight
188 10 59th Ave. NE
Arlington, WA 98223
(360) 435.8086
Fax: (360) 435.0253
www.datalight.com

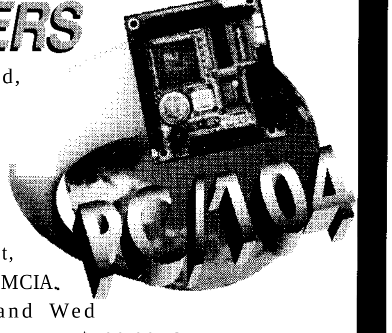
Embedded BIOS adaptation kit
General Software
320-I 08th Ave. NE, Ste. 400
Bellevue, WA 98004
(206) 4545755
Fax: (206) 4545744
www.gensw.com/general

PromICE, Forth-Systeme '386EX
Grammar Engine, Inc.
92 1 Eastwind Dr., Ste. 122
Westerville, OH 4308 1
(614) 899.7878
Fax: (614) 899.7888
www.gei.com

VIPerB06
Teknor Microsystems, Inc.
7900 Glades Rd.
BocaRaton, FL 33434
(561) 883.6191
Fox: (561) 883-669
www.teknor.com

IRS

4 16 Very Useful
4 17 Moderately Useful
418 Not Useful



DEPARTMENTS

62 MicroSeries

70 From the Bench

76 Silicon Update

MICRO SERIES

Joe DiBartolomeo

Standards for Electromagnetic Compliance Testing

Immunity and Susceptibility

3 4

Shipping to Europe? Unlike the U.S., the EC requires testing for EMI immunity. You'd better know EC standards for static discharge, fast transients, and radiated EMI fields.

In Part 2, I discussed the radiated- and conducted-emissions tests mandated for electronic equipment by the FCC and the European Community (EC). By limiting the levels of radiated and conducted emissions, these tests set up an electromagnetic environment for the coexistence of electronic equipment.

It implies electronic equipment must be able to withstand a certain amount of EMI caused by other electronic equipment and natural phenomena. This ability is referred to as immunity or susceptibility.

In this issue, I look at immunity standards. Unlike radiated and conducted emissions where the FCC and the EC are in agreement, there's no similarity at all when it comes to immunity tests.

The FCC requires no immunity testing, while the EC requires it extensively. Since the FCC does not mandate immunity testing for class B digital devices, it's up to the manufacturers, when shipping within or into the U.S., to ensure their products can operate properly in their intended electromagnetic environment.

This approach has been fairly successful with very little government red tape. Most manufacturers design their products to meet, and in many cases

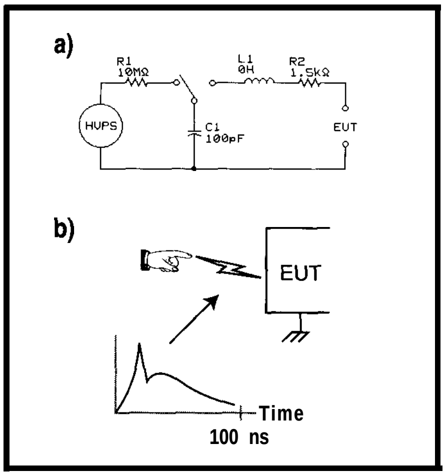


Figure l--This *schematic represents the human body for ESD events. The* **charge on the human body is** *acquired slow/y through the* ***10-MΩ resistor.*** *However,* **charge from** ***the*** **human** *body is* **released quickly** *through the* ***1.5-kΩ*** *resistor (a). The charge transfer between a person and electronic equipment fakes p/ace in that short time prior* **to** *physical contact. The current waveform of the charge* ***transfer*** *is shown here (b).*

exceed, the immunity requirements set up by the FCC emissions standards.

Competitive pressures ensure that most manufacturers do not cut corners on immunity. However, there are always some manufacturers that cut corners to reduce costs or, more commonly, do inadequate testing.

As the electromagnetic spectrum becomes more crowded, the immunity issue will be much more important.

If products had a higher level of immunity to radiated and conducted emissions, many EM1 problems could be avoided. This is one of EC's main arguments for immunity testing.

For a product to be sold into the EC, it must meet both an emissions and an immunity standard. Each immunity standard calls other technical standards to outline the actual tests.

The technical standards, the IEC 1000-4 series of tests, are quite extensive. 1'11 discuss the three most common-electrostatic discharge, electrical fast transients/ bursts, and radiated electromagnetic fields.

Each IEC 1000-4 series standard specifies the test method and gives several severity levels along with several performance criteria. It's up to the calling standard to specify which level of testing is applied. This allows the IEC series to be called by a wide range of immunity standards.

ELECTROSTATIC DISCHARGE

Electrostatic discharge (ESD) **is** by far the most common transient event that electronic equipment is subjected to.

When a person carrying a static charge comes in contact with electronic equipment, the charge transfer and potential equalization can cause currents in the tens of amperes. These currents and their thermal effects cause traces and components in the electronic equipment to fail.

To better understand, test, and design for ESD, the Human Body Model (HBM) was developed. The HBM simulates the discharge event that occurs when a person charged with positive or negative potential comes in contact with electronic equipment. Regulators use the HBM to determine ESD test procedures.

Figure **la** shows the equivalent schematic of the human body for ESD purposes. The model shows that a large static potential is built up, represented by the high-voltage power supply (HVPS).

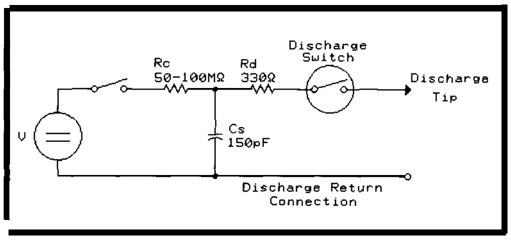
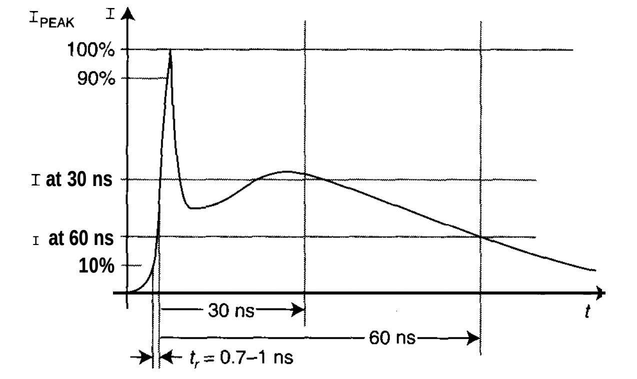


Figure 2—*The* *ESD gun used for testing simulates the charge transfer that occurs between a human and a conducting object.*

The 10-MΩ resistor charges the capacitor C 1, which represents the capacitance of the human body; S1 initiates the charge transfer. In the left-hand position, S1 represent a static charge. When S1 is switched to the right-hand position, the charge is transferred to the EUT.

As you see in Figure lb, the charge is not transferred when the person physically contacts the electronic equipment. The charge arcs from the person to the equipment in that very short time period when the person is close enough to, but not actually touching the equipment.

We've all walked across carpet and touched a metal door knob. Just prior to making physical contact, we get the familiar shock. But, after we make physical contact, we feel nothing. Keep this in mind since it has bearing on the ESD test procedures.



Severity Level	Voltage (kV)	First Peak (A)	Rise Time (ns)	Current @30ns (A)	Current @60ns (A)
1				4	
2	2	7.5 15	0.7-l 0.7-l	8	2
3	4				4
4	6 8	22.5 30	0.7-l 0.7-l	12 16	6 8

Figure 3—In *a current waveform produced by an* ***ESD*** *gun, the charge is transferred in a very short period of time (i.e.,* ***<100 ns).*** *The large, extreme/y fast spike at the* ***start*** *of the waveform simulates the initial charge transfer or spark that occurs when a charged human body comes in contact with a conducting object. The ESD-gun* **waveform parameters are established by the** ***ESD*** **test standard IEC 1000-4-Z.**

ESD GUN

An ESD gun tests equipment for ESD and must be capable of reproducing the waveform in Figure lb. Figure 2 shows a simplified schematic of an ESD gun.

The ESD gun is quite simple. The storage capacitor Cs is charged through resistor Rc and discharged to the EUT via Rd. The switch is usually a vacuum relay and is manually operated.

Figure 3 is a detailed drawing of the ESD current waveform. Note that the initial pulse is extremely short (<1 ns), and the total waveform time is in the order of 100 ns. The current I_{peak} in Figure 3 varies de-

Severity Level	Contact Discharge Test Voltage (kV)	Air Discharge Test Voltage (kV)
1	2	2
2	4	4
3	6	8
4	8	15
X	Special	Special

Table I--These are *the ESD severity levels that a calling standard can dictate. Contact discharge is applied to conducting surfaces with a sharp fip, while air discharge is applied to nonconducting surfaces with a rounded tip.*

pending on the severity level. These waveform parameters are given in the table in that figure.

The ESD gun has two tips-a sharp tip for contact discharge and a round one for air discharge. The gun must provide the waveform in Figure 3 at voltage levels up to 15 kV in both positive and negative polarity. The ESD gun has a ground strap that's connected to the ground plane to provide a return.

ESD TESTS

The standard for ESD testing and measurement techniques is IEC 1000-4-2. The ESD test is fairly simple. It consists of applying an ESD to all points on the EUT that are accessible to an operator during normal operation. The test levels are given in Table 1.

Contact or direct discharge is the preferred method of testing. Even through air, discharge is more representative of the actual ESD event. Air discharge is used where direct discharge cannot be applied (e.g., to nonconducting surfaces). To simulate an ESD event close but not on electronic equipment, indirect discharge is applied.

