EMBEDDED PC SPECIAL BONUS SECTION: EMBEDDED PC

CIRCUIT CELLAR INK®

THE COMPUTER APPLICATIONS JOURNAL

#73 AUGUST 1996

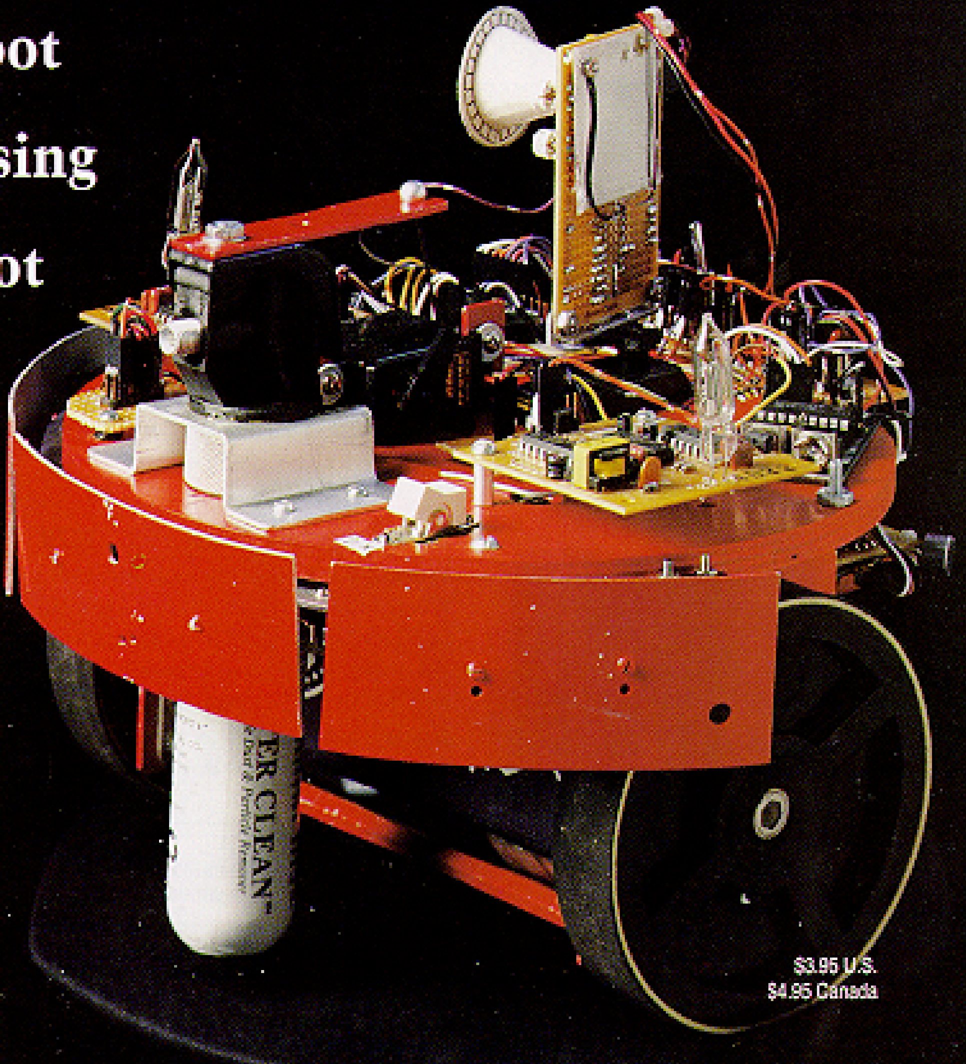
ROBOTICS

Marv the Robot

Obstacle Sensing

Modular Robot Controllers

EPC: $500 Internet Browser



08

0 74470 75349 0

$3.95 U.S.
$4.95 Canada

TASK MANAGER

Luxury to Utility

More and more, the image of the robot in today's society is shifting from the metal-clad humanoid butler to the special-purpose, esoteric-looking device that automates some repetitive or dangerous task. People are used to seeing robots assembling cars or other objects, wandering the halls of hospitals delivering food trays, and removing and defusing bombs for police.

Nowhere is this shift from the luxurious to the practical more evident than in the annual Firefighting Home Robot Contest. More than just a micromouse competition, the contest puts a robot in a mock house. When the "smoke detector" sounds, this robot must find the fire-while avoiding "furniture"-and extinguish it in the least amount of time. Imagine the savings in lives and property if we all had one.

In last year's Robotics issue (*INK* 63), we featured one of the 1995 contest winners. It was such a hit that we're bringing you one of this year's Marv is featured both on our cover and in a full article. The different approaches to solving a common problem are fascinating.

Before Marv, though, our first feature article surveys some of the more popular methods enabling robots to sense obstacles. How many of next year's contest entries will use one of these methods to avoid furniture?

Next, we zero in on much smaller robots-Stiquitos-and find out how researchers are trying to study insect colonies with the help of these tiny six-legged robots.

In our final feature, we continue with Part 2 of our look at in-circuit emulators by concentrating on monitor-based debugging and true in-circuit emulators.

In this month's columns, Ed starts a three-part series on zero-beat tuning, Jeff monitors power usage in his house, and Tom does the spring show tour.

Moving on to Embedded PC, we've all heard reports in the popular press about the under-$500 Internet appliance. Brad Reed of Radisys decided to see whether it could be done using off-the-shelf technology available today. Check out what he found.

Next, Part 2 of ourdiscussion of the WinLight embedded operating system shows what's involved in developing a sample application.

PC/104 Quarter explores the idea of precision timekeeping using components based on the PC/104 bus.

Finally, we apply multiaxis stepper motors to a baby *xy* table and find out how powerful one motor controller can be.

editor@circellar.com

CIRCUIT CELLAR INK®

THE COMPUTER APPLICATIONS JOURNAL

FOUNDER/EDITORIAL DIRECTOR
Steve Ciarcia

EDITOR-IN-CHIEF
Ken Davidson

MANAGING EDITOR
Janice Marinelli

TECHNICAL EDITOR
Elizabeth Laurençot

ENGINEERING STAFF
Jeff Bachiochi & Ed Nisley

WEST COAST EDITOR
Tom Cantrell

CONTRIBUTING EDITORS
Rick Lehrbaum
Fred Eady

NEW PRODUCTS EDITOR
Harv Weiner

ART DIRECTOR
Lisa Ferry

PRODUCTION STAFF
John Gorsky
James Soussounis

CONTRIBUTORS:
Jon Elson
Tim McDonough
Frank Kuechmann
Pellervo Kaskinen

PUBLISHER
Daniel Rodrigues

PUBLISHER'S ASSISTANT
Sue Hodge

CIRCULATION MANAGER
Rose Mansella

CIRCULATION ASSISTANT
Barbara Maleski

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. Second class postage 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 US. 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 photography by 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. 300/1200/2400/9600/14.4k bps, 6 bits, no panty, 1 stop bit, (860) 871-1988; 2400/9600 bps Courier HST, (860) 871-0549. World Wide Web: http://www.circellar com/

EMBEDDED PC

See Pages 63–92 for our Special Bonus Section

INSIDE ISSUE 73

READER I/O

INTERNATIONAL FLAVO(U)RS

I always read Steve's editorial and usually understand what you mean, but I was confused by your reference to Tide (INK 69). What's a box of Tide?

Non-U.S. readers have many difficulties understanding jokes and proverbs involving brand or personal names in them.

For example, until 1992 no sane person over here knew who Ross Perot was. In some magazines (not yours), this name kept popping up to indicate something. At first, I had the impression he was some kind of stand-up comedian. My opinion only changed when his name came up for president. Now, I have a hunch of what was being hinted, but I'll never be sure. Some parts of your culture I'll never understand, without migrating.

When publishing an international magazine (and a very good one-1 know of no better], try to address the international public.

Jan Verhoeven
Tilburg, The Netherlands

Humor is certainly the most difficult aspect of interacting with another culture. You're absolutely right. It's often laden with idioms as well as political and cultural insinuations.

However, take your Ross Perot reference. Although you didn't know who he was, you certainly got the gist of the humor right. No sane person over here knew who he was either.

Steve's humor is very much a part of his culture as an American and an engineer. Take Steve's oft-quoted reference to "my favorite programming language is solder." Of course, there's no such thing as the soldering language, but everyone knew what it meant, finds it funny, and now it's probably the most frequently quoted line attributed to Steve. It's his trademark.

And, I (a good stalwart Canadian) assure you migrating won't help. Cultural humor and jokes are difficult because they're based on old TV programs, commercials, childhood school games, and so on. Ian, we've already missed too much. We can't catch up. Let's just enjoy what we do *get.*

Janice Marinelli

SETTING THE RECORD STRAIGHT

Recently, you published "Autorouter and Board Layout Software Tool Analysis" (*INK* 70). Our product, EdtCAD, was mentioned in an unfavorable light, so I want to set the record straight. After speaking with the author, I discovered the article was written with a very old version of the software.

The comment that the system was hard to use has some validity considering the version the author tried. Realizing that the major bottleneck of the system was the human interface, Electronic Design Tools rewrote the total system about three years ago.

He also complained that the autorouter didn't completely route the demo board, but this was intentional. Since we know that a demo *can* do anything we want perfectly, our tutorial explains that the autorouter parameters were purposely set so the route would *not* be 100%. This enables the user to evaluate the ease and capability of modifying the routing of an autorouted board to obtain a complete layout.

As the article stated, EdtCAD is a complete soup-to-nuts EDA system that includes schematic capture, board layout, and manufacturing output. But, many additional subsystems are available that may be seamlessly integrated into the basic system with the file conversion and interface chores automatically handled by the Project Manager.

Michael Carpenter
President, Electronic Design Tools, Inc.

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 offices at (860) 875-2199.
Fax: All faxes may be sent to (860) 872-2204.
BBS: All of our editors and regular authors frequent the Circuit Cellar BBS and are available to answer questions. Call (860) 871-1988 with your modem (300–14.4k 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 **http://www.circellar.com/**.
FTP: Files are available at **ftp://ftp.circellar.com/pub/circellar/**.

NEW PRODUCT NEWS

Edited by Harv Weiner

EPROM EMULATOR

The E4 EPROM Emulator emulates all devices from 8 to 512 KB (27C-64–27C040) with 100-ns access time. Large-word-size applications (i.e., 16, 24, and 32+ bits) are accommodated by chaining together up to eight emulators. The chaining system also facilitates the use of EEPROM banks spread throughout a large address system.

A data-retention circuit maintains memory integrity while target power is cycled on and off. Real-time error checking and correction while downloading ensures data integrity without long upload-and-compare operations. Downloading through any of the PC printer ports is straightforward and fast. A 64-KB EPROM downloads in 2 s.

The E4 host software automates address calculations and file locations within the target address space. The download control software operates in both full-screen and command-line modes. Downloading from within the control software or in command-line mode enables you to easily integrate downloaded data into any debugging cycle. All popular data formats are supported (e.g., Intel hex, binary, and Motorola "S"). Different formats and word sizes are easily mixed.

The E4 EPROM Emulator package includes the E4 emulator, 8' download cable, printer port adapter, E4 software, and a user's guide. The unit sells for $249 and is backed by a one-year warranty and free software upgrades. The El EPROM Emulator, which emulates devices from 8 to 128 KB, is available for $199.

Scanlon Design, Inc.
5224 Blowers St. • Halifax, NS • Canada B3J 1 J7
(902) 425-3938 • Fax: (902) 425-4098
http://www.isisnet.com/oceana/SDI/ #500

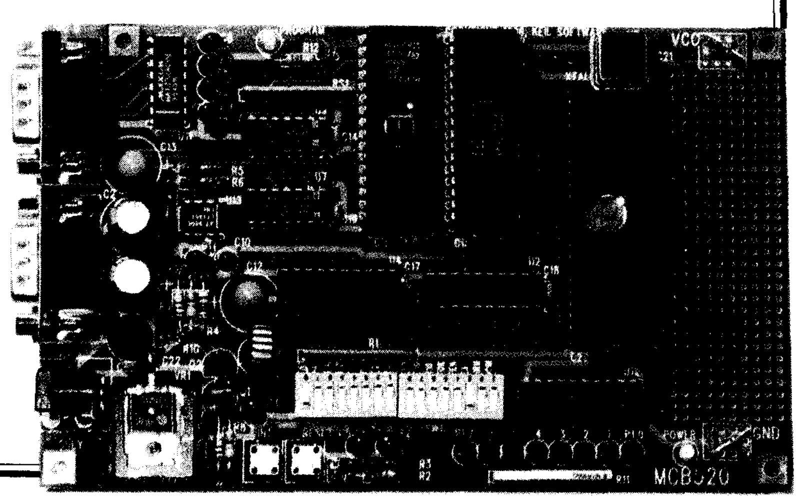
MICROCONTROLLER STARTER KIT

The Keil 520 Starter Kit is based on the Dallas Semiconductor DS320/520 microcontroller and the Keil MCB520 evaluation board. The kit is a great way to get started with the '520. It includes a limited set of development tools and an evaluation board. The evaluation board enables the user to become familiar with the different operating modes of the DS520. (Any 40-pin 8051-compatible device may also be used with this board.)

The MCB520 includes onboard bank switching, flash-memory support, RAM banking, simple expansion, and more. It supports up to 128 KB of RAM, 512 KB of EPROM, and 256 KB of flash memory. Onboard RS-232 drivers for both on-chip serial ports are included, and all chip signals are available via a 40-pin connector. A reset button and user-defined push button are onboard, and a prototyping area is provided.

The MCB520 is supplied with a 2-KB limited version of the Keil 8051 developer's kit. Sample applications are included. The board also has the Keil 8051 monitor (MON51) programmed in EPROM. The monitor communicates with the MON5 1 DOS terminal program (MO N 5 1. EXE) or dScope for Windows. Using dScope for Windows debugger or simulator, a user can quickly download and test a program. Both terminal programs communicate with the 8051 monitor through a PC serial port.

System requirements include an IBM PC/AT, PS/2, '386, or higher with one serial port and Windows 3.1, Windows 3.11, or Windows 95.