Figure 4 shows the test setup. The EUT is placed on a nonconducting table 80 cm above a ground plane. A horizontal coupling plane is put on the table but insulated from the EUT.

A vertical coupling plane is also used, but it must be movable. The ground plane on the floor is connected to the mains ground, and the coupling planes are connected to the ground plane via bleeder resistors.

INDIRECT DISCHARGE

When an ESD occurs in close proximity to electronic equipment, the radiated and conducted emissions produced can affect the EUT. To simulate this, the ESD gun is placed in contact with the coupling planes and then discharged.

A minimum of four contact points for both the vertical and horizontal coupling planes ensures that full coverage can be effected. In the case of the horizontal coupling plane, it's a matter of choosing four points-one on each of the four sides of the EUT, but no further than 0.1 m from the EUT.

For the vertical coupling plane, the discharge point is the center of the plane. This plane is placed no further than 0.1 m from the EUT and the discharge is applied. All four sides must be illuminated.

Normally 10 discharges per point with a l-s minimum between discharges is applied. Test levels start at the lowest voltage level and work up, testing both polarities.

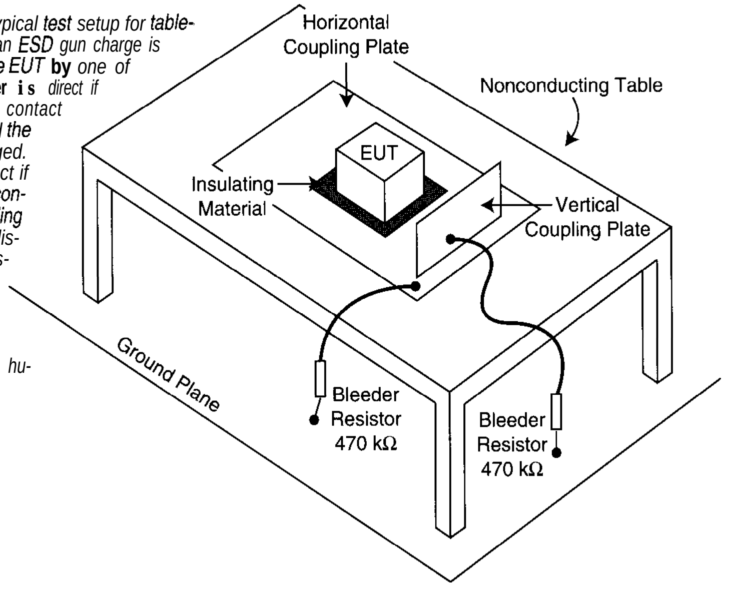


Figure 4-/n *a typical test setup for table-fop ESD testing, an ESD gun charge is transferred to the EUT* **by** *one of three ways.* **Transfer is** *direct if the ESD gun* **is in** *contact with the EUT and the gun is discharged. Transfer is indirect if the ESD gun is in contact with the coupling plate and then discharged. Air discharge involves moving the ESD gun quickly toward the EUT, simulating normal human contact.*

The indirect-discharge test is almost always performed first, since it is the least severe. Failures are normally nonfatal, consisting mostly of the equipment hanging or resetting.

Such failures contrast with the direct- and air-discharge portions of the ESD test, where permanent damage can occur. Therefore, until the equipment passes the indirect discharge test, it's unwise to try the other two parts of the test.

DIRECT DISCHARGE

Although less representative of an actual ESD event, the direct-contact test is preferred since its results are repeatable. All conducting and operating parts of the EUT that a user can contact under normal conditions must be tested.

The ESD gun is brought into contact with the EUT, and then the ESD gun is triggered. Normally 10 discharges are applied with a minimum of 1 s between ESD triggering. Again, the levels start out low and work their way up, testing both polarities.

AIR DISCHARGE

Where direct discharge cannot be used (i.e., with nonconducting surfaces), air discharge is used. The ESD gun is fitted with the rounded tip. With the trigger closed, the ESD gun is moved towards the EUT as quickly as possible and contact is made with the EUT.

This technique closely simulates what happens when a person comes in contact with electronic equipment.

Unfortunately, with this test, it is difficult to get repeatable results. The speed and angle of approach are diffi-

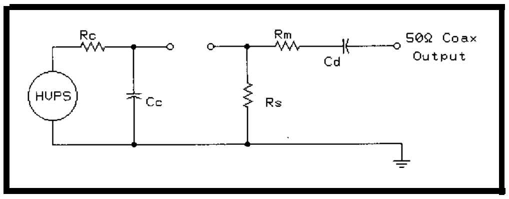
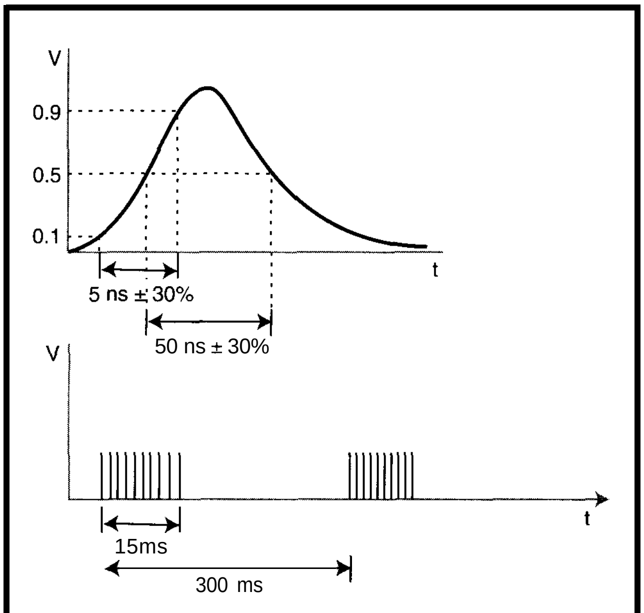


Figure 5—*The EFT/B generator is used to test the immunity of electronic equipment to switching-type transients as mandated by IEC 1000-4-4.*



	Power Supply Ports		I/O Ports	
Severity Level	Voltage Peak (kV)	Repetition Rate (kHz)	Voltage Peak (kV)	Repetition Rate (kHz)
1				5
2	015	5 5	0.25 0.5	5
3	2	5	1	5
4	4	2.5	2	5
X	Special		Special	

Figure 6—*The EFT/B generafor produces a train of pulses which last 15 ms during a 300-ms time frame. The individual pulses are we//-defined. The voltage /eve/ of the applied voltage* V depends on *the severity/eve/* applied. *The calling* standardspecifies *severity levels for* various *test* voltage *levels* and repetition *rates.*

cult to reproduce, not to mention the fact that the equipment tends to get damaged.

As with the direct and indirect tests, normally 10 discharges per point are made. They start with the lowest voltage level and work up, testing both polarities.

The ESD test can be quite time-consuming. And, since test labs charge by the hour, this is one test you do not want to repeat.

FAST TRANSIENT/BURST

The standard for electrical fast transient/burst (EFT/B) testing and measurement techniques is IEC 1000-4-4. The EFT/B test checks the immunity of electronic equipment to switching-transient-type interference. Common sources are the switching of inductive loads and relay bounce. The test is applied to the EUT power and I/O [data, control, and signal) ports.

A simplified circuit diagram of the test generator is shown in Figure 5. The test signal consists of a series of individual pulses applied in bursts.

Figure 6 shows the waveform of the individual pulse and the complete test signal. Individual pulses are quite short (i.e., -50 ns) with burst durations of 15 ms and burst periods of 300 ms.

Individual pulses have several repetition rates depending on the test level. The table in Figure 6 gives these as well as the voltage peaks.

The test signal is directly or indirectly coupled into the EUT's lines. Normally, for power ports, the signal is coupled in via a coupling network, and for I/O lines, a coupling clamp is used. Figure 7 shows simplified versions of both coupling techniques.

For a given piece of test equipment, there could be several different test setups, so it's not possible to show a typical setup. Once a test setup is established, the application of the test is quite simple. Each power and I/O port is subjected to the test signal of Figure 6.

RADIATED EM FIELDS

The standard for EUT immunity to radiated electromagnetic fields (REF) testing and measurement techniques is IEC 1000-4-3. The REF was designed to test the EUT immunity to radiated electromagnetic fields.

The test is quite simple. The EUT is placed on a nonconducting table in a controlled electromagnetic environment (e.g., an anechoic chamber). A transmitting antenna is placed at a distance from the EUT. The antenna radiates a field such that at the EUT a known field strength exists.

Figure 8 shows the test setup. The antenna is driven by a sine wave in the frequency range of 80-1000 MHz, which is amplitude modulated by a 1-kHz sine wave to an 80% level. The frequency is swept at a rate no faster than 1.5 x 10^{-3} decades per second or stepped at intervals of 1.01 times the previous step.

EVALUATING TEST RESULTS

Due to the wide variety of electronic equipment subjected to immunity testing, result evaluation is done on an individual basis by comparing test results to normal operating conditions.

Normal conditions are usually established by the manufacturer prior to testing. They must reflect the normal operating conditions implied in the manufacturer's equipment literature.