Keil Software
16990 Dallas Pkwy., Ste. 120
Dallas, TX 75248
(408) 229-9968
Fax: (408) 229-8303
http://www.keil.com/ • sales@keil.com #501

NEW PRODUCT NEWS

SYNTHESIZED SIGNAL GENERATOR

The SG-100 Synthesized Signal Generator is the latest in a new class of signal generators based on Direct Digital Synthesis (DDS) technology. A digital signal processor has direct digital control over every aspect of the DDS system. Frequency, phase, level, and the I and Q rails are controlled digitally, resulting in clean, precise, modulated waveforms. The combination of DSP and DDS technology provides signal-analysis functions which process and analyze an externally applied signal.

The SG-100 delivers fully synthesized DC to 20-MHz modulated or unmodulated waveforms with 0.1 -Hz frequency resolution. The basic unit includes linear and log sweep, AM, FM, PM, SSB, BPSK, FSK, burst, DTMF generation, DTMF detection, and power-level measurement. For each modulation mode, the user selects a modulating waveform that is either internally generated or externally supplied.

Flash memory holds the SG-100's operating software, so new functions can be added while the unit is in-field. User-friendly features include a large, easy-to-read illuminated LCD display. The user can see all modulation parameters simultaneously. A full numeric keypad and encoder provide direct editing of each parameter without confusing submenus. RS-232 remote control with programmable baud rate is included.

The SG-100 also features a DC offset capability, TTL/CMOS sync output in all modes, external logic input for gating of output signal, and triggering and configuration save and restore. The unit measures 5.1" x **9.3" x 10.2"** and weighs -3.5 lbs. The SG-100 sells for $795.

Telulex, Inc.
2455 Old Middlefield Way S. • Mountain View, CA 94043
(415) 938-0240 • Fax: (415) **938-0241**
http://www.telulex.com/ • sales@telulex.com **#502**

PORTABLE SYSTEM CONTROLLER

The DS1670 Portable System Controller integrates four functions—timekeeping, CPU monitoring, A/D conversion, and nonvolatile RAM control-into a single device. It reduces component count, power requirements, and the heat generated by components in space-conscious portable systems such as mobile phones, personal digital assistants, and any handheld, portable product requiring a timekeeper.

To further reduce component count, the DS1670 has a CPU supervisor that monitors the microprocessor and resets the system when the processor goes out of control or when the supply voltage is unstable or exceeds limits. The chip is a low-power device, drawing less than 500 nA in backup mode.

The DS 1670 provides automatic backup and write protection for external SRAM. A real-time clock provides seconds, minutes, hours, day, date, month, and year information with leap-year compensation. It also provides an alarm interrupt.

The chip provides a 3-channel, 8-bit, monotonic A/D converter that uses a successive-approximation technique to convert the analog signal into a digital code. Communication with the DS1670 is accomplished through a simple 3-wire interface.

The DS1670 is available in 20-pin TSSOP and SOIC packages. The TSSOP sells for $5.20 in quantity.

Dallas Semiconductor
4401 S. **Beltwood** Pkwy.
Dallas, TX 75244-3219
(214) 450-0448
Fax: (214) 450-3715
http://www.dalsemi.com/

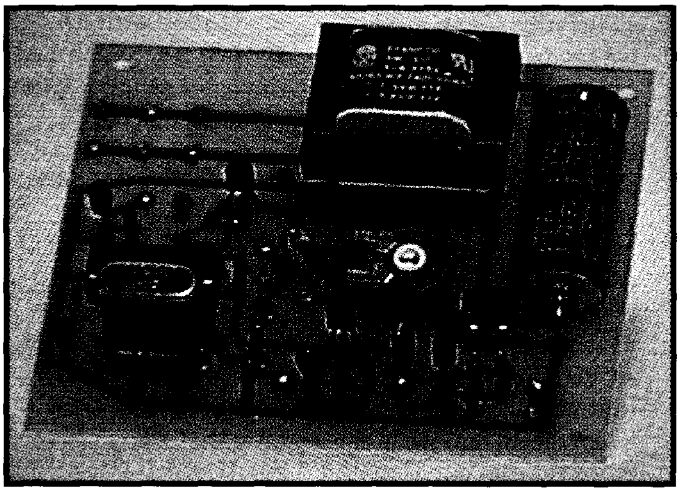
#503

NEW PRODUCT NEWS

OVER- /UNDER-VOLTAGE PROTECTOR

The **SPPC-1** Smart Protector Type 1 PC board controls an off-card solid-state relay (not supplied) to disconnect a load if the AC power-line voltage falls below or exceeds 100-130 V. The power available for the controlled relay is 5 VDC at 6 mA maximum, so a solid-state relay is required. Load current depends on the relay rating.

A Microchip PIC16C71 microprocessor, powered by a rechargeable NiCd battery, is programmed to monitor the AC power-line voltage. If the voltage goes outside the 100-130-V limits, the relay opens and the load disconnects. The circuit automatically resets itself and reconnects the load after 90 s when the line voltage returns within limits. If a power outage occurs, the microprocessor enters sleep mode to conserve battery power, but it continues to monitor the AC line. When power returns; it automatically resets. An onboard circuit trickle charges the NiCd battery.



A built-in test circuit simulates an out-of-limits line voltage with a single-pole, normally open push-button switch (not supplied). The transformer-isolated PC board measures 4” x 3.38” and features eyelets on the offboard wiring pads for extra strength.

The SPPC-1 sells for $35 plus shipping.

TDL Electronics
5260 Cochise Tr. • Las **Cruces,** NM 88012-9736
(505) 382-8175 • Fax: (505) 382-8810 • **rtipton@zianet.com**

#504

LOWDROPOUTREGULATOR

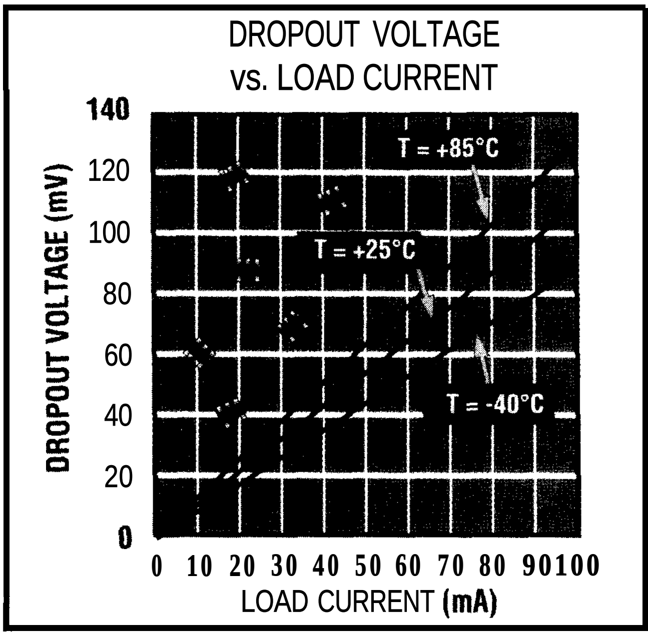
The **MAX8863T/S** and **MAX8864T/S** low-dropout, low-quiescent-current linear regulators are designed primarily for battery-powered applications. The devices feature dual-mode operation, supplying either an adjustable 1.25-5.5-V output or a preselected output for load currents of 100 mA (3.175 V for “T” versions or 2.85 V for “S” versions).

The devices consist of a 1.25-V reference, error amplifier, MOSFET driver, P-channel pass transistor, dual-mode comparator, and internal-feedback voltage divider. The PMOS pass transistor allows the low 100-µA supply current to remain independent of the load, making these devices ideal for portable equipment such as cellular phones and modems.

Other features include low-power shutdown, short-circuit protection, thermal shutdown protection, and reverse battery protection. The MAX8864 also includes an autodischarge function, which actively discharges the output voltage to ground when the device is placed in shutdown mode.

These devices are available in a miniature 5-pin SOT-23 package and are screened for the extended industrial (-40°C to +85°C) temperature range. Prices start at $0.75 in quantity.

Maxim Integrated Products
120 San Gabriel Dr.
Sunnyvale, CA 94086
(408) 737-7600 • Fax: (408) 737-7194
http://www.maxim-ic.com/ **#505**



Sensing Obstacles with Mobile Robots

FEATURE ARTICLE

Chuck McManis

Critical to mobility is knowing where you are. Chuck takes an in-depth look at three sensing technologies: switch bumpers, infrared light, and ultrasonics. He closes with a glimpse at radar and stereoscopic vision.

Mobile robots need to sense obstacles. Why?...primarily to avoid being damaged or causing damage to their environment.

When looking at obstacle detection and avoidance systems, you need to consider common design parameters like cost, range requirements, sensor processing requirements, and characteristic obstacles.

One sensor rarely meets all requirements. Typically, robots have several sensors with overlapping capabilities.

Through sensor fusion, the robot combines multiple sensor feedback into a single internal picture. When picking sensors, you need to be aware of their strengths and weaknesses. The robot's picture of the world should have no major gaps.

Let's take a look.

BUMPER SWITCHES

Bumper switches simply replace a large collision with a smaller one. They're designed to be more mechanically compliant than the robot. A bumper switch translates a mechanical contact with an electrical closure.

The ideal bumper switch requires nearly zero pressure to activate and is infinitely compliant. In practice, however, good results are common with microswitches or "feeler" switches. Implement them by attaching a feeler and electrical leads to the switch, as shown in Figure 1a.

However, for greater flexibility, implement a bumper switch using a conductive wire within a conductive tube. You build this bumper type by inserting a spring wire into an insulating tube and then putting the wire and tube into an out-conducting tube (see Figure 1b).

The inner tube guides the wire, keeping it from touching the conductive tube. When the wire is deflected by encountering an obstacle, it completes the circuit.

Since a sensor is inherently a binary switch, connecting it to a digital input port is simple. As Figure 1c illustrates, I use a pull-up resistor on the digital input port and the bumper switch connects to ground. If my robot's chassis is at the same ground potential as my circuit board, a single wire joins the computer to the bumper switch.

Bumper switches are extremely cost effective. Cost is under $5 per sensor, while a self-manufactured sensor costs no more than $1.

They are easily processed in C or assembly. Here's a typical C statement for reading a digital I/O port:

```
if ((*portAddress & 0x1) == 0)
  backup();
```

Here, I assume the switch connects to ground (e.g., a 0 value means that the switch is closed) and that the I/O port is readable as a memory location.

You can also process bumper switches by tying them into interrupt lines on the robot's CPU and servicing the interrupt when the bumper collides with an obstacle.

The bumper switch detects walls, vertical drops (e.g., the top of the stairs), and objects with sufficient weight that they are not moved by collision with the bumper.

They are particularly useful when the robot knows its environment (e.g., a micromouse maze) and mostly needs to confirm its position. Since they are inherently "near" detectors, they aren't useful on high-speed robots.

Bumper switches are generally limited to acceleration influences. When the robot accelerates, the bumper's mass has a specific inertia, which can cause a false trigger initially.

ULTRASONICS

Ultrasonic detectors operate using reflected sound waves. An ultrasonic detector consists functionally of a gatable sound source, a transmission transducer, a reception transducer, and a sound detector.

Ultrasonic-sensor theory is straightforward. A transmitter emits a sound pulse. The receiver is turned on and waits for an echo from an obstacle.

The time between the transmission and echo is multiplied by the speed of sound and divided by two. This yields the distance to the obstacle. Since the speed of sound at sea level is 1130 ft./s, this time is easily measurable by even relatively slow 8-bit microprocessors.

This sensor's accuracy and reliability are affected by design and environmental parameters. Consider first that the speed of sound varies based on atmospheric pressure and temperature.

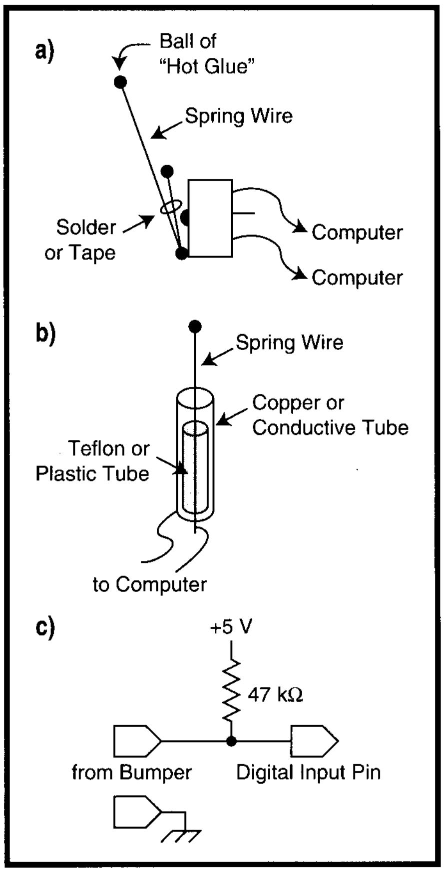


Figure 1—*Bumper switches are the easiest sensors to integrate into a robot. They are robust and reliable. You can build them out of microswitches (a), or they can be made easily from any conducting material (b). They share a common interface to the robot (c).*

This variation, however, is manageable. Robots that don't change altitude by more than a couple of hundred feet or work under variable temperatures simply calibrate initial conditions and work from there.

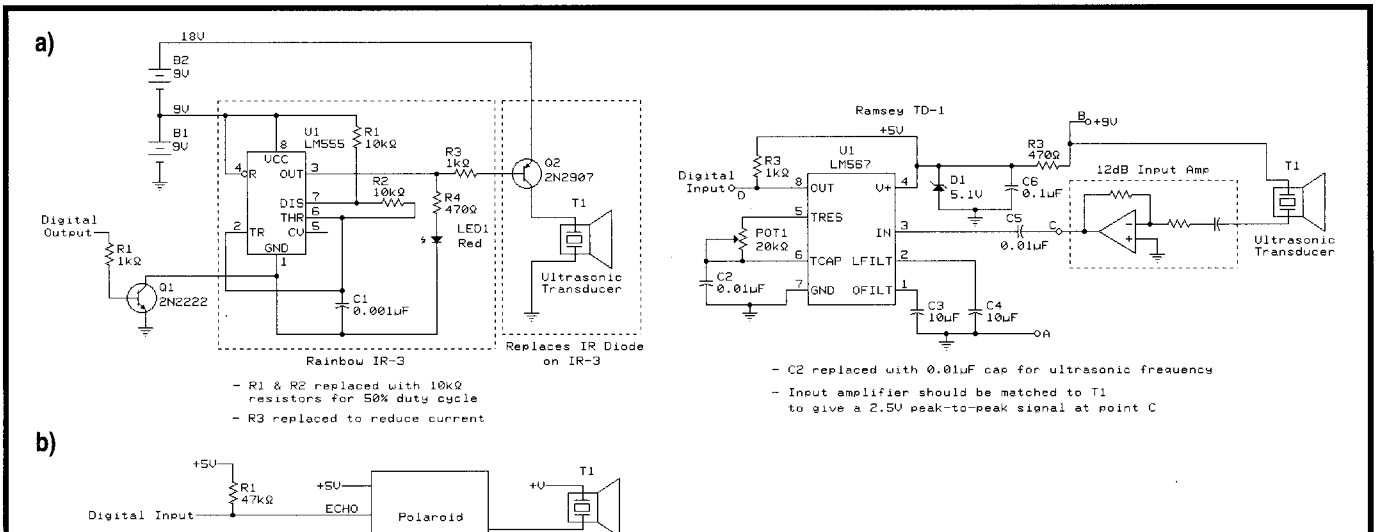


Figure 2—*Ultrasonic sensors are more complicated but can "see" farther. They can a) be built from components or kits or b) there are more sophisticated systems available as packaged modules.*

The design and implementation of an ultrasonic sensor affects the accuracy and repeatability of its results. Two examples are shown in Figure 2.

The first is an inexpensive sensor built from available kits (see Sources). The second is a prepackaged Polaroid solution sold as a unit.

The component example shows how the sensor could be built. To create an ultrasonic detector from components, I modified two kits: an IR transmitter from Electronic Rainbow and a tone detector from Ramsey Electronics.

The tone-generator kit's schematic is shown in Figure 2. It's a simple LM555 timer operating in the astable mode. The modifications consist of putting a transistorized power switch on the board and adding a buffer amplifier to isolate the piezoelectric transducer from the '555.

Figure 3 shows the tone-detector kit's schematic. The ultrasonic transducer and amplifier provide input for the tone detector.

For a more integrated design, use an inexpensive DSP to generate the tone. A Fourier analysis on the receiver's input detects the received tone.

The Polaroid product saves on component cost by combining the transmitter and receiver transducers in a single unit.

However, short distances are difficult to detect. The transducer must stop vibrating with the transmission tone before it can be used to detect the echo.

Ultrasonics can be expensive. Prepackaged modules cost $50 when ordered directly [minimum 2 units) and over $70 from third parties. The component implementation is less than $20, and if you breadboard the circuits, you save even more.

Processing an ultrasonic sensor takes more time than processing a simple bumper since it has ping, detect, and analysis phases. A separate processor could monitor the sensor, but that adds complexity and cost.

To me, the most difficult part of processing an ultrasonic sensor is knowing when to quit. Specifically, when do you decide an echo won't return and therefore stop measuring?

Listing 1-A *simple ultrasonic driver function that is processor dependent takes longest when there are no obstacles to detect.*

```
#define MAX-DISTANCE 30000

float ping0

  int ticks = 0;
  *port = 1;            /* Turn on tone generator */
  while (ticks < MAX-DISTANCE) {
      if ((*port & 1) == 0)
        break;          /* Echo was detected */
      ticks++:
  }
  return (ticks * CALIBRATION_CONSTANT);
}
```

Assume you aren't interested in items more than 20' away. It takes about 40 ms (20' each way) for an echo. By 41 ms, it's detected nothing.

Depending on the transducer's configuration, its window ranges 10-20". To scan the forward area of a robot (-45" to +45° from straight ahead), you may need up to 10 scans of 0.4 s each, if done serially. Of course, you can scan in parallel with more transducers.

For local processing, a C function like that in Listing 1 can check the sensor. This loop is processor-timing dependent. (It won't work on interrupt-driven systems!)

The code sends the tone and waits for an echo. When the echo returns or the loop counts to the maximum time for an echo to return, the loop exits.

The resulting ticks are multiplied by a floating-point constant gleaned from two known distances. The result is returned in inches, feet, or meters.

For processor-independent timing, you must access a hardware counter/timer. In Listing 2, timer0 is a pointer to a 32-bit hardware-timer circuit.

Finally, replace **CALIBRATION_CONSTANT** with a call to a function that samples temperature and pressure sensors and returns a constant that's accurate for the current conditions.

The main advantage to ultrasonics is that they can return absolute distance. With additional work, they'll do this for multiple obstacles. They detect a wide variety of materials and are not fooled by colors like infrared sensors are. Though they're good for mobile robots, they are more expensive.

The two biggest factors with ultrasonics are material and reflections. Ultrasonic sensors can completely miss a door opened just enough for the sound to be reflected out of the room.

Reflected energy is entirely derived from surfaces nearly perpendicular to the sensor. However, ultrasonics see narrow surfaces other sensors miss.

Reflective-surface material degrades response. Noise-canceling materials—such as soft drapes-are only moderately visible. On the other hand, if the sensor detects an echo, the distance is most often calculated correctly.

Listing 2-A more *accurate ultrasonic driver uses hardware timers to measure the flight time of an ultrasonic pulse.*

```
float pint0

  long tl, t2;
  int i = 0;
  tl = *timer0;          /* Read the timer */
  *port = 1;             /* Turn on the tone */
  while (i++ < MAX-DISTANCE)
     if ((*port & 1) != 0
       break:
  t2 = *timer0;          /* Read the timer again */
  return (abs(t2-tl) * CALIBRATION_CONSTANT);
```

INFRARED

Infrared "break" sensors require an obstacle to pass between the transmitter and detector. However, a Sharp single-module IR detector can also detect *reflected* light.

IR sensors work a lot like ultrasonic sensors. However, instead of measuring flight time (which is very, very fast], the presence or absence of reflected light indicates an obstacle.

The detector consists of a detector, amplifier, and band-pass filter (see Figure 3a). When light modulates at the center frequency of the band-pass filter, its presence is detected and amplified. At an internally set threshold, the output pin goes active low.

Variations come from changes in the transmitter. The modules are sensitive to IR light modulated at their filter's center frequency. Light intensity varies with a surface's reflective properties. A wider transmitted beam and a band-pass filter reduce intensity.

The band-pass filter has two components: the frequency and the modulation duty cycle. The ideal modulation source is a sine wave at the center frequency of the band-pass filter.

For cost reasons, most people use a square wave. The closer the wave is to a 50% duty cycle, the more energy is present in the frequency fundamental.

Figure 3b shows an implementation of an IR sensor. The tone generator can be the same as that of the ultrasonic example (the frequencies are the same) or a microprocessor's pin. The detector feeds directly into the input pin.

IR sensors are practical. IR detector modules now sell for $3-6, and an IR LED goes for $0.25. Total cost is similar to bumper switches built from premanufactured microswitches.

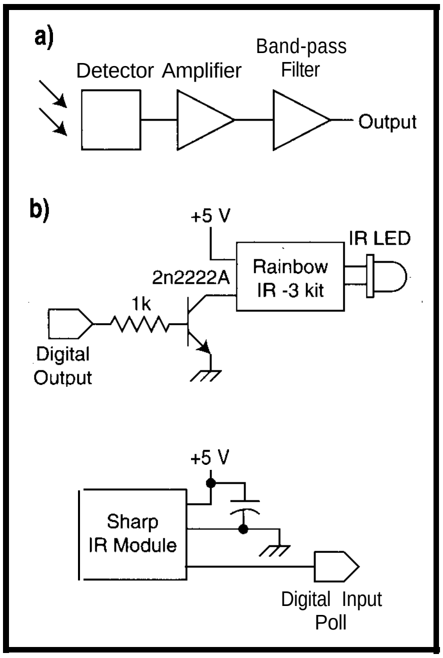


Figure 3—The infrared detector is designed to amplify and defect infrared signals from a TV remote control (a). By using a separate transmitter and defector, we can turn IR signals info a bumper system (b).

IR's advantage is its lack of contact, but it misses obstacles and sees a large number of false positives. Listing 3 offers the simplest processing loop with a hardware tone generator.

This loop turns on the modulated IR LED and then cycles for a while, collecting samples from the IR detector. If the number of true samples exceeds a certain threshold (i.e., 75%), the detect status is returned as true.

The loop is required because the high number of IR sources commonly cause spurious detections. Big sources of interference are fluorescent lights and TV remote controls.

A better detection routine checks for both presence and absence of the modulated light (see Listing 4).

This routine's advantage is that it checks to see that the reflected light

Listing 3-This function reads an infrared defector with a simple software low-pass filter to eliminate spurious interference.

```
int irdetect()

  int i, detect = 0;
  *port = 1;                    /* Turn on IR LED */
  for (i = 0: i < 100; i++)
    if ((*port & 1) == 0)       /* Detected LED */
      detect++:
 *port = 0;                     /* Turn off IR LED */
  return (detect > 75);
```

comes from the LED you are turning on, rather than an ambient source.

One cycle significantly enhances reliability--three to five on/off cycles is very robust. It also notes when you detect light during an off cycle and flags that as known interference.

IR bumpers are sensitive to ambient conditions and surface reflectivity. You can evaluate these sensors by having a CCD camera sensitive to IR light view an IR illuminated area.

You can check environmental targets by looking for changes in light level on a video monitor attached to the camera. If you detect the change in light level, the IR detector will too.

With standard high-intensity IR LEDs (i.e., 50-mA drive current), detection ranges 4-6'. You can intentionally degrade light-source intensity or modulate the detector's center frequency. If you do this in code, you detect long and short range from a single sensor.

Of the sensors I've examined in detail, IR sensors are the most susceptible to environment. A system that works indoors is blind outside.

Environments with dark or IR-absorbent surfaces give shorter detection ranges than those with highly reflective obstacles. And, some materials that appear opaque to the unaided eye are transparent to IR light. An IR-sensitive camera illustrates this.

'The new compact fluorescent bulbs give more interference. Although they eliminate flicker, their upper harmonics are often in the 36–44-kHz range, which affects most modules.

IR sensors can use visible light and not just IR light. The advantage to IR, however, is the availability of inexpensive single-package detection modules.

Photodetectors sensitive to red or green light also work. With these sensors, the light source modulates in a relatively low frequency [i.e., tens of kilohertz). This modulation is less common in naturally produced light.

COMPLEX FUTURE TECHNOLOGIES

Two technologies to watch for in the future are radar and stereo vision.

Recent developments at Lawrence Livermore National Lab promise to make radar more affordable. A micropower impulse radar provides on-chip radar for about the cost of two ultrasonic modules. Using radar and well-known landmarks, a robot can localize itself and determine a path to its goal.

On the vision front, the cost of DSP chips is being driven down by the modem market. Cameras are matchbook-size PCBs and pin-hole lenses.

Since the primary barrier to integration into mainstream robots is cost, these sensors seem headed to low-end applications within the next five years.

Listing 4—*This infrared driver function eliminates interference by monitoring both on and off slates of the IR transmitter.*

```
int irdetect()

  int i, detect1 = 0, detect2 = 0;

  *port = 1;    /* IR LED is on */
  for (i = 0; i < 100; i++)
    if ((*port & 1) == 1)
      detect1++;
  *port = 0:    /* IR LED is off */
  for (i = 0; i < 100; i++)
    if ((*port & 1) == 0)
      detect2++;
  return ((detect1 > 75) && (detect2 > 75));
```

CHOOSING YOUR SENSORS

The goal of any robotic sensor is to detect as many things as possible for the lowest possible cost.

That cost is more than financial. Processing overhead and ease of integrating sensor information is crucial, too.

The choices made by the robot designer and builder involve understanding what must be detected and when it must be detected for the robot to react appropriately. •J

Chuck McManis is the Director of System Software with FreeGate Corp. He is also the chairman of the Peninsula Homebrew Robotics Club. You may reach Chuck at cmcmanis@netcom.com.

SOURCES

Ultrasonic sensors and transducers
All Electronics Corp.
P.O. Box 567
Van Nuys, CA 91408-0567
(213) 380-8000
Fax: (818) 781-2653

Radar Technology
Artech House, Inc.
685 Canton St.
Norwood, MA 02062

Micromint Inc.
4 Park St.
Vernon, CT 06066
(860) 871-6170
Fax: (860) 872-2204

Polaroid Corp.
119 Windsor St., Ste. 2-B
Cambridge, MA 02139
(617) 577-4681
Fax: (617) 577-3213

IR detector modules
Sharp Microelectronics Group
5700 NW Pacific Rim Blvd., Ste. 20
Camas, WA 98607
(206) 834-2500
Fax: (206) 834-8903

IR-3
Electronic Rainbow Industries, Inc.
6227 Coffman Rd.
Indianapolis, IN 46268
(317) 291-7262
Fax: (317) 291-7269

TD-1
Ramsey Electronics
793 Canning Pkwy.
Victor, NY 14564
(716) 924-4560

Radar on a Chip
Lawrence Livermore National Lab
http://www-lasers.llnl.gov/lasers/idp/mir/mir.html

I R S

401 Very Useful
402 Moderately Useful
403 Not Useful

Designing Marvin

FEATURE ARTICLE

John Piccirillo

Building autonomous mobile robots is a challenging task. But, John succeeds with style. Listen up for the design and construction of Marvin, a winner in this year's Firefighting Home Robot Contest.

Designing and building autonomous, mobile robots is challenging, creative, frustrating, and yet satisfying. To me, the activity's appeal comes in the mix of electronics, mechanics, computers, and design strategy.

A contest is an excellent way to focus these skills on a well-defined and measurable goal. One such contest is the annual Firefighting Home Robot Contest held in Hartford, CT.

The objective is to build a computer-controlled robot that navigates through a mock house, locates a fire (a lit candle), and extinguishes the flame in the shortest time. My robot Marvin (see cover photo) won second place.

Here, I'll describe Marvin's design and construction and some design choices made by other contestants.

CONTEST RULES

The rules provide the rationale and restraints for Marvin's design. These rules specify the approximate dimensions and layout of the robot house shown in Figure 1.