The IEC 1000-4 series classifies the test results as:

- normal performance within specified limits
- temporary degradation or loss of function or performance which is self-recoverable
- temporary degradation or loss of function or performance which needs operator intervention or system reset

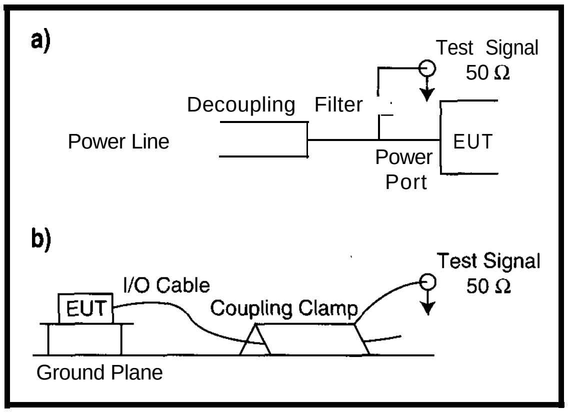
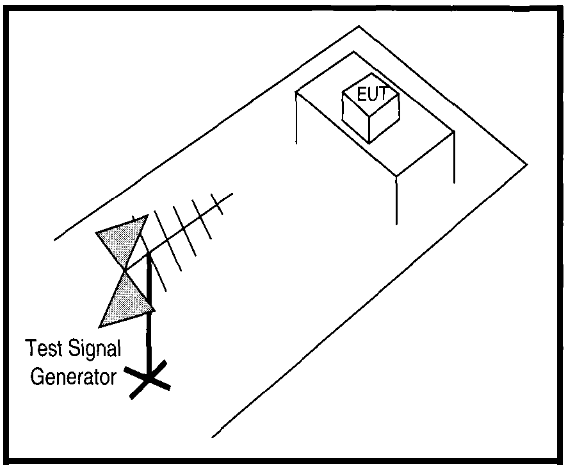


Figure 7a—*Coupling* of *the* fast transient *burst signal to either AC or DC power pork is normal/y done via a coupling network. Coupling of the* fasf *transient burst signal to I/O parts* is *normally done via a coupling c/amp. b--Here, you see a simplified test setup.*

Figure 8—In a simplified test setup for the radiated-electromagnetic field immunity test mandated by IEC 1000-4-3, the EUT is placed at distance d from the transmitting antenna, which produces a known field strength at the EUT.



- degradation or loss of function which is not recoverable due to damage of equipment or software or loss of data

A calling standard can use these classifications to evaluate test results or specify its'own.

The generic standards use performance criteria to evaluate test results:

- A-the equipment continues to operate as intended. No degradation of performance or loss of functionality is allowed during the tests. Criterion is applied to phenomena that are continuously present (e.g., REF).
- B-after the test, the equipment continues to operate as intended. During the test, performance degradation is allowed. Criterion is normally applied to transient tests (e.g., ESD or EFT/B).
- C--during the test, functionality may be lost. After the test, the equipment must recover on its own or via the operator. Criterion is normally applied to tests that cause interruptions beyond the control of the EUT (e.g., power-line interruptions).

Prior to starting the test, an evaluation criterion, normally one of the above, is selected. The test result is positive (i.e., pass) if the EUT operates as intended with respect to the evaluation criteria.

There is some flexibility in evaluating immunity test results, so to save time and, of course, money, carefully select normal operation and evaluation criteria.

HEADING TO THE LAB

The combination of last month's article on radiated and conducted emissions tests and this article on immunity tests gives the minimum test set that most unintentional radiators are subjected to.

If you know what to expect, you can modify designs to increase the chances of passing EMI testing on the first attempt.

Next month, I'll look at test labs, how to select a test lab, and what to do prior to, during, and after the trip to the lab. Remember that the test labs normally charge by the hour, so it's important to be prepared. ❑

Joe DiBartolomeo, P. Eng., has over 15 years' engineering experience. He currently works for Sensors and Software and also runs his own consulting company, Northern Engineering Associates. You may reach Joe at jdb.nea@sympatico.ca or by telephone at (905) 624-8909.

REFERENCES

FCC, *Code of Federal Regulations,* Title 47, Parts 15 and 18, 1995.

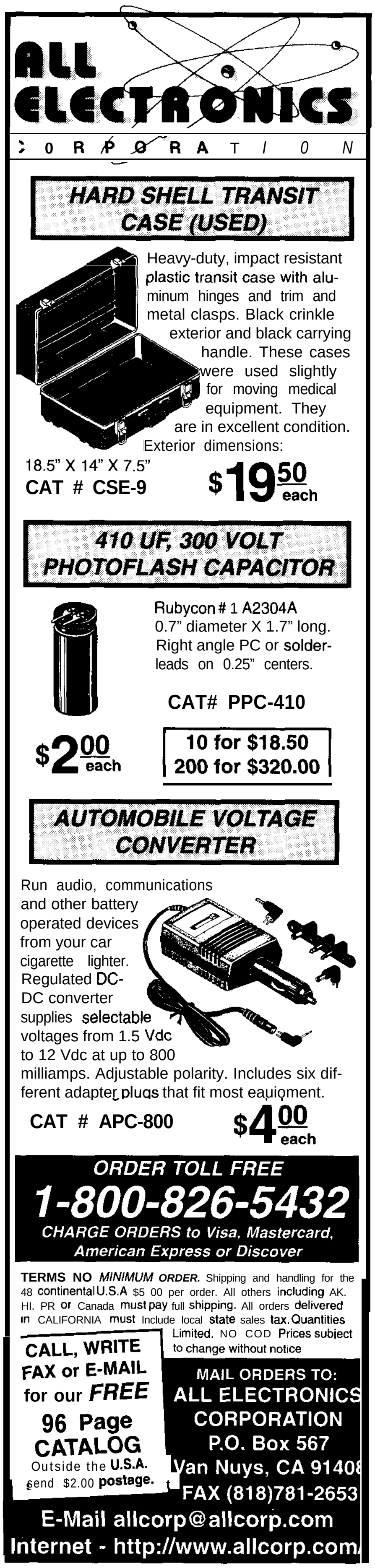
IEC Standard 1000-4-1, *Electromagnetic Compatibility, Testing and Measurement Techniques, Overview of Immunity Tests, Basic EMC Publication, 1992.*

C. Marshman, *The Guide to the EMC Directive 89/336/EEC,* EPA Press, Ambo, UK, 1992.

T. Williams, *EMC for Product Designers,* Butterworth and Heinemann, Oxford, UK, 1996.

I R S

419 Very Useful
420 Moderately Useful
421 Not Useful



FROM THE BENCH

Jeff Bachiochi

You Can Take It With You

Finding Your Way, Electronically

This month, Jeff takes the Precision Navigation compass and puts it in a hand-held portable enclosure with a processor, LCD, and power control. His goal: to build a compass that survives a week or two of outback camping.

I can't believe how small cell phones are becoming. Dick Tracy's wrist radio has certainly entered the realm of reality (though it'll still be a while before we get teleconferencing on our wrist).

You have to wonder where it will end up or what will happen to stop it. For now, I guess we can just ooh and ahh at new developments.

As a continuation of last month's investigation into Precision Navigation's Vector 2x compass, I want to show you how I interfaced this module to a microprocessor combined with an LCD and regulated power supply. Voilà, I created a portable electronic compass like the one in Photo 1.

Let's start with a little background on how the 2x module creates a bearing output from the earth's magnetic field.

RIDING MAGNETIC LINES

Since the earth's magnetic lines of force affect a coil like any other magnetic field, coils can be used to measure the force of the earth's field.

PNI uses the coil as part of an oscillator in their Vector 2x compass module. The oscillator's frequency changes with the strength of the magnetic field.

The frequency deviation is at its maximum when the coil aligns parallel to the earth's magnetic field and at its minimum when the coil is at right angles to the field. North and south can be easily determined, but east and west are still a bit nebulous.

Now, duplicate the same circuit, placing coils at right angles to one another-one at maximum deviation and the other at minimum. The outputs then vary like the sine and cosine of a particular angle.

With magnetic compasses, you must watch your surroundings. Close proximity to metal objects draws the earth's lines of force from their natural paths. Mechanical compasses cannot compensate and are drawn toward the-object.

Although the Vector 2x module compensates for static magnetic fields created by hard-iron distortions (e.g., the metal frame of your automobile), it doesn't have nonvolatile memory to store this calibration setting. Calibration is lost along with power.

If you use the module in a situation where calibration is necessary and don't want to manually calibrate it each time you power up, you must use it in the Raw mode. The Raw mode calculates and stores the calibration externally.

One calibration constant is needed for each axis (x and y). In Raw mode, the module outputs a 16-bit signed number for the x-axis followed by a 16-bit signed number for the y-axis.

The signed numbers reflect the relative field strength picked up by each of the coils. The coils' output should be equal but opposite when the module is rotated 180”.

Under perfect conditions, the lines of force move in a straight path. But, when a distortion is present, these lines move toward the source of the distortion as shown in Figure 1. This curve translates into a difference in measurements between two readings

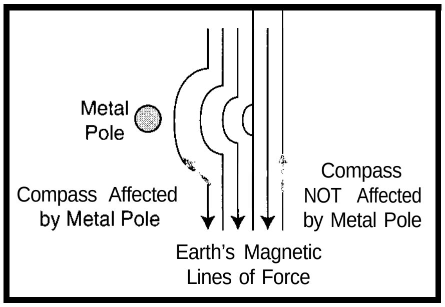

Metal Pole
Compass Affected by Metal Pole
Compass NOT Affected by Metal Pole
Earth's Magnetic Lines of Force

Figure 1—*A metal object can indeed affect the earth's magnetic lines of force just like light is drawn toward a black hole.*

of the same coil (180" from one another).

To calculate the offset constant for each axis, use the formula:

$$x_o = \frac{(x_1 + x_2)}{2}$$

$$y_o = \frac{(y_1 + y_2)}{2}$$

where x_0 is the offset constant for the x-axis, and x_1 and x_2 are the first two 16-bit signed numbers from each of the two Raw mode readings taken 180° apart.

Similarly, y_0 is the offset constant for the y-axis, and y_1 and y_2 are the second two 16-bit signed numbers from each of the two Raw mode readings taken 180" apart.

These two correction constants can be stored locally and used to adjust all future Raw data received while the module remains at its present location with respect to the distortion. Each offset constant must be subtracted from the Raw data before a bearing can be calculated using the formulas:

$$x_c = x_r - x_o$$

$$y_c = y_r - y_o$$

where c is the compensated value for the axis, r is the axis raw value, and o is the axis offset constant.