The floor of the structure is black, the white walls are 13" high, and all doorways and halls are 18" wide. A white stripe is painted on the floor across each doorway.

After the robot is placed in the home circle (H), a contest judge places a candle in one of the rooms (A, B, C, or D) at random. The robot starts and attempts to find and extinguish the candle. The sum of the best two out of three times determines a contestant's score. The lowest score wins.

To up the ante, there are bonus modes of operation and penalties for bumping walls or the candle. Your score is the time (in seconds) it takes to find and extinguish the candle.

Robots starting on a 3.5-kHz buzzer tone [i.e., a smoke alarm) win a 5% time reduction, while robots returning to H after the candle is extinguished win a 10% reduction. Autonomous (as opposed to tethered) operation is rewarded with a 30% reduction.

Operating in a "furniture" mode gets you a 90% reduction. In the furniture mode, yellow cylinders measuring 4½" diameter x 12" high are placed in each room. Time score reductions can be compounded.

I'm competitive, so I designed Marv to enter all modes simultaneously.

TECHNICAL APPROACHES

Before getting into Marv's design, let's look at approaches by other contestants. Their efforts furnish a valuable baseline of techniques that do and don't work, and those that are promising but need improvement. Since this was the third annual contest, information was available from the previous two events, including videotapes.

From the tapes, descriptions I received from other contestants, and my own observations, I knew Marv needed three basic subsystems: navigation, candle detection, and flame extinction. I also needed a microcontroller to control operations, a portable power supply for autonomy, and a sound switch to detect the buzzer.

A variety of methods extinguish the candle. The most popular are fans, but there are also pumps or pressurized cylinders that squirted water, dry-chemical extinguishers, snuffers that flip baking soda on the candle, and blown-up balloons that steer into the flame.

Fans require getting close to the flame, which takes extra time and complicates the furniture mode. Robots that squirted water had problems with accuracy, range, and getting enough on the flame. I decided against all these methods.

Since the candle flame was to be 6-8" off the floor, the flame detector needed a field of view (FOV) broad enough to see the candle tall or short, near or far, and not be confused by other light sources in the room, including camera flashes.

Detectors for every spectral region were tried: UV photoelectric sensors, CdS photoresistors for the visible, near-IR phototransistors (about 1 µm), and pyroelectric detectors for the mid IR (8-14 µm).

Although near-IR phototransistors are the most popular, their advantage is dubious. Most of a candle's light—the yellow flame-comes from black-body radiation by hot carbon particles (soot) formed during combustion.

The color temperature of the flame is about the same as the filament in an incandescent bulb. Single detectors using visible or near-IR radiation compare their output to the threshold for ambient light. So, this method is risky.

The background from white walls isn't even. There are shadows, and flashbulbs and camera or camcorder IR rangefinders might interfere with the detector. Most troublesome for all detectors is the unknown background lighting-incandescent, fluorescent, halogen, high-pressure sodium, or what? The rules don't specify.

Navigation is the most difficult task. Most of the robots that failed (about 50%) got stuck. Getting to the candle is also the most time-consuming job. Many approaches-some very unusual-were tried.

The layout of the robot house is known to ±1″ in the furnished dimensions. Some robots used this information to establish a fixed search path stored in computer memory. With incremental shaft encoders or stepper motors, they navigated the rooms.

The biggest problem seemed to be making accurate 90" turns. In several instances of augmented dead-reckoning, the robot navigated part way through the structure and then re-aligned itself to null out odometry or turning errors. Some alignment techniques physically contacted the walls, and some detected the white line on the doorway floors.

A few wall followers showed up, including an elegant model originally designed for a micromouse contest. Wall following was difficult in this year's contest because room D was detached from the rest of the structure.

In the first year, a very fast robot collided and spun off the walls, ricocheting from room to room. It put the candle out in 5 s! To accommodate the new bumping penalties, it collided with virtual walls detected by a pulsating IR detector.

Two robots opted to not turn. One changed direction by lifting the robot on a central disk, rotating the whole machine, and lowering it in its new orientation. Another used two sets of wheels mounted perpendicular to one another. A motor operated cams to change wheel sets.

IMPLEMENTING MARV

After reading the rules, considering what others had done, and thinking

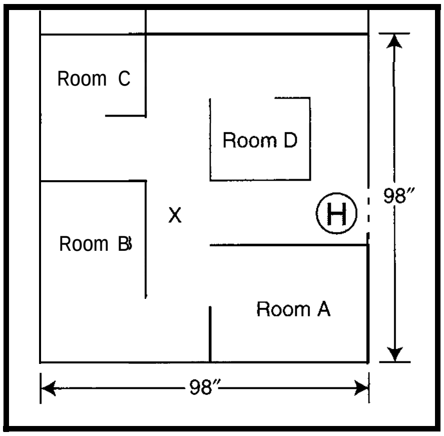


Figure 1--Robots start in the home circle (H) and try to find and extinguish a lit candle placed randomly in one of the rooms. The fastest two out of three trial runs determines the score.

about the problem, I had a general idea of what I wanted to do and how. Some ideas didn't work out, but I won't go into those. I'll describe the components Marv uses and go over the strategy of how it all works together.

Getting Marv to move at a reasonable speed through the house without getting lost or bumping into walls was the most difficult part of the design. I used stepper motors because they have accurate and repeatable positioning. Also, steppers don't require interrupt capability. Marv used the Parallax Basic Stamp II microcontroller which can be programmed in PBASIC.

The stepper motors were surplus items purchased from Herbach and Rademan. They are 5-V unipolar step pers that draw 1.5 A, have a holding torque of about 50 oz.-in., and a 7.5" step angle. The motor shaft was ¾″ long and ¼″ in diameter with a flat.

I used an L297 stepper-motor controller and a L702 quad Darlington switch for each motor (see Figure 2). They operated in half-step mode.

Stepper control is an art. This configuration doesn't optimize the steppers. The first-place robot used a $400 commercial stepper-driver board. It stepped circles around Marv.

With 4″-diameter wheels, each step moved Marv about ⅛″—accurate enough for moving a fixed distance, but barely adequate for turns. With a wheel base of 7½″, one step rotated Marv about 2" when pivoting around its center, and 1" when pivoting about a stationary wheel. A 1" turn error gives Marv adequate clearance for a 8′-long hall.

It took a long time to find wheels that mounted directly on the stepper-motor shaft. Marv's plastic wheels have a 3⅞″ diameter and ¾″ width.

I wrapped double-sided foam tape around the wheels and topped it with a layer of electrical rubber tape. These additions yield a flat, even surface with good traction.

The wheel attaches to the stepper shaft with a hex-holed sleeve. It has a ½″ outside diameter and a W-hex inside diameter. The sleeve is tapped for a set screw, which locks against the flat on the stepper shaft.

A hole drilled through the wheel hub and sleeve is tapped for a 4-40 machine screw. This holds the wheel securely in place.

The UV flame detector, a commercial unit sold by Hamamatsu for $75, consists of a UVtron tube and a driving circuit board. It worked perfectly and trouble-free from the beginning.

The UVtron tube converts UV photons to an electrical pulse which is amplified and conditioned by the circuitry. The unit puts out 5-V pulses that are 10 ms wide (changeable with external capacitance if desired).

The pulse rate is proportional to the intensity of detected UV light, with a saturation rate of about 20 Hz. Since it operates in the solar-blind region (185–260 nm), it doesn't see sunlight or light produced by sources with a glass envelope because glass absorbs UV.

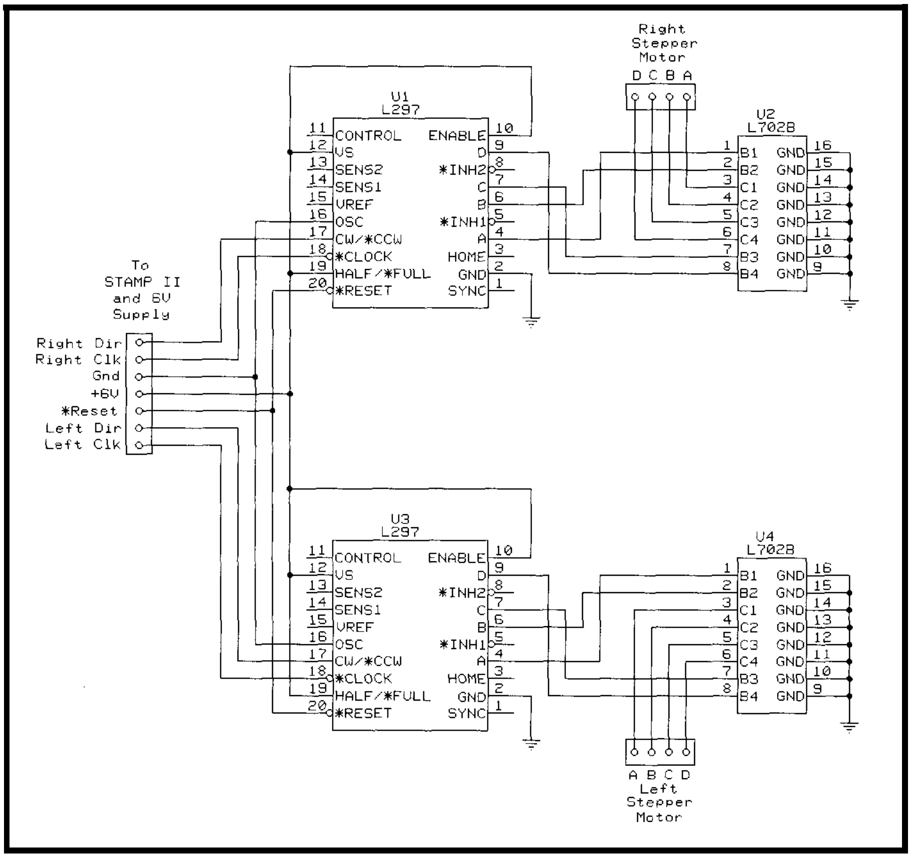


Figure 2-h the *dual stepper-motor driver circuit, microcontroller pulses to the clock input (CLK) step fhe motor. The polarity of the signal on the direction (DIR) input sets the rotation to clockwise or counterclockwise. Each motor is independently driven. The reset (RST) pin initializes fhe motors to their home position.*

In my limited experiments, it didn't see electronic flashes either. It sees a lamp if it uses a quartz envelope. The detector easily detects a candle flame at 15'.

Because the UV detector had a broad FOV and easily saw reflected light, another detector located the candle inside a room. The second flame detector was a lithium-tantalate pyroelectric detector. These detectors, commonly called passive infrared (PIR), detect people by their body-heat radiation in home-security systems.

Marv used an Eltec 442. A Fresnel lens mounted in front of the detector increased the sensitivity and narrowed the FOV. The detector, Fresnel lens, and a mounting kit were purchased from Acroname for $45.

The Eltec 442 has dual elements and an internal FET amplifier. Because of the dual element, the unit only detects differences in radiation received by both elements. So, it's ideal for scanning. Scanning too fast, however, results in an output that is only a few tenths of a volt in amplitude.

And, the output rides on a 2.5VDC level, which is not suitable for direct input to a microcontroller. Its sensitivity also wasn't high enough at the FOV edges. It needed to accommodate both 6" and 8" candles.

To solve these problems, a circuit amplified the PIR output and passed it through a window comparator, as shown in Figure 3. The op-amp gain is 9, which may be overkill. The detector also sees the warm candle after it goes out. The detector and its circuit board are mounted on a Futaba S148 servo.

I wasn't satisfied with many of the flame-extinction techniques. I wanted something that worked from a doorway and didn't require pinpoint aiming. I also wanted it to work close to furniture without interference.

I found a CO, dispenser-a lens duster-just the right size at 5" high x 1 ¼″ diameter. The tests I conducted were successful beyond expectations.

The CO, blast put out a candle in 1 s. In fact, it put out a row of candles 2' long from 4' [the size of the largest room) in a single puff. The blast also put out candles behind furniture. One CO, cylinder holds five 2-s discharges.

A condenser microphone detects the 3.5-kHz starting buzzer. The microphone output is amplified with a 741 op-amp preamplifier and a '386 audio amplifier. This feeds a 567 tone decoder tuned to 3.5 kHz.

A Stamp II single-board computer controls Marv's functions. The Stamp II has a 2-KB EEPROM for program and data storage, **16** I/O pins, and 32 bytes for variables.

The Stamp programs in PBASIC and is easy to use. Marv uses 15 of the I/O pins and most of the EEPROM. A Basic Stamp I (256-byte EEPROM and 8 I/O pins) controls the CO, servo, although it could be directed by the Stamp II.

To save space, I used the Stamp II module without its carrier board. It is a 24-pin DIP mounted on a piece of circuit board with 12-pin male headers on either side. The circuit board measures **1%” x** 1¾″. It attaches to the robot platform with double-sided tape.

Two battery supplies are used. A 6-V, 4.0-Ah gel cell powers the steppers and servos. A NiCd pack of 10 AA batteries powers all the electronics.

A 7805 regulator provides 5 V for the Stamps, PIR, and buzzer-detector

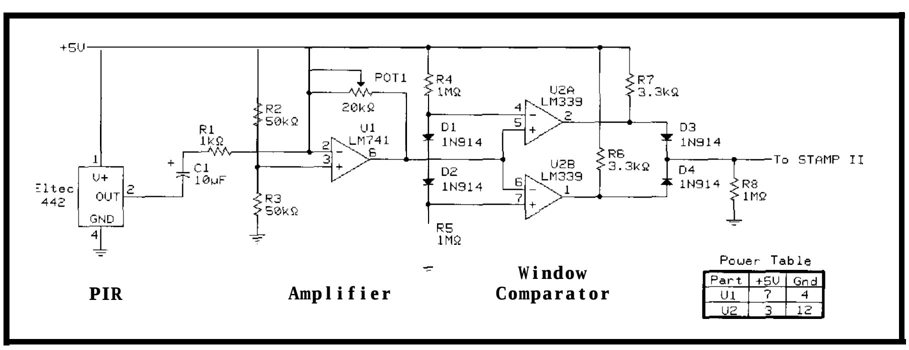


Figure 3—*In the PIR signal-conditioning circuit, detection pulses from the PIR sensor are amplified, thresholded, and pulled up to 5 V before being sent to the microcontroller.*

circuitry. The grounds of the two battery supplies connect together and to a layer of aluminum foil glued to the underside of the robot platform.