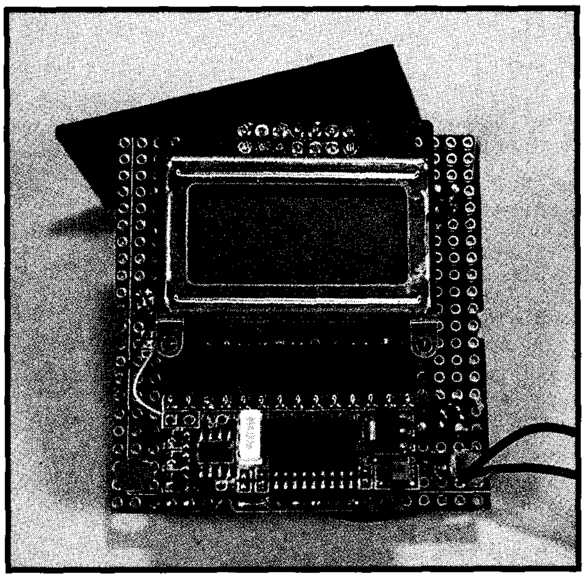


Photo 1—*The electronic compass is stripped out* **of ifs** *PacTec* **enclosure** *to show the small LCD and PicStic* **on** *the fop surface. The compass module is on the opposite* **side.**

		D3	D2	D1	DO	
Function Set	RS=0, R/*W=0	0	0	1	DL	high nibble
		N	F	X	X	low nibble
	where DL: l&-bit mode,, 0=4-bit mode N: 1=2 lines, 0=1line F: 1=5 x 11 dots/char, 0=5 x 7 dots/char X: don't care					
Display On/Off	RS=0, R/*W=0	0	0	0	0	high nibble
		1	D	C	B	low nibble
	where D: 1 =display On, 0=display Off C: 1=cursor On, O=cursor Off B: 1=cursor blink, O=cursor steady					
Shift Set	RS=0, R/*W=0	0	0	0	1	high nibble
		S/C	R/L	X	X	low nibble
	where S/C: 1=display shift, O=cursor shift R/L: 1=shift right, O=shift left X: don't care					
Entry Mode	RS=0, R/*W=0	0	0	0	0	high nibble
		0	1	I/D	S	low nibble
	where I/D:1=increment , O=decrement S: 1 =auto shift, 0=no shift					

Table 1—*These* **nibble commands initialize** *the* **2 x 8 LCD.**

To calculate the bearing from 0–359°, use:

$$B = \tan\left(\frac{x_c}{y_c}\right)$$

where the bearing is equal to the arctangent of the compensated x-axis value divided by the compensated y-axis value.

CHARACTER-CLASS LCDs

Small dot-matrix LCDs usually come with a built-in processor for controlling the dots (segments) and providing control of a precoded character set. This eliminates the overhead necessary for enabling individual **dot** segments, keeping the interface and commands to an absolute minimum. The hardware interface consists of an 8- or 4-bit data bus and up to three control signals.

The smallest LCD I've seen is the Optrex DMC-50448. This 2-line x 8-character display measures less than 1.5 in^2.

LCDs that are small enough to fit into handheld enclosures are difficult to find, but this one fits well in the Pac Tec HM-9V plastic box. The 9-V battery compartment built into the case allows easy access to the battery without removing screws.

I used a simple interface. Only two control lines are needed to permanently enable the LCD's enable line. The 4-bit data mode requires two writes for each character, but it's a good tradeoff over the 8-bit bus width.

Initializing the LCD is a bit tricky for the 4-bit mode since the first write is always in 8-bit mode. The 4-bit bus uses bits 4-7 on the LCD. The first write to the control register (RS=O) assumes the interface is 8 bits wide. If the value 2 (0010) is written on the nibble bus, it's read as an 8-bit word 0010xxxx (the lower bits are don't care).

This command places the LCD's processor into 4-bit mode for the rest of the command transfers. From now on, all values must be passed in high- or low-nibble format.

Only a few registers need initializing before the LCD can be used. Since most LCDs use the same processor regardless of their size, the same procedures are appropriate for any of them.

In addition to the control commands shown in Table 1, there are also commands to home the display and cursor to location 0,0(row,column) and to clear the display. Addressable DD (data display] and CG [character generator) RAM provide powerful positioning potentials. Listing 1 shows the commands to initialize this tiny 2 x 8 LCD.

PRESERVING PRIMARY POWER

I want this compass to fit in my shirt pocket, and I'd like to go out for a week or two and not worry about my battery dying. To conserve battery life, I picked a regulator with an on/off control line-Maxim's MAX833.

When the control input to the regulator is held high, the regulator is on. If it's driven low, it goes into shutdown and draws only 1 mA from the battery. Referring to Figure 2, notice while the power is off, the input is held low with a 100-kΩ resistor.

To turn the compass on, a push button connects the battery temporarily to the regulator's control input, which turns on the regulator's 5-V output. This 5-V output powers the three components of the compass-the LCD, the 2x compass module, and the processor.

The first thing the processor does is set an output high which holds the regulator's control input high. This happens before you can remove your finger from the power switch, allowing the power to remain on.

PYGMY PROCESSING PROPORTIONS

I wanted to use a tiny processor for obvious reasons. I fully intended to put a PIC processor directly on the circuit board. However, I broke down and put a SIP socket on the circuit board and wired it to accept a PicStic **1** module, which already has the crystal and I/O prewired to SIP pins. This approach really simplifies the amount of wiring necessary for the whole project.

The processor initializes the LCD, asks the Vector 2x compass module for a bearing, and displays this bearing in degrees. The LCD uses six output lines of port B. The first four are data, and the second two are the control lines.

The data is strobed into the LCD by the E line. The LCD sees the data as character or control data depending on the state of the RS (register select) line.

Of the two remaining bits on port B, one requests a conversion from the 2x module and the last bit holds the regulator's control input high. The PicStic has two additional bits (port A bits 3 and 4). The compass module has two outputs which must be read by the processor, data, and end-of-conversion outputs. These are wired to the two bits on port A.

Now all the I/O is used, but two more functions are needed. It's time to share bits.

Since the LCD data is only strobed into the LCD during an E-enabled strobe, there's no problem sharing one of the data's output port bits (port B bit 0) with the compass module. This output bit becomes not only the least significant bit for the LCD but also the SCK to the compass that clocks out the data following a conversion.

The conversion is started by toggling the P/C line low for **10** ms. The EOC

Listing 1—*This partial listing shows bow compiled BASIC commands form the main loop of fhe electronic compass program.*

```
START1: B8=0
LOOP:    W5=W5+1
         IF W5>2000 THEN SLP
        BO=PINS
        B0=B0&$F0
        BO=BO | $01
        PINS= B0
LOOP1:   GOSUB EPC              'ENABLE VECTOR 2x CONVERSION
        PAUSE  10
        GOSUB DPC
        CALL CHKEOC             'CHECK END OF CONVERSION
         IF B0<>0 THEN LOOP3
LOOP2:  CALL CHKEOC
         IF BO=O THEN  LOOP2
LOOP3:  PAUSE 10                'GIVE THE VECTOR 2x SETTLING TIME
        CALL EPC
        PAUSE 10
        CALL CONV               'GET THE CONVERSION
        GOSUB DPC               'DISABLE VECTOR 2x (SLEEP)
        PAUSE 10
        CALL CHKSW              'IS THE MODE SWITCH PRESSED
         IF MODE>1 THEN LOOP4
         IF MODE=0 THEN MODE0  'MODE=0 IS BEARING MODE
        GOT0 MODE1
LOOP4:   IF MODE=129 THEN WMODEO
        MODE=1
        WRITE 0,MODE: WRITE 1,B5: WRITE 2,B6: WRITE 3,B7

MODE1:  W8=B5*100 : W6=B6*10 : W6=W6+W8 : W6=W6+B7 'CALC BEARING
        READ 1,B16 : READ 2,B14 : READ 3,B9
        W8=B16*100 : W7=B14*10 : W7=W7+W8 : W7=W7+B9 'GET OLD BEARING
        W8=W7-W6                'FIND THE DIFFERENCE
        SEROUT 7,N9600,(#W6,13,10)
         IF W8=0 THEN LCD3      'SAME
         IF W8=359 THEN LCD4    '359
         IF W8=65535 THEN LCD4 '-1
         IF W8=358 THEN LCD5    '358
         IF W8=65534 THEN LCD5 '-2
         IF W8=65177 THEN LCD2 '-359
         IF W8=1 THEN LCD2      '1
         IF W8=65178 THEN LCD1 '-358
         IF W8=2 THEN LCD1      '2
         IF W8>32000 THEN CHK_180        'MINUS NUMBER

CHK180: IF W8<180 THEN LCD6    '<180
        GOT0 LCD0              '>180
CHK_180:IF W8>65356 THEN LCD6 '>-180
        GOT0 LCD0              '<-180

LCDO:   B12=60 : B13=60 : B14=60 : B15=42
        B16=32 : B17=32 : B18=32 : GOT0 F_LINE  '<<<*____
LCD1:   B12=32 : B13=60 : B14=60 : B15=42
        B16=32 : B17=32 : B18=32 : GOT0 F_LINE  '_<<*____
LCD2:   B12=32 : B13=32 : B14=60 : B15=42
        B16=32 : B17=32 : B18=32 : GOT0 F LINE  '__<*____
LCD3:   B12=32 : B13=32 : B14=32 : B15=42
        B16=32 : B17=32 : B18=32 :  GOT0 F_LINE  '____*____
LCD4:   B12=32 : B13=32 : B14=32 :  B15=42
        B16=62 : B17=32 : B18=32 :  GOT0 F_LINE  '____*>__
LCD5:   B12=32 : B13=32 : B14=32 :  B15=42
        B16=62 : B17=62 : B18=32 :  GOT0 F LINE  '____*>>_
LCD6:   B12=32   B13=32 : B14=32 :  B15=42
        B16=62   B17=62 : B18=62 :  GOT0 F_LINE  '____*>>>

F-LINE:  GOSUB P_LINE
        READ 1,B5 : READ 2,B6 : READ 3,B7
        B12=32 : B13=32 : B14=B5+48 : B15=B6+48
        B16=B7+48 : B17=DEG : B18=32
        GOSUB P_LINE
        GOT0 LOOP

WMODEO:  MODE=0
         WRITE 0,MODE
MODEO:  B12=32 : B13=32 : B14=B5+48 : B15=B6+48
        B16=B7+48 : B17=DEG : B18=32
        GOSUB P_LINE
```

(continued)

line drops and remains low until an EOC occurs. The P/C line may be now dropped low again to enable the conversion data to be read from the module.