A few weeks before the contest, I decided Marv needed bumpers. It was a long shot that proved unnecessary.

A three-section front bumper wraps around the front half of the platform. The sections attach to microswitches used only if and when Marv enters far into a room looking for a candle.

I wanted a relatively small platform for Marv-the maximum is **12" x 12" x** 12"—to allow margin for error in traveling around the house. All of Marv's parts are mounted above and below a ⅜" plywood disc 8" in diameter.

Above the platform are two UV flame detectors, a PIR sensor, two Stamps, and a CO, extinguisher. There are also two servos (for scanning the PIR detector and discharging the CO, cylinder) and the power-distribution strips. Below the platform are the stepper-motor control board, the buzzer-detector board, the stepper motors and wheels, and the batteries.

The stepper motors attach to a U-shaped bracket of ⅛" aluminum bolted underneath the platform. The bracket is positioned in the middle so that Marv pivots around his center.

Two 6"-long aluminum standoffs join the open ends of the U bracket to keep it from flexing. A ball caster supports the rear of the robot. Marv's weight is evenly distributed, and the bottom of the hemispherical CO_2 cylinder, which clears the floor by about a ½", serves as a forward skid.

The 6-V gel cell just fits between the motors, the rear ball caster, and the CO, cylinder, and it rests on the standoffs. The NiCd battery pack rests on top of the gel cell. It's a compact arrangement, weighing 8% lbs.

STRATEGY

Early on, I decided on a dead-reckoning strategy since it takes less hardware, software, sensor sampling and computer processing, and-1 hoped—would be faster.

Marv follows a stored set of instructions and stepper-pulse counts to travel through the structure. The instructions aren't in a strict sequence. Branching occurs based on results from the UV and PIR detectors.

Once on, Marv starts on the buzzer tone and proceeds from point H to point X, immediately turns left, and goes to the doorway of room B. If the UV detector doesn't see a candle, Marv backs to the doorway of room A, then room D, and finally moves to room C.

All rooms but C are searched with only one turn. When the candle is found, Marv pivots to face the room and scans with the PIR detector. If the candle isn't found, he enters the room about **10"** and rescans.

This scan finds the candle if in the hidden corner. If it's still not found, he proceeds farther into the room.

During this move, Marv monitors his front bumpers. If a bump occurs, Marv backs up and searches again. If the candle isn't located, he assumes it's behind the furniture just bumped and discharges the CO,.

After a CO_2 blast, Marv checks to see if the candle is still lit. If it is, he completes the present scan, in case a false alarm occurred midway through.

If the candle is not found, he proceeds to the next step as before. Although it's not part of the contest, this feature enables Marv to put out more than one candle in a room.

TESTING

For testing, I built a replica of the house. This turned out to be important. I discovered and corrected many failures and tested alternate strategies.

I ran the course at least a hundred times before the contest. Since the contest environment is never exactly the same, I had special test programs.

In Marv's case, the house structure, overhead lighting, and white paint were different. All necessitated software adjustments before the contest.

CONTEST DAY

Of the 33 contestants, three extinguished the candle in all three runs. Six other contestants put the candle out twice, and five more put it out once.

Marv put the candle out all three times in furniture mode. He had an optimized search strategy and an efficient extinguisher, but he wasn't speedy. Heftier stepper motors and a better controller would make him more competitive.

Each year, the contest changes a little. People learn from the work of others, bringing new challenges.

Hey! It's not too early to begin for next year's contest! ■

John Piccirillo works on missile defense at Teledyne Brown Engineering. His interest in robotics was inspired by the work of MIT's Rodney Brooks. He may be reached at jpicciri@nebula.tbe.com.

SOURCES

Firefighting Home Robot Contest
http://shakti.trincoll.edu/~jhough/fire_robot/comp.html

Basic Stamp II
Parallax, Inc.
3805 Atherton Rd., Ste. 102
Rocklin, CA 95765
(916) 624-8333
Fax: (9 16) 624-8003

PIR Kit
Acroname, Inc.
P.O. Box 1894
Nederland, CO 80466
(303) 4150850
sensors@acroname.com

UV flame detector
Hamamatsu, Inc.
P.O. Box 6910
Bridgewater, NJ 08807
(908) 231-0960
Fax: (908) 231-1218

Unipolar Stepper Motors
Herbach and Rademan
P.O. Box 122
Bristol, PA 19007-0122
(215) 788-5583
Fax: (215) 788-9577

404 Very Useful
405 Moderately Useful
406 Not Useful

Modular Robot Controllers

Ingo describes some of the experiments for Stiquito II, a robot with Nitinol-wire-propelled legs. He looks specifically at the design strategies necessary for a controller system to be flexible enough for such robots.

FEATURE ARTICLE

Ingo Cyliax

When asked to design a robot controller for a Niti-nol-based Stiquito [1] robot several years ago, I was over-whelmed. The robot is small. It's quite a challenge to build a nontrivial con-troller small and light enough.

Early tests of Stiquito involved tethers connecting Nitinol-based leg actuators to an external PC-based controller so it could walk. We wanted to build many of these inexpensive ant-like robots to study colony behav-ior. Clearly, a tethered approach was not acceptable.

With Stiquito II [2], a bigger version powered in a novel way, it became feasible to design a controller suitable for colony life.

After describing some of the experi-ments on Stiquito II, I'll take a look at the design strategies necessary for a flexible enough controller system to work with Stiquito II and other robots. I'll finish by discussing an IR-based communication system we use.

MISSION IMPOSSIBLE?

Like the original Stiquito, Stiquito II is Nitinol-propelled. But, with its 2 DOF (degree of freedom) legs, it is stronger and can move backwards.

Although bigger and stronger than the original, Stiquito II is still limited by its size (3" x 1" x 1") and the payload it can carry (1.5 oz.).

As if these restrictions aren't chal-lenging enough, the use of Nitinol further complicates matters. It re-quires about 1.25 W per leg actuation (2.5 V at 0.5 A], so an onboard power source is infeasible. The Nitinol also needs high-current drivers.

The experiments for Stiquito II and the colony varied in complexity. Let's look at two typical missions.

We wanted to develop efficient gaits for hexapod robots by using genetic algorithms. This mission requires direct control of the actuators from a remote computer.

A program on a host CPU calculates which actuator needs to fire when. A genetic algorithm tries to evolve a gait which propels the robot.

The gait's success is measured in how fast the robot walks and requires a simple leg controller with a commu-nication channel to a host CPU. The channel lets the CPU individually control each leg's actuators (up and down, forward and back).

Another mission involves a colony of robots that know how to walk au-tonomously, but need a high-level simulation to provide a virtual world.

The robots physically interact with an environment complete with ob-stacles by walking through it in a colony. Since each robot introduces some randomness, it's hard to simu-late virtually.

The virtual world includes virtual sensors for food and pheromone trails. These variables are hard to implement in a real robot colony and are simu-lated in this computer network.

For this mission, a fairly high-level controller is needed on the robot. The controller knows how to generate various walking gaits and may have some whiskers to sense the environ-ment and avoid collisions.

Each robot needs a low-bandwidth communication channel between a gateway host and the robot-agents. The simulation needs methods to sense the location of the robot-agents in the physical colony.

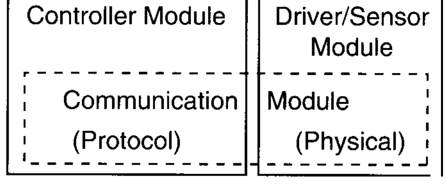


Figure 1—*The robot controller is divided info two physical modules: the driver/sensor module and the controller module. The communication system physi-cally interfaces with the driver/sensor module while the logical interface, where protocol is implemented, is in the controller module.*

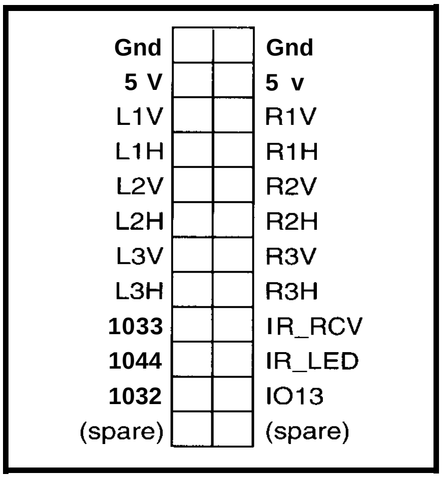


Figure 2—*The connector chosen is a 0.1″ dual-row header on the controller module and a mating socket on the driver/sensor module. There are pins for logic power/ground, the communication system, and the 12 actuators. Signal names such as L1V refer to "Left One Vertical" and R2H to "Right Two Horizontal." The legs are labeled in pairs from front to back (1–3).*

Using off-the-shelf microprocessor-based controllers would seem to do the trick. However, most of the sampled controllers are either too big and/or costly to consider.

MODULAR DESIGN

After analyzing the systems required for the missions, I developed the three modules in Figure **1.** Some modules change with the mission while others depend on the robot's physiology. Let's look at the modules:

- the driver/sensor module interfaces to the Nitinol and possibly some sensors. This module deals with power management.
- the controller can be low- or high-level, as required by the mission.
- communication breaks into two submodules: physical (link layer] and logical (protocol layer). The driver module holds the physical communication interface. The controller handles the protocol, which interprets and converts actions on the control mechanism.

By isolating the interface from the control and protocol, you can design mission-specific controller modules that plug onto the robot's driver/sensor module. To make the interface truly plug-and-play, a physical and electrical interface has to be specified.

The colony robots have 0.1″-centered dual-row header and sockets. With this physical interface, the controller module can serve as the robot.

For digital signals, 5-V TTL logic levels are chosen, which allows a variety of controller architectures to interface. With the hexapod, at least 12 I/O signals are needed for the leg actuators and two signals for communication.

The power (5 V) and ground interface are on one end of the connector to allow expandability and compatibility. A controller with more I/O signals should still plug into a driver module with the minimum I/O signals.

In fact, for the initial implementation, I left some room for expansion by allowing for 18 I/O signals plus 2 unassigned pins, as you can see in Figure 2.

With a modular design, there are other benefits as well. For our prototype, I built four identical controller modules and driver modules. With several identical modules, it was easy to identify whether a problem was due to faulty components or design.

I could also retarget controllers to different robots. The controller module was retargeted for an RC-servo-based hexapod by redesigning the interface and reprogramming the controller. The whole process took less than a day.

Finally, having several smaller modules opens up the possibility of networking them on one robot. That way, it can performs more complex tasks than a single module can provide.

STIQUITO II CONTROL SYSTEM

Stiquito I and II use Nitinol-based actuators. Flexinol is a special type of shape memory alloy made from Nitinol (nickel titanium alloy). However, it's commonly called *Nitinol wire.*

All we need to know about Nitinol is that it contracts **10%** of its normal length when heated above 90°C. It's a linear actuator which only pulls when heated. There are two actuators per leg: one moves the leg up, and other moves it back.

Since the Nitinol only pulls, the legs are made from piano wire. They act as springs, returning the leg to its relaxed position when the Nitinol cools.

To get the Nitinol wire above its activation temperature, a 500-mA current is passed through it. This procedure is called I^2R heating since the heat comes from resistive in the wire:

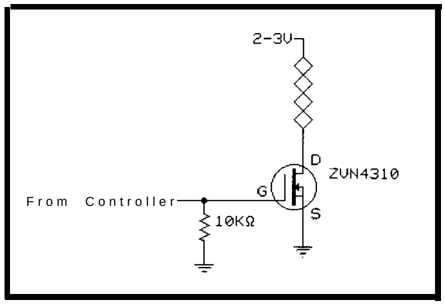


Figure 3—*The logic-to-Nitinol interface uses high-current MOSFETs to provide the 500 mA needed to activate the Nitinol. There 12 of these drivers--one for each actuator.*

$$P = I^2 \times R$$

The wires used on our bug are 5 Ω, so there's 1.25 W per leg actuation.

DRIVER/SENSOR BOARD

Each leg-actuator driver needs to switch about 500 mA. Since on/off actuation is all we need, a high-current MOSFET should do the job.

I selected one of Zetex's E-line MOSFETs. It comes in a TO92 package and drives up to 900 mA without a heatsink.

These MOSFETs are also particularly good since they can be driven directly from the TTL levels specified for the interface. As Figure 3 shows, I added a 10-kΩ

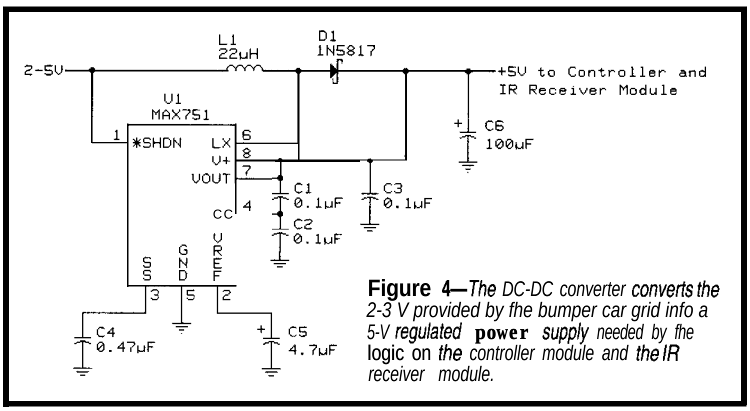


Figure 4—*The DC-DC converter converts the 2-3 V provided by the bumper car grid into a 5-V regulated power supply needed by the logic on the controller module and the IR receiver module.*

Listing 1—*This shows the finite state machine for a simple tripod gait with directional control. With this gait, a robot walks forward and backward and* **turns.**

```
entity gait is port (
  clk   : in bit;
  forw  : in bit;
  left  : in bit;
  right : in bit;
  back  : in bit;
  l1h, r1h, l2h, r2h, l3h, r3h: out bit;
  l1v, r1v, l2v, r2v, l3v, r3v: out bit );
end gait;
architecture auto of gait is
type state-type is (S0, S1, S2, S3, S4, S5, S6, S7);
signal pcnt: state-type;
signal ncnt: state-type;
  -- pragma CLOCK clk
  -- pragma CUR-STATE pcnt
  -- pragma NEX_STATE ncnt
begin
  process(pcnt, forw, left, right, back)
  begin
    case pcnt is
      when S0 =>
        ncnt <= S1; l1v <= forw or back;
        r2v  <= (forw or back) and right; l3v <= forw or back;
      when S1 =>
        ncnt <= S2; l1h <= back; r2h <= back and right;
        l3h <= back; l1v <= forw or back;
        r2v  <= (forw or back) and right;
        l3v <= forw or back; r1h <= forw;
        l2h <= forw and left; r3h <= forw;
      when S2 =>
        ncnt <= S3; l1h <= back; r2h <= back and right;
        l3h <= back; r1h <= forw or back;
        l2h <= (forw or back) and left; r3h <= forw or back;
      when S3 =>
        ncnt <= S4; l1h <= back; r2h <= back and right;
        l3h <= back; r1h <= forw; l2h <= forw and left;
        r3h  <= forw; r1v <= forw or back;
        l2v <= (forw or back) and left; r3v <= forw or back;
      when S4 =>
        ncnt <= S5; r1v  <= forw or back;
        l2v <= (forw or back) and left; r3v <= forw or back;
      when S5 =>
        ncnt <= S6; l1h <= forw; r2h  <= forw and right;
        l3h <= forw; r1h  <= back; l2h <= back and left;
        r3h  <= back; r1v  <= forw or back;
        l2v <= (forw or back) and left; r3v <= forw or back;
      when S6 =>
        ncnt <= S7; l1h <= forw; r2h  <= forw and right;
        l3h <= forw; r1h <= back;
        l2h <= back and left; r3h <= back;
      when S7 =>
        ncnt <= S0; l1h <= forw; r2h <= forw and right;
        l3h <= forw; l1v <= forw or back;
        r2v  <= (forw or back) and right; l3v  <= forw or back
        r1h <= back; l2h <= back and left; r3h <= back;
    end case;
  end process;
  process(clk,ncnt)
  begin
    if(clk = '1' and not clk'stable) then
      pcnt <= ncnt;
    end if;
  end process;
end auto;
```

pull-down resistor to the gate of the MOSFET to make sure it stays off when not driven to a high or low state.

Powering the Nitinol leg actuators requires 2-3 V, which is less than the 5 V necessary to power the controller logic. However, regulating a 5-V power supply to the 2-3 V required is not very efficient at high currents.

The solution-the MAX75 1 from Maxim, a 5-V regulated DC-DC converter (see Figure 4). It uses a 22-µH inductor and several capacitors. It provides a regulated 5-V output from an unregulated 2-5-V supply.

At this point, you're probably wondering how to get a 2-5-V power supply to the robot since this robot can

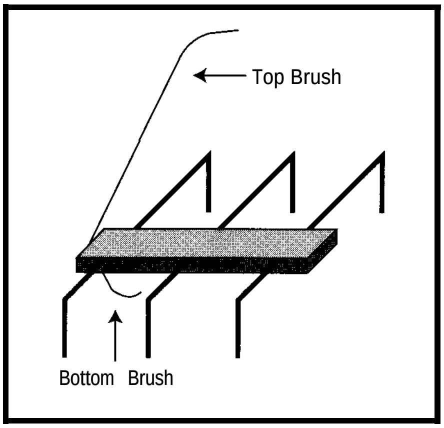


Figure 5—The brush on top of the Stiquito connects with a galvanized ¼" steel mesh above the robot, which carries the 2–3 V required by the Nitinol. The bottom brush connects to a copper ground plane made from a single-sided PCB. The copper needs to be cleaned occasionally to remove oxidation.

barely carry its own controller and driver module. In fact, to generate a tripod gait-a common gait for hexapods-at least six actuators have to be active at the same time.

This works out to about 3 A at 2.5 V or 7.5 W. Since a battery or solar-cell array cannot provide the energy density and the colony set up precludes a power tether, an alternate method of powering is needed.

Dave Braun came up with the idea of using brushes to get power from an overhead wire screen and a ground reference from copper sheet flooring (see Figure 5). This method provides each robot with a nearly unlimited power supply.

For the physical communication channel, I chose 40-kHz IR receiver modules, which are readily available and convenient. I'll talk more about the IR communication later.

A MOSFET drives an IR LED which signals the location and status to a video camera interfaced to a workstation. Software on a workstation finds the robot by looking for bright lights in a sampled frame (see Figure 6).

CONTROLLER MODULE

The controller module implements the mission controller algorithm and executes the protocol used for the communication channel.

The controller can be based on any technology applicable to the mission. I chose to use Xilinx FPGAs, mostly because they have a high I/O count and are in-circuit programmable.

In particular, I like the XC3030PC 44, which has 35 usable I/O pins and comes in a 44-pin PLCC package. The controllers could also be based on PIC or 68HC11 microprocessors.

In addition to the Xilinx FPGA, the controller module contains a 480-kHz ceramic resonator driven by the FPGA and an Atmel AT17C128 serial configuration EEPROM [see "Easy-to-Use Serial EEPROMs," *INK 71)*

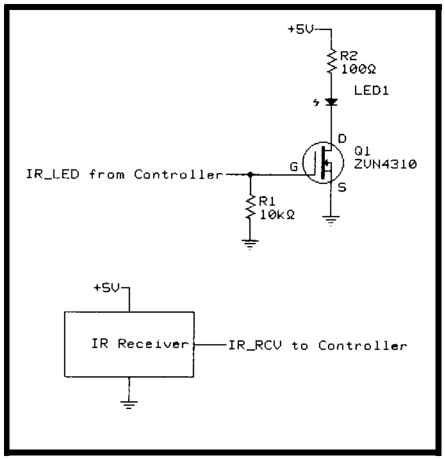


Figure 6—The standard 40-kHz IR receiver module, normally used in TV/VCR remote-control receiver applications, is available from Radio Shack and Digi-Key. A MOSFET drives an IR LED, which finds the robot in a video frame. The LED can also be used to communicate when modulated with a 40- or 32-kHz carrier.

I chose a 480-kHz resonator instead of a higher frequency crystal or oscillator module to reduce the number of dividers taking up logic on the FPGA. A download-cable connector provides remote configuration of the FPGA from a host as well as a 3-bit address jumper block. This block gives the controller an identity when used in a multiagent environment.

The controller can have an ID from 0 to 6. The ID 7 is special in our cur-

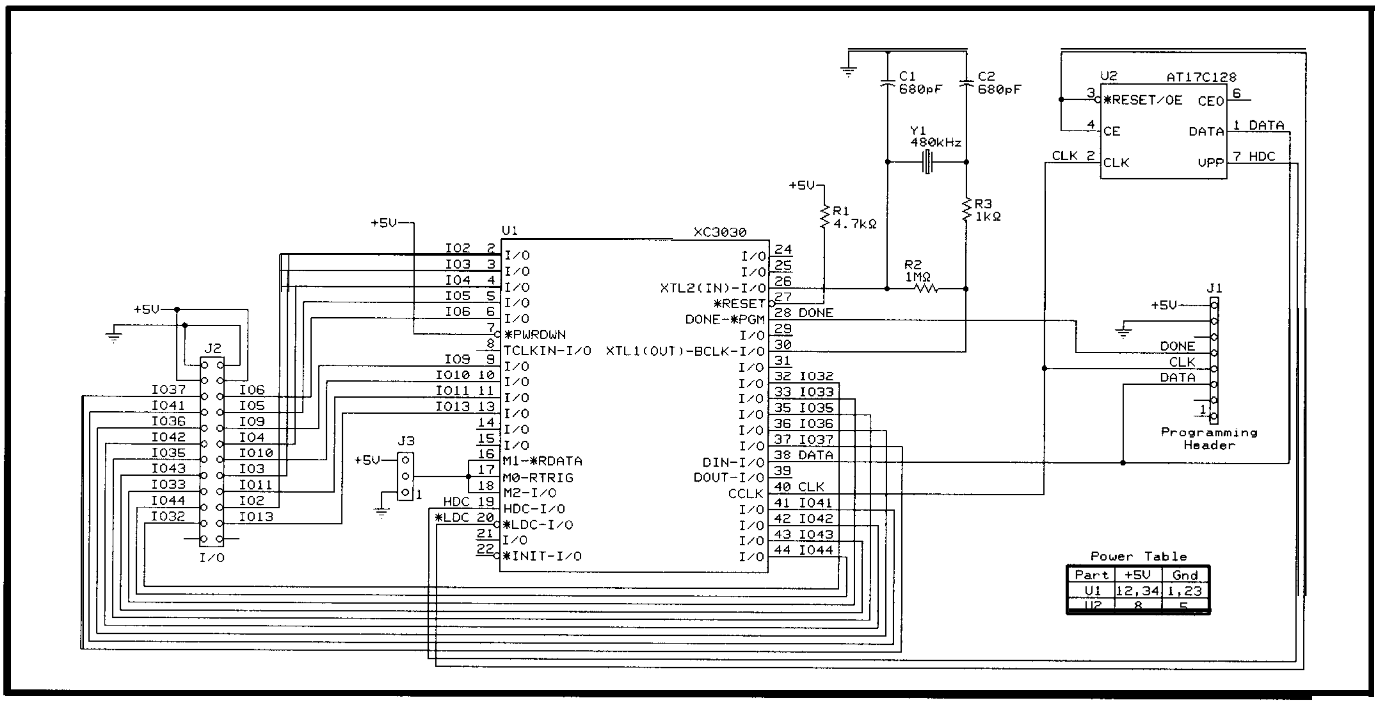
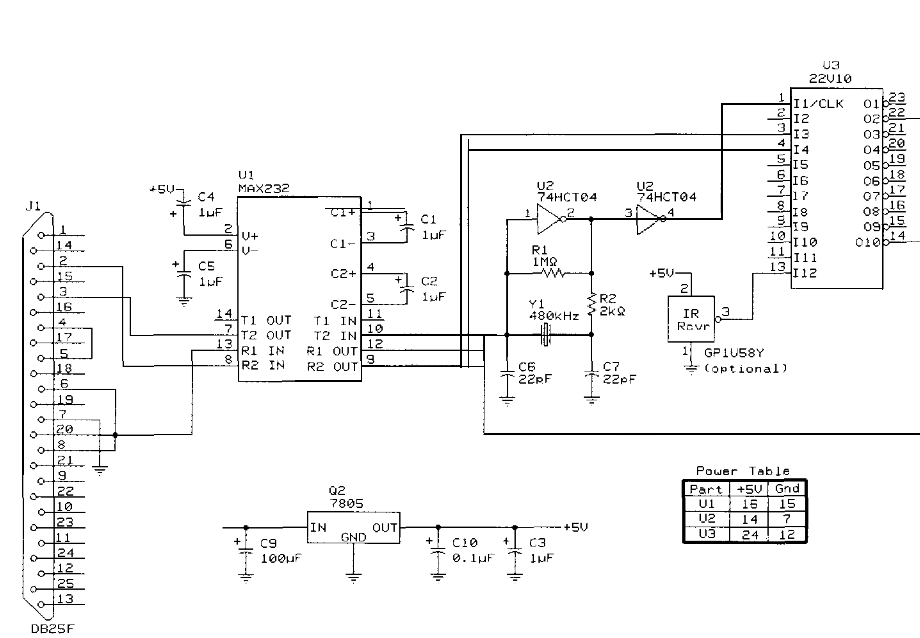


Figure 7—This controller module is based on a Xilinx FPGA. J2 is the I/O connector which mates with the driver/sensor module. J1 connects the controller to an FPGA development system via a download cable, while debugging the control logic in the FPGA. During normal operation, the FPGA is initialized from the serial configuration PROM.



Part	+5V	Gnd
U1	16	15
U2	14	7
U3	24	12

Figure 8—*The PAL used in this IR transmitter implements a gafed divide-by- 12 prescaler to provide a 40-kHz signal from fhe 480-kHz resonator. It a/so logically ORs the signals received from the optional IR receivers, which could be used for a bidirectional communication channel.*

rent communication scheme, which means "to all." Since selecting an ID of 7 on the controller is redundant, selecting this ID causes the controller to go into an autonomous diagnostic mode, as you can see in Figure 7.

The I/O signals from the FPGA directly control the leg actuator MOSFETs. They interface directly with the IR receiver module and the IR LED driver MOSFET. Digital inputs can also be read from the driver board.

COMMUNICATION SYSTEM

The communication system between the robots and the host is IR based. A host uses an IR-transmitter to convert standard RS-232 signals directly to 40-kHz by using on/off keying. The IR module on the robot provides a TTL-level signal with the same timing as the RS-232 signal sent by the host (see Figure 8).

Asynchronous serial protocol with a word structure of 8N1 at 300 bps was chosen for the IR channel. The transmitter and IR modules on the driver encode only the signal levels. The host and controller implement the protocol. Other types of link-layer protocols could have been used.

Depending on the mission, the received words are further decoded in command signals for individual leg actuators or commands to the gait controller. As you see in Figure 9, two of the command protocols currently implemented are simple and complex.

Since the controller is FPGA-based, the controller algorithm is a modular logic design. For the more complex mission, there are four modules: the I/O interface, asynchronous receiver, gait generator, and command decoder.

The I/O interface module defines the I/O pins on the FPGA and ties them to specific pin numbers. It also

A2	AI	A0	leg3	leg2	leg1	leg0	ON/*OFF
Simple Command Word							
A2	A1	A0	BACK	LEFT	RIGHT	(unused)	IR_LED
High-level Command Word							

Figure 9-*Currently, two control-word formats are used. "Simple" is used when control of each actuator is desired, while the "complex" format is used for high-level control.*

specifies the gates necessary to use the ceramic resonator as a clock network.

The serial receiver module implements the receiver part of a USART. It synchronizes the bit clock to the start symbol in the serial word and does the serial-to-parallel conversion for the data. The serial word is decoded into forward, reverse, left, and right signals, which are passed to the gait-generator module, as shown in Figure 10.

The gait generator knows about the tripod gait-the most common gait used by hexapod robots and insects. The six legs are divided into two groups of three legs which are activated at the same time. Each group cycles through a back-lift-forward-down cycle, as depicted in Figure 11.