The processor clocks the SCK line and reads the data the 2x module places on the SDO line. When all 16 bits (32 bits in Raw mode) have been read, the P/C line is raised, tristating the SDO output line from the compass.

The reason I mention tristating is that I share this input with a second push button. A small pull-up resistor holds this tristated line high while the push button isn't pushed.

Pushing the button pulls down the line with a smaller but still weak pull-down so that, if the button is held down during a compass conversion read, it doesn't interfere with the data driven from the compass.

However, after the compass's data has been transferred, the switch can be sampled for input status because the compass is no longer driving the line.

This second push button toggles the compass's mode. In mode 0, the display shows the compass bearing. When

Listing 1—*continued*

```
        B12="B":B13="e": B14="a": B15="r": B16="i": B17="n": B18="g"
        GOSUB P_LINE
        GOT0 LOOP
P_LINE  FOR BO=O TO 7             '8 CHARS TO PRINT
        LOOKUP B0,(B12,B13,B14,B15,B16,B17,B18,32),B1 'THE CHARS
        LCD=B1/16                 'MSNibble
        GOSUB STB                 'TO LCD
        PAUSE 1
        LCD=B1//16                'LSNibble
        GOSUB STB                 'TO LCD
        PAUSE 1
        NEXT BO                   'DO ALL 8
        RETURN

SLP:    LOW 7
        GOT0 SLP


.
ASM

_CHKEOCCLRF     _B0               ; GET EOC FROM VECTOR 2x MODULE
        BTFSC   PORTA,4
        BSF     _B0,0
        GOT0 DONE

_CONV   MOVLW   _B4               ; GETS CONV FROM VECTOR 2x MODULE
        MOVWF   FSR
        MOVLW   4
        MOVWF   _B19
_LP1    MOVLW   4
        MOVWF   _B21
        CLRF    _B20
_LP2    BCF     PORTB,0
        BSF     PORTB,0
```

(continued)

Listing 1-continued

```
        BCF     STATUS,0
        BTFSC   PORTA,3
        BSF     STATUS.0
        RLF     _B20
        DECFSZ  _B21
        GOTO    _LP2
        MOVF    _B20,W
        MOVWF   INDF
        INCF    FSR
        DECFSZ  _B19
        GOTO    _LP1
        GOTO    DONE

_CHKSW  BTFSS   PORTA,3    SEE IF MODE SWITCH IS PRESSED
        BSF     _B8,7
        GOTO    DONE
ENDASM
```

the second button toggles the mode (to mode 1}, the present bearing is stored into EEPROM. The display then shows which direction you must turn to stay on that particular bearing.

An asterisk alone indicates you are dead on, and arrows to the right or left reveal the direction you need to turn to remain on that bearing. Additional arrows indicate how far off course you are. The stored bearing is also displayed below the arrows.

The processor subtracts the present bearing from the stored bearing. This difference determines how many arrows are displayed and in which direction. A press on the second button returns you to mode 0, and you again have a real-time display of the present bearing.

The compass does not have an off switch. Instead, the processor counts conversions and drops the control line to the regulator, turning itself off if the second button isn't pressed within the time-out period (presently set to several minutes).

The mode is also saved in EEPROM when it is switched. This way you can set a bearing, store it (mode 1}, and romp off toward your first bearing. Even if the compass shuts off, when you power it up, you return to the last mode you were in. You're immediately ready to find another bearing point using the last stored bearing.

FIND YOUR OWN WAY

I hope these articles have started the gears turning in your head. Whether you're an orienteer, robotics junky, or just want to build a cool new display for your car's dashboard, I think you'll have fun with this compass module.

Oh yeah, computations based on operating current shows this circuit will run approximately 50 continuous hours on a standard 9-V alkaline battery. I think that's sufficient to get me wherever it is I'll need to go-and back again. Anyone need directions? ❑

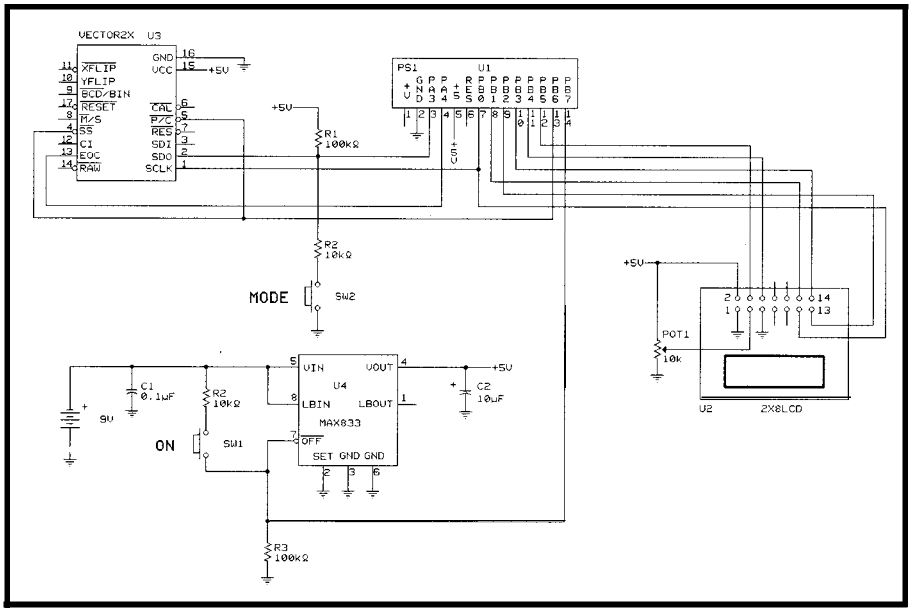


Figure 2-The four main circuit components include the compass module, processor, **display,** and power control.

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@circellar.com.*

SOURCES

Vector 2x compass module
Precision Navigation, Inc.
1235 Pear Ave., Ste. 111
Mountain View, CA 94043
(415) 962-8777
Fax: (415) 962-8776

DMC-40448 2 x 8 LCD
Optrex America
44160 Plymouth Oaks Dr.
Plymouth, MI 48 170
(313) 416-8500
Fax: (313) 471-4767

PicStic 1 microcontroller
Micromint, Inc.
4 Park St.
Vernon, CT 06066
(860) 871-6170
Fax: (860) 872-2204

HM-9V Pac Tec Enclosure
Pac Tec
8425 Executive Ave.
Philadelphia, PA 19 153-3893
(215) 365-8400
Fax: (215) 365-4420

MAX833
Maxim Integrated Products
120 San Gabriel Dr.
Sunnyvale, CA 94086
(408) 737-7600
Fax: (408) 737-7194

I R S

422 Very Useful
423 Moderately Useful
424 Not Useful

Not Your AVRage MCU

SILICON UPDATE

Tom Cantrell

When it comes to Atmel's AVR family, Tom feels 8 bits provide plenty of excitement and RISC. It's not high end, but it is highly integrated, offering a load-and-store register-oriented architecture and a 10+ MIPS rate.

With all the hoopla over PCs and 32-bit+ superchips, it's easy to overlook the low end. It's true that most popular 8-bit architectures could use a shot of Grecian Formula, so it might appear there's not a lot of action to overlook.

I admit desktop machines are fascinating-especially now that high-end microprocessors, having achieved supremacy, will be the horses that carry us beyond the frontiers of computing as we know it.

I've heard it said that the PC now represents 30% of the chip business. I'm not sure exactly what that means (e.g., revenue vs. units, just PC boards, or all the I/O devices as well). But anyway you cut it, it's a huge number.

Nevertheless, if 30% is PC, that means 70% isn't. And, 8 bits is a big chunk of that. In fact, though it may not have the sizzle of the PC biz, I think the low-end embedded front is more exciting in many ways.

Whether it's kinder and gentler air bags or a self-cleaning kitty box, there's no shortage of emerging applications. However, unless suppliers press on with ever better chips, it will be hard to keep up the pace of innovation.

Blessedly for all of us fortunate enough to work with chips, there's always a company able and willing to prod things along just when a bit-or 8 bits, in the case of the Atmel AVR family-of excitement is called for.

RISC VS. REALITY

Naturally, Atmel labels AVR a RISC, a term that has devolved to mean anything that isn't an 'x86 (which itself is morphing into a RISC anyway).

Nevertheless, AVR adheres to the original spirit, if not letter, of RISC more than many. At the same time, AVR designers weren't afraid to compromise theological purity for the sake of practicality.

What are the worthwhile lessons to take away from the RISC revolution? Here's my take on the key points and how AVR measures up.

Much of the impetus for the RISC was fueled by the desire to move up from ASM to C. Though I'm not a big fan of the language, there's no doubt that a machine without a C compiler is hard to sell, so you may as well have a good one.