The gait is implemented as a simple eight-state finite state machine. Each state represents a position in the cycle. A VHDL description of this state machine is shown in Listing 1.

When the controller module was retargeted to the RC-servo-based hexapod, it was only necessary to change the I/O interface module to generate PWM signals for the servos instead of the Nitinol's steady-state signals. The serial receiver, the gait generator, and command decoder remain the same.

We use a combination of free CAD tools like Caltech's diglog, an old PLD assembler retargeted to generate Xilinx netlists, and Xilinx software to capture and generate the FPGA design.

Although the sample VHDL code for the gait generator in Listing 1 compiles into a valid Xilinx netlist with Alliance 3.0, we don't use it. It is, however, much easier to read than the PLPL code we use now.

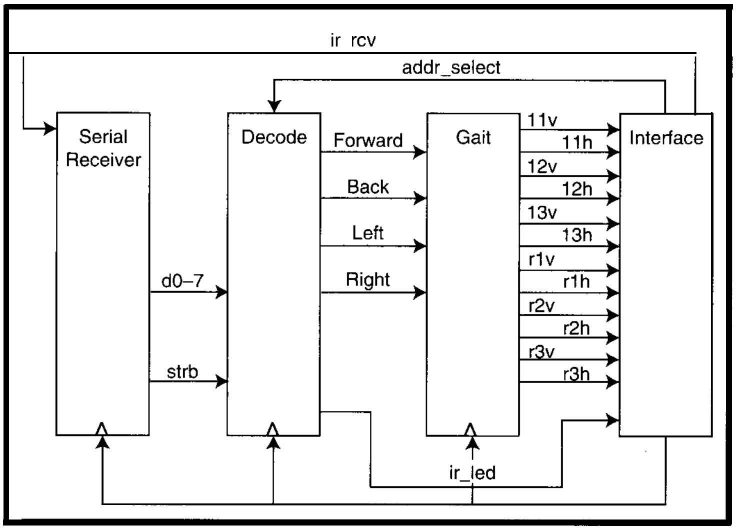


Figure 1 *O-Here is one possible architecture of the contrcller FPGA logic. The* serial receiver module performs the *async* serial-fo-parallel *conversion. A* command *decoder decodes fhe necessary signals for the gait controller. The* simple command decoder *could be* replaced with a protocol module *for complex multibyte* protocols.

NEXT STEPS

One improvement would be to use more complex controllers like a PIC, MC68HC11, or even MC68HC16. These processors come in different configurations and some include onboard A/D converters.

Using higher-density FPGAs would also be interesting since more complex functions and CPUs could be implemented in an FPGA.

Alternatively, dumber controllers could be used. For example, autonomous state machines could be used in small PLD-based controllers.

In some applications, even analog-based controller modules could interface with the driver board.

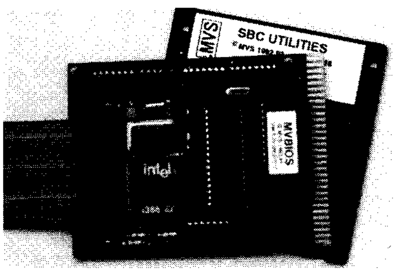
We now only have a primitive communication channel between the robot and host. It uses the IR LED on the driver module and a video camera. We could improve the channel's bandwidth by using the IR LED on the driver board to transmit at 40 kHz. The host then uses 40-kHz receiver.

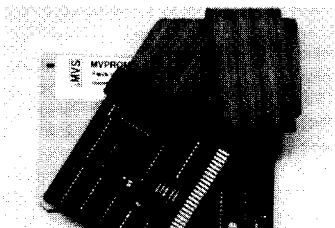
In fact, our current transmitter module has provisions to implement the receiver by converting the TTL signals from a IR receiver module to RS-232 levels. This conversion provides for a communication network where any agent (host or robot) can talk to anyone.

Other communication schemes could be implemented. For example, a duplex system could use 32 kHz for one of the channels.

The controller architecture is still evolving. The biggest challenge is powering the actuators and providing a stable-logic power supply from the same power source.

I hope to improve this controller by working out the kinks and adding





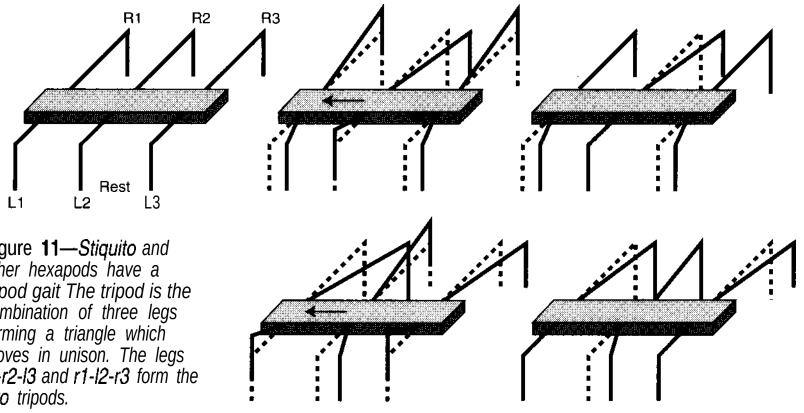


Figure **11**—*Stiquito and other hexapods have a tripod gait The tripod is the combination of three legs forming a triangle which moves in unison. The legs l1-r2-l3 and r1-l2-r3 form the two tripods.*

features. With forethought and planning, it's possible to create designs flexible and extensible-even when when you're on a tight budget! ❑

Ingo ***Cyliax works in computer sciences at Indiana University where he does system administration and spends a lot of time in the analog VLSI and robotics lab working on Stiquitos. He is also a partner at*** *EZComm* ***Consulting. You may reach*** *Ingo* ***at*** *cyliax@EZComm.* ***corn.***

REFERENCES

[1] J.W. Mills, "Stiquito: A Small Simple, Inexpensive Hexapod Robot, Part 1: Locomotion and Hard-Wired Control," Technical Report 463a, C. S. Dept., Indiana Univ., Bloomington IN, 1992.

[2] J.W. Mills, "Stiquito II and Tensipede: Two Easy-to-Build Nitinol-Propelled Robots," Technical Report 414, C.S. Dept., Indiana Univ., Bloomington IN, August 1994.

SOURCES

AT17C128 serial configuration EEPROM
Atmel Corp.
2125 O'Neal Dr.
San Jose, CA 95 13 1
(408) 441-0311
Fax: (408) 436-4200

Flexinol
Dynalloy, Inc.
18662 MacArthur Blvd., Ste. 103
Irvine, CA 92715
(714) 476-1026
Fax: (714) 476-3167
dynalloyQdeltanet.com

MAX75 1
Maxim Integrated Products
120 San Gabriel Dr.
Sunnyyale, CA 94086
(408) 737-7600
Fax: (408) 737-7194
http://www.maxim-ic.com/

Stiquito Colony Project
http://www.cs.indiana.edu/robotics/colony.html
http://www.cs.indiana.edu/hyplans/cyliax.html

XC3030PC44
Xilinx
2100 Logic Dr.
San Jose, CA 95124-7114
(408) 559-7778
Fax: (408) 559-7114
http://www.xilinx.com/

MOSFET
Zetex, Inc.
47 Mall Dr., Ste. 4
Commack, NY 11725
(516) 543-7100
Fax: (516) 864-7630
http://www.zetex.com/

Software
diglog: http://www.pcmp.caltech.edu/Chipmunk/
PLPL: ftp://ftp.cs.indiana.edu/pub/goo/PLD/
Alliance: ftp://ftp.ibp.fr/ibp/softs/masi/alliance/

I R S

407 Very Useful
408 Moderately Useful
409 Not Useful

In-Circuit Emulators

Part 2: Debuggers for Embedded Systems

After spending last month looking at software simulation and ROM emulation, the authors spend this month zeroing in on the more sophisticated monitor-based debuggers and true in-circuit emulators.

FEATURE ARTICLE

Graham Moss, Ross McMillan & Ken Mardle

Developing code for a microcontroller can be a nightmare. Let's look at the types of debugging tools available and compare strengths and weaknesses.

Part 1 described the problem of debugging microcontrollers in embedded systems and discussed debugging tools based on software-simulation and ROM-emulation techniques.

In this article, we discuss the operation and capabilities of monitor-based debuggers and ICEs.

MONITOR-BASED DEBUGGERS

Monitor-based debuggers approach the emulation by adding extra program code (the monitor kernel) to the control program. The idea has roots as old as the concept of a stored-program computing device.

The monitor communicates with a host and controls program execution and execution monitoring. You can examine and set the contents of internal controller registers and memory, set breakpoints for halts, and begin execution from any address.

Some vendors of monitor-based debugging products address only the software issue. They provide a monitor kernel for the target (either in source code or link-relocatable form) and a companion debugger to run on the host. You provide the communications link and other support features.

Other vendors provide varying levels of hardware to plug into the target in place of the controller. As with ROM emulators, some provide additional hardware-based features for triggering test equipment and tracing program execution.

Unfortunately, adding the monitor code to the user's program is easy. The difficulty is providing a way for the monitor to communicate with the host and minimize the resources stolen from the microcontroller.

Stolen resources is the greatest shortcoming of any monitor-based debugger. Physically, you lose at least one or two I/O pins for bidirectional serial communications with the host.

You lose the memory occupied by the monitor kernel and the internal RAM used by the monitor. You may also lose a UART, timer, and one or more interrupts and an interrupt pin. Less obviously, you can lose stack depth since the monitor may require the stack to control execution.

Since stolen resources can be such a major problem, many vendors provide work-arounds. Many use a more capable [i.e., has more I/O pins, memory, and peripherals) debugging controller than the one being emulated. If the monitor uses only the additional capabilities, the target loses no resources.

Unfortunately, the situation is rarely this ideal. In most cases, stolen resources are still a reality.

You can also use a controller in an externally expanded mode. Loss of external memory is preferable to a loss of significant internal memory or an important internal subsystem.

External expansion costs I/O pins, but you can regenerate those with an external latch and buffer devices mapped into the device's external memory address space (see Figure 1).

However, with this technique, I/O instructions may need to be converted to the equivalent memory-access instructions. These changes may be difficult or impossible for controllers with bit-oriented instructions that can be used with I/O ports, but lack equivalent addressing modes in memory.

Also, you may not be able to make the changes if you're dealing with compiler-generated code or using a third-party relocatable library without source code.

Basing the debugger on a "multiplexed bondout device" is a third way to avoid stolen resources. This device is a microcontroller with extra internal logic. It allows unused time within normal external memory-address cycles to multiplex the port information for the ports sacrificed. It permits operation in an externally expanded mode. Simple external logic properly regenerates the sacrificed ports without changing the code (see Figure 2).

Unfortunately, this problem is the only one the multiplexed bondout device solves. Many debuggers are only usable with target systems which operate the microcontroller in pure microcontroller mode (i.e., all code fetched from internal memory and all pins used for I/O).

They generally don't operate in externally expanded or mixed mode. (In mixed mode, code is fetched from from internal memory, while data or user peripherals are mapped in external memory.)

Warning: some vendors of monitor-based debuggers with multiplexed bondout devices promote their products as ICE systems. While cleverer designs provide some of the advantages of a true ICE, they do not-and cannot-provide the same degree of overall transparency.

We trust this article will help you ask the questions to expose such deceptions. If the price for an ICE system seems too good (currently any system much under $lOOO), ask searching questions.

MONITOR-BASED DEBUGGER FAULTS

Even if all the issues associated with stolen physical resources can be solved, monitor-based debuggers have other significant shortcomings.

For instance, consider setting a breakpoint so the monitor regains control when a program's address is reached during execution. Typically, the monitor replaces the original instruction at that address by a jmp or call which vectors execution back to the monitor. When executed, the monitor regains control, replaces the original instruction code, and interacts as required.

For many microcontrollers, a jump or call occupies two or three bytes. But, the original instruction where the breakpoint was located may only have occupied one. As a result, the breakpoint temporarily corrupts one or more subsequent instructions.

In many instances, this isn't a problem. However, consider the code fragment shown in Listing 1. It's taken from an 8051-based multiprocessor application using token passing to control access to a common communication bus.

If a breakpoint is placed on the single-byte ret. at the bottom of the send-packet routine, the first instruction of the following send_byte routine is overwritten. If the access-control token is acquired on entry to the send-packet routine,the send_byte routine is called before the target ret is reached and the corrupted instruction is struck.

Similarly, if a breakpoint is placed on the single-byte setb C instruction thattells relinquish-token to release the access token (rather than withdraw the token request), the call to the relinquish-token is overwritten. If the access-control token is not acquired within a reasonable time, the conditional jump is taken to the sp_exit label and the corrupted call is struck.

In both cases, the outcome is unpredictable. It could be innocuous, or it could lead to a incorrect result (e.g., not releasing an acquired token).

It could mislead you into thinking you had quite a different bug from the one you're chasing or cause execution to go awry. The breakpoint might never be reached, and the monitor might lose control. In applications involving control of mechanical or electrical systems having substantial stored energy (e.g., engine and motor controllers), such loss of control can be disastrous.

The most annoying aspect of this problem is when it will occur. If you're debugging at the assembly-language level, you might avoid it in most cases, although the instances you miss waste a lot of time. When debugging a high-level language program at the source-code level, however, you'll be hard-pressed to predict such errors.

Some monitors are clever enough to recognize when this problem occurs (e.g., short forward jumps), and they report that they cannot set a breakpoint under this condition. While this feature is arguably better than crashing the debugger, it prevents you from exercising your code under the conditions you established.

Monitor-based debuggers also use scarce memory resources including program memory, internal memory, and stack space. If your application already uses most of the microcontroller's internal memory, and the monitor doesn't provide additional memory for its own use, you may find it difficult to use.