This shift, in turn, forced computer architects to break the ice with compiler writers. Lo and behold, the hardware folks discovered all their fancy instructions and intricate addressing modes were of little use to a code generator. Worse, the extra instruction bloat not only increased the cost of the chip but limited the speed.

The essential by-product of the new cooperation between CPU and compiler designers was the "load/store" machine with "lots of registers." These terms refer to a design in which the only way to access (slow) memory is by load and store instructions, while all other instructions operate only on (fast) registers, which is why lots of them are needed.

AVR fully lives up to the basic tenets of RISC with a load/store architecture and programmer's model comprising 32 eight-bit registers. For the most part, the registers are completely general purpose, one minor exception being that a few serve double duty as index registers. Also, operations with immediates are restricted to the top half (R1631) of the register file.

Another by-product of the keep-it-simple-and-fast philosophy, especially in the context of pipelining, is that RISCs traditionally featured fixed-length instructions. AVR passes muster here as well since practically all instructions are 16 bits long.

High clock rates and pipelining also go hand in hand. After all, maximum CPU clock rate is only as fast as the quickest pipe stage. However, increasing the pipeline depth and clock rate is less of a concern (or even a penalty, given cost and usability issues) in embedded apps. The AVR compromise-a two-stage pipeline running at up to 24 MHz-seems reasonable.

RISCs are often characterized as offering single-cycle instruction execution. Strictly speaking, this mainly refers to ALU ops, with loads and stores, conditional branches, and numerics usually taking more. Even the fanciest desktop CPUs, though able to dispatch multiple instructions per clock, are hard pressed to sustain more than 1 IPC (Instruction Per Clock) for real programs.

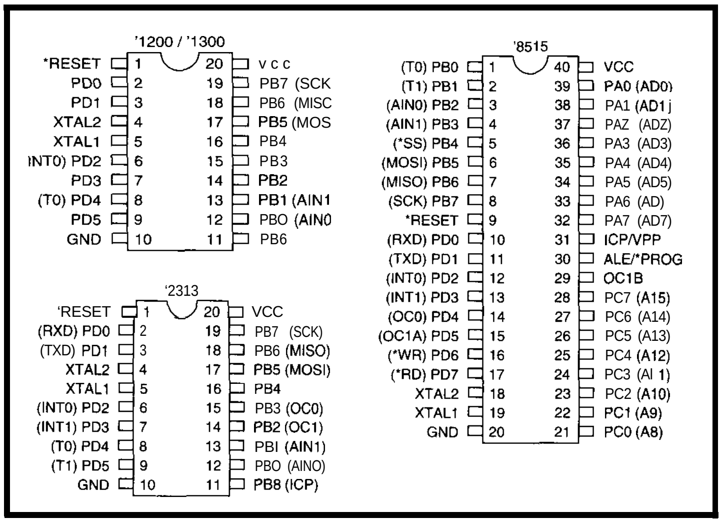


Figure I--Three versions *of the AVR MCU have been announced--the '1200/'1300, the '2313, and the '8515.*

Here, AVR does an excellent job. With a few minor exceptions, ALU ops execute in a single cycle, as do untaken conditional branches. Taken branches and loads and stores consume two clocks, with only a handful of instructions taking more (e.g., the slowest instruction, RET, is four clocks).

Contrast this with other 8-bit chips penalized by lower clock rates (don't forget the marketing megahertz is often divided down by the time it gets to the CPU), more (sometimes many more) clocks per instruction, or both. With an average instruction probably taking a mere two clocks or so, AVR delivers a superior 10+ MIPs at full throttle.

Finally, though there's been a tendency towards creeping CISCism, AVR exhibits admirable restraint in this regard. The chip has a very straightforward instruction set that's notably easy to program in assembler. The same can't be said for desktop compiler-centric RISCs, which in the interest of supporting C, practically force you to use it at all times.

Architecture is well and good, but it isn't the only critical factor in typical embedded applications. Equally important are the decisions concerning what memory and I/O are integrated and how well they work.

So far, Atmel has announced three delivery vehicles for the AVR architecture as you see in Figure 1. Given the CPU's minimalism, the lineup fittingly starts with the lean and mean '1200/ '1300 at $1.80 in I k quantities. Adding a little more memory and I/O gets you to the '2313 for $3. Finally, the '8515 ($6), puts more onchip and supports external memory and I/O expansion.

As with the architecture, Atmel has done a good job assimilating the best and brightest memory and I/O features. In other words, there's nothing that hasn't been done before, but also nothing missing or kludgy.

Though not the first to integrate flash for program store or EEPROM for data, Atmel is one of the first to offer both. Furthermore, programmability is accommodated from the beginning rather than added on at the end.

For example, all AVRs support both low-voltage serial programming (see Table 1) and high-voltage (12 V) parallel programming. This enables the designer to combine the best aspects of in-system and gang programming, depending on the particulars of the application.

Similarly, where a tacked-on EEPROM might require a slow, complicated driver, AVR's built-in intelligence makes access easy. The AVR driver is a whopping two instructions long-specify the EEPROM address and issue a read or write command. Write timing (~2.5 ms) is as simple as polling a bit, while reads are full speed.

REDUCED IN SIZE COMPUTER

Photo 1 offers a close look inside the entry-level '1200. Note that the '1300 is exactly the same, except it doubles the onchip data EEPROM size from 64 to 128 bytes.

On the surface, the '1200 block diagram in Figure 2 looks conventional enough. But, there are more than a few interesting points worth noting.

For instance, the '1200 is more an in-out than a load-store machine. I/O functions (including the data EEPROM) are accessed using separate In and Out instructions, as shown in Table 2.

Thus, without any onchip RAM or external bus, there aren't many places to load from or store to. Besides immediate loads, the chip does support indirect load and store within the register file by

Instruction	Instruction Format				Operation
	Byte 1	Byte 2	Byte 3	Byte 4	
Programming Enable	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx	Enable serial programming after RST goes high
Chip Erase	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx	Chip erase both 1-K and 64-byte memory arrays
Read Program Memory	0010 H000	xxxx xxxa	bbbb bbbb	oooo oooo	Read H (high or low) data o from program memory at word address a:b
Write Program Memory	0110 H000	xxxx xxxa	bbbb bbbb	iiii iiii	Write H (high or low) data i to program memory at word address a:b
Read EEPROM Memory	1010 0000	xxxx xxxx	xxbb bbbb	oooo oooo	Readdataofrom EEPROM ataddress b
Write EEPROM Memory	1110 0000	xxxx xxxx	xxbb bbbb	iiii iiii	Write data i to EEPROM at address b
Write Lock Bits	1010 1100	111xx012	xxxx xxxx	xxxx xxxx	Writelockbits. Set bits *1,2*='0' to program lock bits
Read Device Code	0011 0000	xxxx xxxx	xxxx xxxx	oooo oooo	Readdevicecodeo

Table 1—*All AVRs* **can** *be* **programmed in system via** *clocked serial port by issuing the Programmfng Enable command while Reset is asserted.*

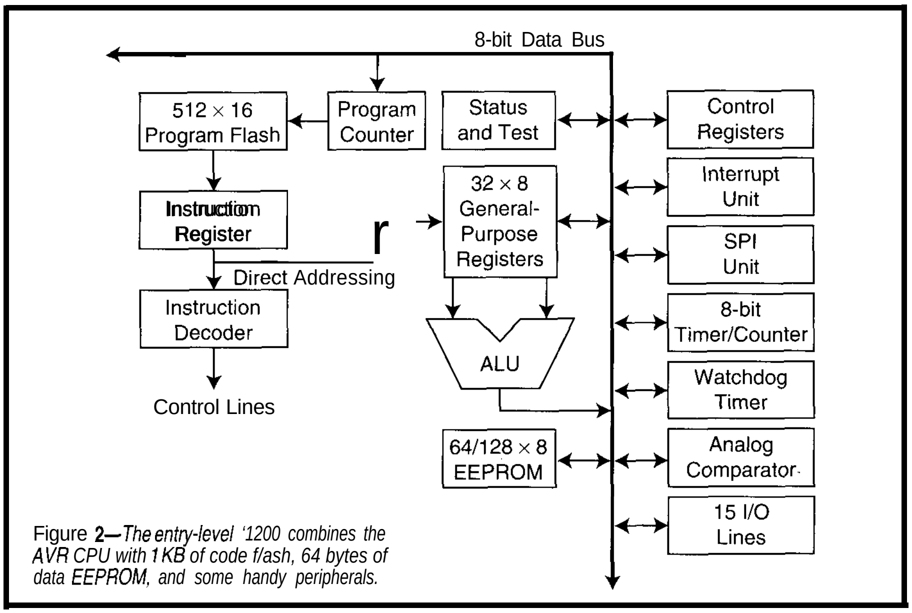


Figure 2—*The entry-level '1200 combines the AVR CPU with 1 KB of code f/ash, 64 bytes of data EEPROM, and some handy peripherals.*

assigning one register (R30, also known as Z) as a pointer to another.

Note the internal oscillator provides an autonomous timebase (8^2 timeout selections between 16 and 2048 ms) for the watchdog timer. So, it keeps working even when the main oscillator is shut off in power-down mode.

The fact that it runs at 1 MHz seems like overkill, but it proves handy. The chip can be configured to run off the internal clock, eliminating an external crystal or clock source altogether.

One place the '1200 seems to cut a little deep is the hardware stack. It's only three levels, yet it must accommodate both hardware interrupts and CALL and RET instructions. Furthermore, the stack is not accessible via software (i.e., no PUSH or POP instructions, not that there's room to store much there anyway).

However, there are extenuating circumstances. Most basic is that, without any RAM, there's no place for a stack. And, without a stack, a stack pointer seems superfluous. In fact, the bigger AVR siblings with onchip RAM do have a conventional S P, PUSH, POP, and so forth.

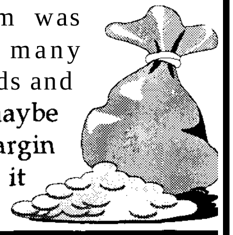
And, there are only three interrupt sources-external (INTO, programmable as either low-level or positive- or negative-edge triggered), timer, and analog comparator. The fact that a '1200 program is quite small in size (5 12 instructions max.) would seem to limit pressure on the stack as well.

Overall, I don't really have a problem with the '1200 hardware stack concept, but I wonder if it shouldn't be just a few levels deeper. The good news is the lack of stack activity (only the PC is saved) translates into a speedy interrupt response of only four clocks.

One likely source of an interrupt is the 8-bit timer/counter. Befitting the minimalist philosophy, the

Address Hex	Name	Function
$3F	SREG	Status register
$3B	GIMSK	General interrupt mask register
$39	TIMSK	Timer/counter interrupt mask register
$38	TIFR	Timer/counter interrupt flag register
$35	MCUCR	MCU general control register
$33	TCCRO	Timer/counter 0 control register
$32	TCNTO	Timer/counter 0 (8 bit)
$21	WDTCR	Watchdog timer control register
$1E	EEAR	EEPROM address register
$1D	EEDR	EEPROM data register
$1C	EECR	EEPROM control register
$18	PORTB	Data register, port B
$17	DDRB	Data direction register, port B
$16	PINB	Input pins, port B
$12	PORTD	Data register, port D
$11	DDRD	Data direction register, port D
$10	PIND	Input pins, port D
$08	ACSR	Analog comparator control and status register

Table 2 —*AVR reverts to the concept of using special In and Out instructions, rather than memory mapping, for I/O register access. However, the instruction set also includes opcodes to set, clear, and branch on I/O register bits.*



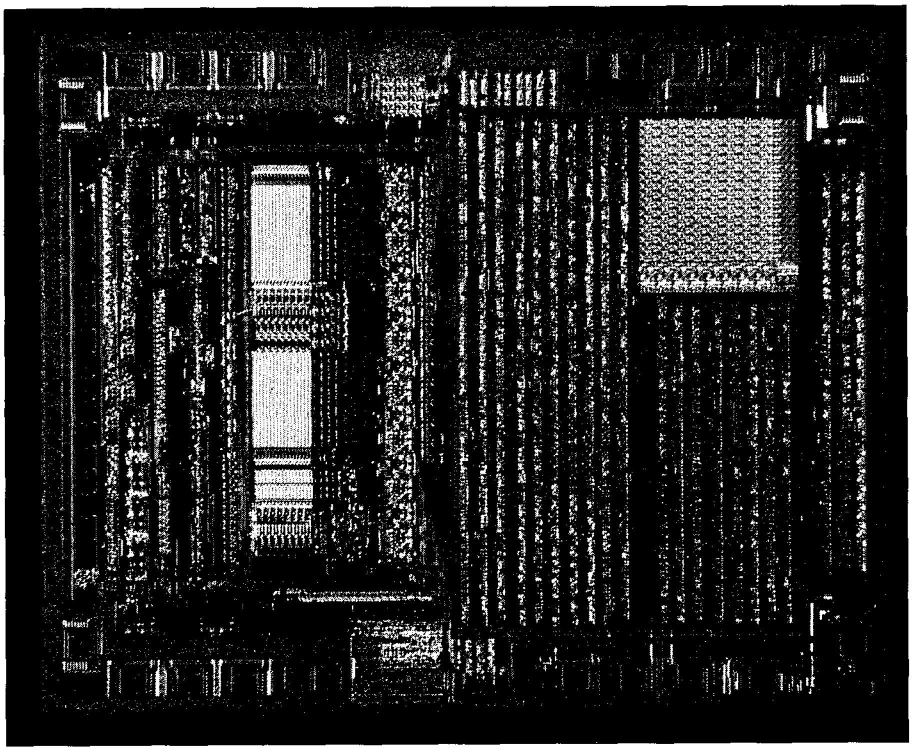


Photo I--The *'1200* accesses *I/O* and memory using *separate In/Out,* rather than load/store, instructions.

unit doesn't feature a lot of modes and functions. However, it handles basic duties with speed and versatility.

Internally prescaled clock selections include CLK, CLK/8, /64, /256, and /1024 (i.e., from 50 ns to -50 µs at 20 MHz]. Alternatively, an external clock input (with programmable edge polarity) running up to CLK/2 can be connected.

The analog comparator lives up to its name by comparing the values on two pins. Its single-bit output can be read directly or generate an interrupt when it goes high or low or toggles.

A simple comparator like this can be coerced into duty as an ADC. One way is to discharge an RC that ramps a comparison voltage and measure the time it takes to trigger the comparator. Another hack is to generate the comparison voltage with a poor man's DAC concocted from a PWM output or external resistor network.

I/O lines not otherwise allocated for specific use (e.g., comparator, timer, serial port, etc.) are bit programmable for direction. In input mode, programmable pullups are offered, as is the ability to read the output latch or pin. Rail-to-rail outputs feature high drive (IOL up to 20 mA} capable of driving LEDs directly.

BIGGER BROTHERS

The '23 13 retains the 20-pin form factor while packing some more memory and a lot more I/O. Compared to the '1200, both program flash and data EEPROM capacity are doubled to 2 KB and 128 bytes, respectively.

The '23 13 also includes 64 bytes of RAM. Besides giving those load and store instructions something to do, the appearance of RAM comes with minor architectural upgrades. These include a conventional stack, 16-bit ADD and SUB, and extra addressing modes (e.g., indirect with displacement, autoincrement, autodecrement, etc.) which conscript a few of the general-purpose registers for x, y, and z index duty.

Besides all the '1200 functions, the '23 13 goes further on the I/O front by incorporating some big-ticket peripherals. It's 16-bit timer/counter (shown in Figure 3) has all the options-input capture (with glitch filter), output compare, PWM mode, and the like.

There's also a full duplex UART with a dedicated baud-rate generator that can generate standard baud rates independently of the CPU clock rate. For instance, the CPU can handle most

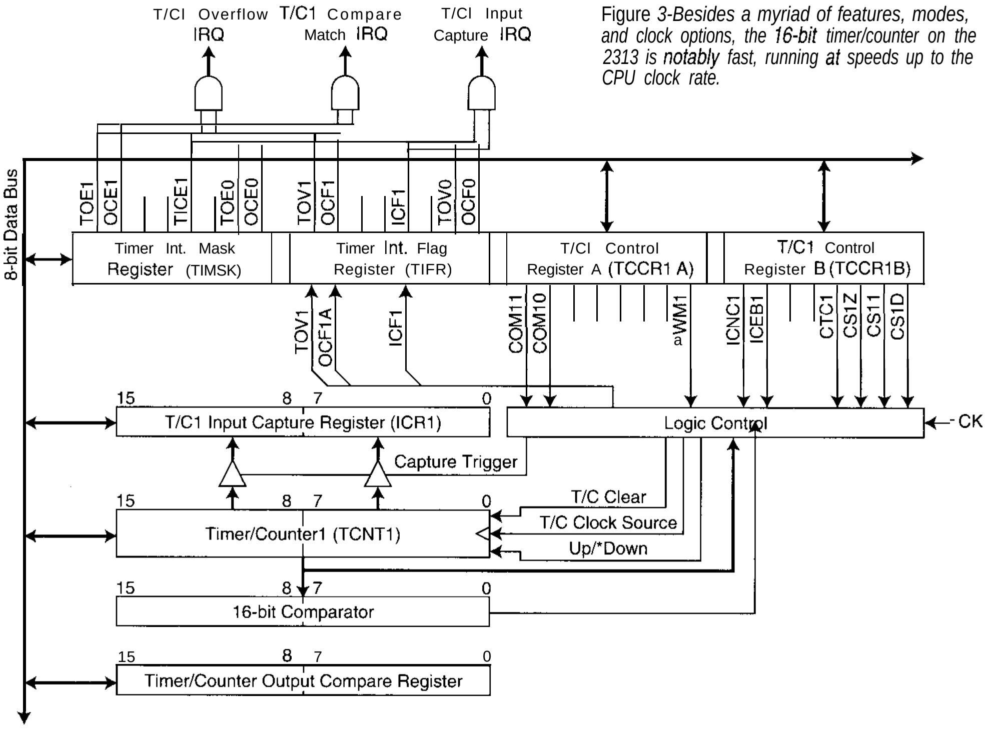


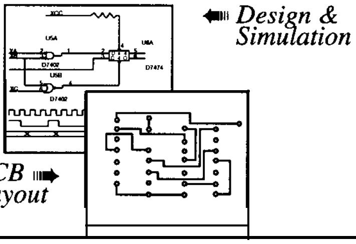
Figure 3-*Besides a myriad of features, modes, and clock options, the 16-bit timer/counter on the 2313 is notably fast, running at speeds up to the CPU clock rate.*

standard data rates from 2.4 to 115 kbps, at more than a dozen frequencies between 1.8432 and 20 MHz.

The UART also has false start bit, framing, and overrun detection. It includes a 9-data-bit format that provides a hook for parity checking or multidrop network addressing, but full implementation of those features requires software.

Using the '23 13 architecture and I/O enhancements as a base, the '85 15 further increases memory to 8-KB program flash, 256 bytes of data EEPROM, and 256 bytes of RAM. Switching to a 40- or 44-pin (DIP or PLCC) package enables the larger chip to support external bus expansion for memory or I/O. Also, the '8515 makes the SPI (Serial Peripheral Interface) port, restricted to serial programming duty on the smaller chips, available for application use as well.

The bus interface relies on ALE (Address Latch Enable) and *RD and * WR strobes familiar to 805 1 users. In fact, a close look at the '8515 pinout shows that it



can practically plug into an 805 1 socket. This makes sense when you remember that Atmel has been offering '5 1 derivatives for some time.

The fact the data bus (i.e., ADO-7) is only eight bits wide may give pause. However, unlike the '5 1, the '85 15 external bus is intended only for data access, not instruction fetch, somewhat alleviating bus-bottleneck concerns.

Keep in mind that, also like the '51, the '85 15 doesn't have any WAIT input or onchip wait-state generator. Though it may not be an issue for older, more leisurely CPUs, bus timing is a concern for '85 15s running at higher clock rates. A memory or I/O chip has to offer an access time somewhat less than the CPU cycle time (i.e., under 50 ns for a 20 MHz CPU).

C-U-LATER

There's always a lag between chip introduction and broad third-party support. Meantime, Atmel is covering the bases with its own emulator and low-level software tools, as well as a full-fledged C suite developed by IAR.

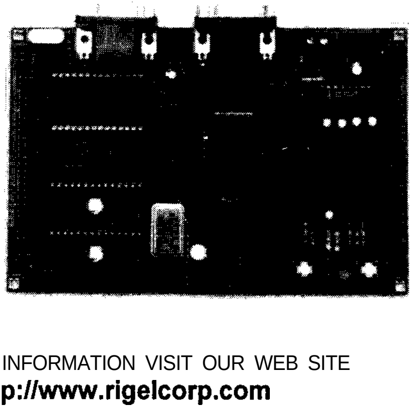
Type	Kevword	Description
Function	interrupt	Creates an interrupt function that is called through an interrupt vector. The function preserves the register contents and the processor status.
	monitor	Turns off the interrupts while executing a monitor function
	C-task	Declares a function as not callable (e.g., main) to save stack
Variable	no_init	Puts a variable in the no_init segment (battery-backed RAM)
	sfrb	Maps a byte value to an absolute address
	sfrw	Maps a word to an absolute address
	tiny	Accesses using 8-bit address
	near	Accesses using 16-bit address
	flash	Accesses in the program address space
Segment	codeseg	Renames the Code segment
	constseg	Creates a new segment for constant data
	dataseg	Creates a new Data segment (These are mostly used to place code and data sections in nonconsecutive address ranges.)
Intrinsic	_SEI	Enables interrupt
	_CLI	Disables interrupt
	_NOP	Inserts NOP instruction
	OPC	Inserts the opcode of an instruction into the object code
	_LPM	Returns one byte from the program address space
	SLEEP	Enters sleep mode
	WDR	Resets watchdog

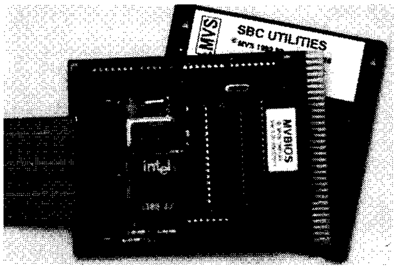
Table 3—*In addition to full ANSI compatibility, the IAR C compiler includes a number of practical extensions as well.*

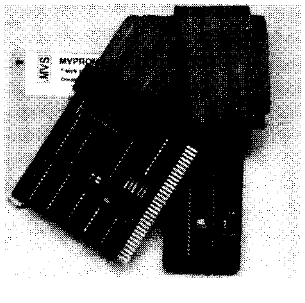
The tool chain starts with Atmel's Windows-based assembler and simulator. The latter gets extra credit for simulating the operation of interrupts and onchip I/O as well as the CPU.

Atmel certainly encourages taking a closer look at AVR with an aggressive price for these tools-namely, $O! You'll find they're free for the taking on the company's CD-ROM databook and Web page.

By contrast, the Atmel emulator is a pricey item intended for industrial-strength development. Based on actual hardware emulation of the CPU, the unit delivers hardcore real-time capabilities not found in entry-level tools. Beyond powerful breakpoint and execu-







tion control facilities, advanced features include 32 K x 96-bit trace buffer, 32-bit time stamp, and software-adjustable clock.

Similarly, the C compiler from IAR is loaded with full ANSI C compatibility, including IEEE 32-bit floating point, struct, union, enum, bitfield, and so forth. It costs $1595 for the compiler only or $2195 with simulator and source level debugger. DOS and Windows versions are available.

That's grand, but I suspect I'm not alone in complaining that it's often difficult to figure out how to make C do simple things [e.g., write an I/O port or handle an interrupt). Thankfully, IAR includes a number of handy extensions in this regard, as listed in Table 3.

BEST OF BREED?

Overall, I think it should be clear by now that AVR is a real dog-and a mutt to boot! Now, before the folks over at Atmel blow a gasket (or put out a contract), let me deftly extricate myself.

First of all, remember that old saying that a dog is man's best friend? Well, 8-bit chips are a designer's best friend, at least when it comes to whipping up ever-whizzier and more affordable embedded gadgets. And from my experience, a mutt is far more likely to make a good friend than a persnickety, bordering-on-psychotic purebred.

Remember that purebreds are, as the name implies, bred purely for a special purpose far removed from the owner's current application. I mean, how many sheepdogs actually get to herd sheep? As well, purebreds all too often seem to demonstrate the dangers of swimming in a stagnant gene pool.

By contrast, a good mutt combines the best characteristics of its various ancestors, while avoiding the worst excesses. With the speed of a PIC, easy register-oriented architecture of a Z8, and bit-handling of a '5 1, but with none of these chips' historical warts, AVR may be just the doggy in the window for you. ❑

Tom Cantrell has been working on chip, board, and systems design and marketing in Silicon Valley for more than ten years. He may be reached by E-mail at tom.cantrell@circellar.com, by telephone at (510) 657-0264, or by fax at (510) 657-5441

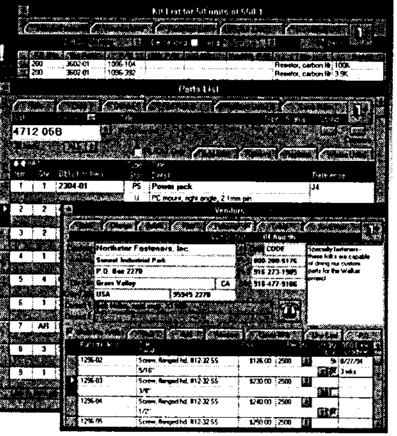
SOURCES

AVR
Atmel Corp.
325 Orchard Pkwy.
San Jose, CA 95131
(408) 441-0311
Fax: (408) 436-4300
BBS: (408) 436-4309
literature@atmel.com
www.atmel.com

C compiler
IAR Systems
1 Maritime Plaza
San Francisco, CA 94111
(415) 765-5500
info@iar.com
www.iar.com

I R S

425 Very Useful
426 Moderately Useful
427 Not Useful



PRIORITY INTERRUPT

It's Win-Win and $10,000

given my approach to solving design problems, it's no wonder I'll probably be remembered most for saying, "My favorite programming language is solder!" While I stilt profess it to some extent, the reality of making cost-effective engineering decisions has seriously modified that stance. In fact, these days, it really gets a chuckle out of Jeff and Ken when they hear me say, "Just add a little software routine to replace...."

Unless you've spent the last 15 years under a rock, you've noticed that technology has evolved a couple orders of magnitude. The simple component-cost tradeoff between using a hardware or software solution isn't as simple as it once was. Today, the actual component cost might be secondary to the costs of developing the solution. Everything isn't as cut and dried as it used to be.

The same is also true of how we choose to cover embedded control designs within the pages of *INK*. Just like it would be remiss of a communications magazine not to acknowledge the significance of cellular technology, we would be ignoring the obvious not to increase our coverage of PC-based embedded applications.

The reason for the explosion isn't any conspiracy among programmers. The answer is much simpler. Every engineer has a PC; it's only natural to use it.

More and more frequently, the swiftness and cost of development is the primary issue. If asked to quickly produce a I-IO-kHz sweep frequency generator, hardware people might naturally gravitate toward a couple of hardware oscillators. Software types would obviously synthesize it using interrupts and counter/timers. Whether they use a PIC or a PC for the latter depends mostly on how management views delivery and production cost issues. Did I ever tell you about the engineer who was given a day to come up with a circuit to capture a 2-KB serial transmission and store it? He used a PC/104 system with ProComm and stored it to flash. The fact that there might also be a $5 hardware solution (if you have a lot of software-development time available) was considered irrelevant if it held up delivering the $1.25 million MRI machine for even a day.

Last month, *INK* introduced a new design contest specifically focused on the embedded PC. Probably the most significant difference between this contest and ones we've had before is that there are $10,750 in cash prizes. Even a third-place finish gets you $2000.

Unlike previous contests, we've decided to make this an embedded-PC contest. Picking an 80x86 processor (or any of its variants) as the target platform provides a more even competition. Winning is a matter of design finesse, not complexity. By having everyone use the same core technology, judges can focus on the merit and scholarship of an entry rather than dealing with the appropriateness of processor architecture.

The reason this contest has such significant awards is **only** due to the support and contributions from the sponsors. They recognize the significant role Circuit *Cellar INK*'s readers are making in the development of computer technology. It is their hope that, in the process of considering how to take part in this contest, you'll review their products and perhaps even incorporate them into a winning entry.

You don't have to physically build anything to win. Just describe and document the specifics in a way that would enable someone who had that task to proceed along the proper direction. You don't have to write the actual code, either. You only have to describe the logical process and flow required to do it.

This years design contest is a win-win situation. The level of support and commitment invested by contest sponsors is a declaration of their belief in you as a designer. Successful product designs require efficient engineering. The goal of this magazine, as always, is to document and promote such accomplishment.

Steve

steve.ciarcia@circellar.com