You might temporarily reduce the size of data buffers, but if you're dealing with a

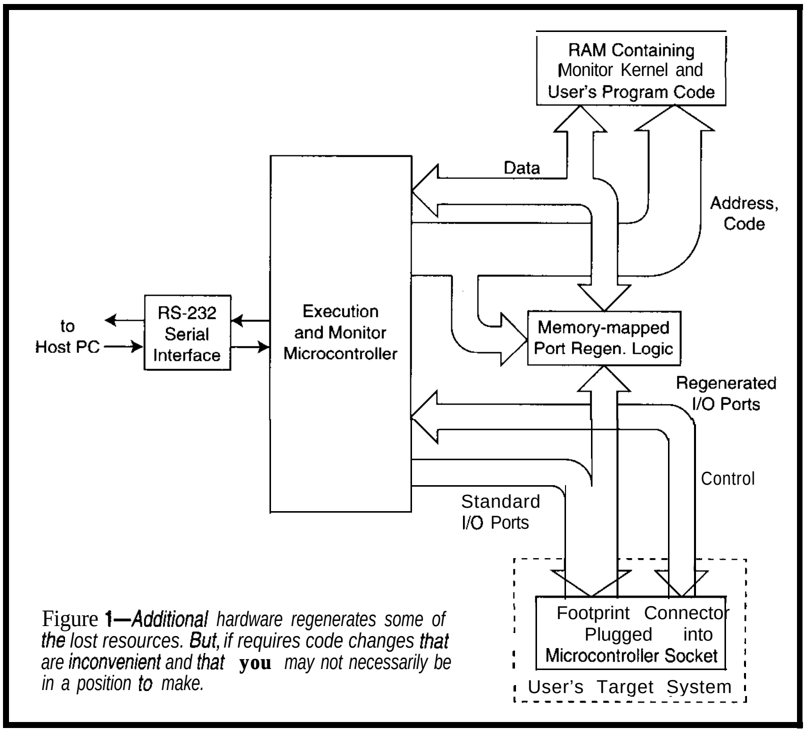


Figure 1—*Additional hardware regenerates some of the lost resources. But, if requires code changes that are inconvenient and that* **you** *may not necessarily be in a position to make.*

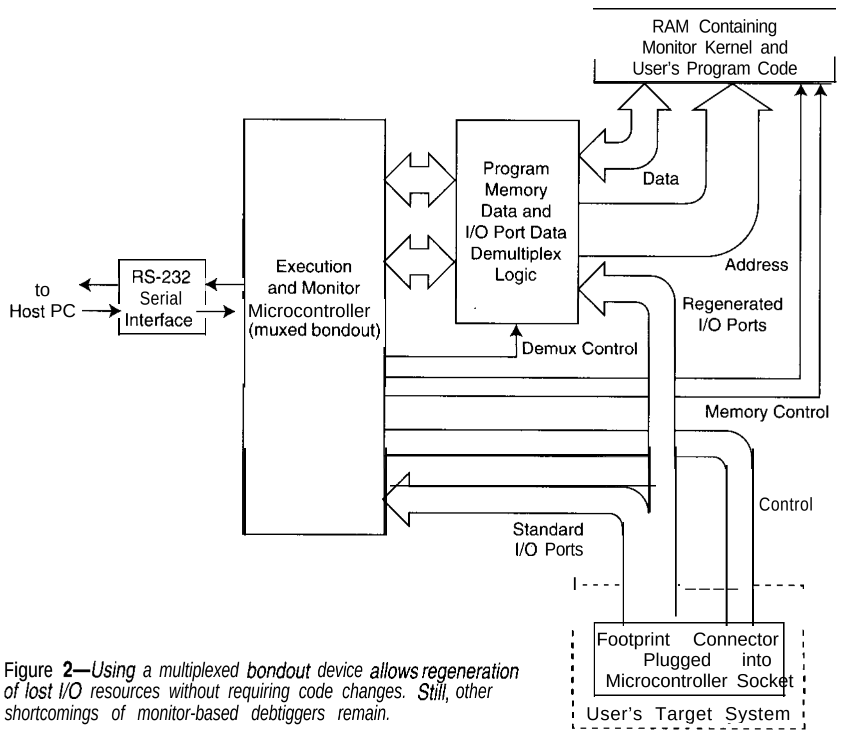


Figure 2—*Using a multiplexed bondout device allows regeneration of lost I/O resources without requiring code changes. Still, other shortcomings of monitor-based debtiggers remain.*

low-end microcontroller with only 1 KB of code space and 64 bytes of data memory, the monitor's overhead may be intolerable.

Even if there's sufficient code and data space for the monitor, stack-depth reduction may be a significant problem. Some microcontrollers limit their hardware return stacks to four levels. Losing one level can be crippling.

Lack of adequate stack depth can cause a perfectly well-behaved bug-free program to misbehave or exhibit apparent bugs when debugging. When stack-related problems occur, debugger control is lost, and problems can be difficult to track.

If you suspect such a problem, temporarily increase the memory allocated to the stack. Fill the stack area with a sentinel value and execute the program. Then, examine how much of the stack area has been written over (i.e., no longer contains the sentinel value). You may save yourself hours of unnecessary debugging.

Another shortcoming of monitor-based debuggers is that, by nature, it is exposed to the program being debugged. If the bug you are trying to locate results in spurious writes to data memory, it may corrupt data used by the debugger.

Alternatively, the bug may corrupt the states of I/O pins or other subsystems used by the monitor, disabling communications with the host. When this happens, you can't use the debugger to find the problem.

Interrupts pose a significant problem as well. To regain control at any time, the monitor must make use of an interrupt. Often this requires giving up an interrupt pin. Generally, that pin is associated with the highest interrupt priority possible, although some controllers allow interrupt priorities to be promoted for any interrupt source.

This problem is exceptionally difficult if only one interrupt pin is available and the target is using it for a critical purpose like detecting AC-supply zero crossings for timekeeping or phase control. Even in systems which regenerate I/O ports, it's often impossible to properly reproduce the functionality of a lost interrupt pin.

You must also be careful not to disable the interrupt. You need to be able to interrupt execution and-regain control from a wayward program.

At best, this precaution is inconvenient if you need to turn the interrupt off for any reason-perhaps to ensure an indivisible multibyte update for a timer. At worst, a bug crashes the program and disables the interrupt.

The final shortcoming of the monitor-based debugger is its inability to tightly control peripheral subsystems when single-stepping. Even though it appears to execute one instruction at a time (e.g., serial communications UARTs may appear to send a whole character in one instruction instead of the usual hundreds of instruction cycles), timers may count the equivalent of many instructions.

This shortcoming makes it impossible to debug timing-related problems. It also poses problems for external subsystems in the target which rely on

Listing I--Some *code can be a minefield for software breakpoints with monitor-based debuggers,*

```
; attempt to send a buffered packet to the host
; r0 -> packet buffer, r7 = packet length
; dptr = packet sequence number
send-packet:
  call acquire-token     attempt to acquire bus-access token
  jnc  sp_exit           defer transmission if unsuccessful
  call initialize-CRC    else initialize CRC for new packet
  mov   a, r7
  call send-byte         send packet-length byte
sp_loop:
  mov   a, @r0           get data byte from packet buffer
  call send-byte         update CRC and send data byte
  inc  r0                point to next byte in buffer
  djnz  r7,  sp_loop;    repeat for each byte in the packet
  call send&CRC          send packet CRC
  setb c                 flag that token may be released
sp_exit:
  call relinquish-token ; release token/withdraw request
  ret
send-byte:   ; update packet CRC and send byte in A to host
  ...
  ret
```

precisely timed and controlled waveforms generated by timers within the microcontroller.

On systems which support reading timer values by the monitor, write a simple program which starts the timer running and then enters an endless loop of a few NOP instructions.

Run the program from the monitor and stop it in the loop. Display the timer before and after single-stepping through a NOP instruction.

Compare the two timer values with the expected execution time of the NOP. If the two counts differ significantly-typically, by tens or hundreds of counts-the debugger isn't tightly controlling peripheral subsystems.

This test isn't foolproof since some monitor-based debuggers read and reset timers while single-stepping. Since underlying timer hardware can be incremented many times, associated timer-related I/O, like capture-and-compare logic, misbehaves.

Such debuggers cost a few hundred to over a thousand dollars, depending on their features, hardware, and ancillary functions. Lower-cost systems are common among cash-strapped developers and educational institutions.

Upper-end systems exhibit fewer limitations, but they're in the same price bracket as some ICEs and by comparison may not be cost-effective.

Despite their limitations, monitor-based debuggers are popular. For some microcontroller families, they're the only choice if the manufacturer has not manufactured emulators.

In the end, their usefulness depends on the success of the designer in addressing the debugger's limitations.

BONDOUT-BASED ICEs

The term in-circuit emulator is properly reserved for emulation systems which consume none of the resources of the device being emulated. This type of emulator represents the ultimate in emulation technology. It's as close as you can get to full transparency for debugging purposes.

To emulate a controller without consuming its normal I/O resources, the address and data buses for program memory and some control signals must be accessible outside the device.

Traditionally, a special development version of the microcontroller is built. It has additional pins wired to extra bonding pads connected to the signal nodes on the silicon die and is referred to as a *bondout*. A production controller in a 68-pin PLCC package might be supplied in a 121-pin ceramic PGA package with extra control logic.

In some cases, the same silicon die is used in normal production devices and the extra bonding pads are left unconnected.

The market for bondouts is small, and the costs of development and packaging are high since die bonding is often done manually. Bondouts tend to be expensive, costing 15-25 times that of their production counterparts.

True costs may be higher. Manufacturers subsidize bondouts to gain design-ins. For commercial and technical reasons, many manufacturers restrict access to bondout devices. Companies using them to build ICE systems must agree to fairly stringent conditions.

Instead of physically bonding out the extra signal lines, you can time-multiplex the I/O pins so they carry the internal address, data, and control information and the normal I/O port data. Relatively simple external multiplexing and demultiplexing logic gives the illusion of a bonded-out controller.

This approach requires the manufacturer to add logic on the silicon die. It provides the multiplexing functions and puts the device in a special mode when the multiplexing is enabled.

Since the amount of logic needed is small, most manufacturers design one die so low-cost mass-produced devices can be used. For obvious reasons, low-cost ICE systems are inevitably based on multiplexed bondouts.

You probably realize by now that if an ICE is to be truly transparent, there must be more to a bondout device—either a true bondaut or a multiplexed one-than just providing access to the address and data signals associated with the program memory.

They also need to provide access to the contents of registers, data memory, and peripheral subsystems. And, it's important to provide special control signals to provide tight control over the timers.

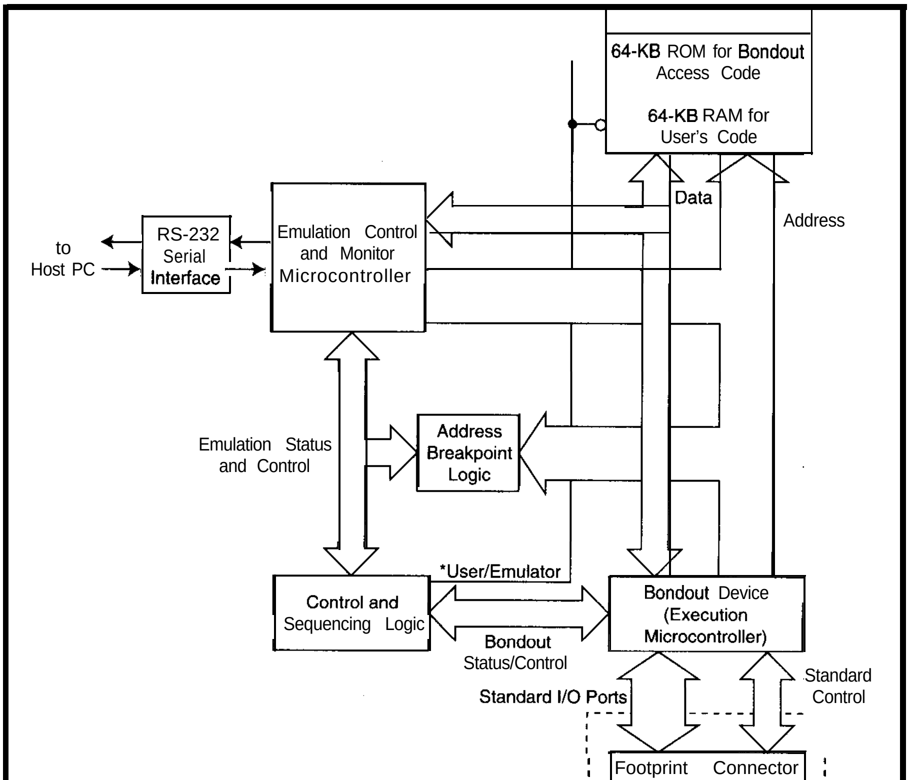


Figure 3-Here is *the architecture of a typical bondout-based in-circuit emulator. These emulators represent the ultimate in debugging transparency, but they are complex and traditionally expensive.*

In some ways, these facilities are more easily obtained with a true bondout than a multiplexed one. In general, the highest performance in terms of speed of operation is provided by ICE systems based on true bondouts. Some vendors offer a lower cost unit based on a multiplexed device and a higher cost one based on a true bondout.

The somewhat simplified architecture of a bondout-based ICE is shown in Figure 3. Standard ports and control signals for the bondout attach directly to the target footprint connector. Specially bonded-out address, data, and control signals connect to the emulator's memory and control logic.

Memory is generally provided for the code being debugged and for the special code accessing internal data within the bondout device. Breakpoint logic monitors the address bus and halts execution when a match with target breakpoint addresses occurs.

LIMITATIONS

While it's obvious why bondout-based ICEs provide such a high degree of transparency, we need to ask what their limitations are. Compared to the other classes of emulation discussed, bondout-based ICEs:

- have far fewer limitations
- can perform virtually any debugging task the others can
- can perform tasks with a much higher degree of transparency

However, no ICE is 100% transparent. ICEs share a number of minor limitations with the other forms of emulators (with the possible exception of software simulators).

First, subtle differences in the electrical and timing characteristics of I/O pins exist between production controllers and emulators. These variations arise due to differences in device packaging, the use of ribbon-type cables for emulator footprint connectors, or the presence of buffers in emulators.

There are also differences in the fabrication processes used for production devices and true bondouts and the need for demultiplexing and port-regeneration logic with multiplexed bondouts. In general, these differences aren't a problem if the target is designed within normal timing and I/O drive and loading constraints.

By nature, emulators generate significant amounts of electrical noise and are susceptible to external noise. This can be a problem with controllers in mixed-signal systems where low-level analog signals are degraded.

Problems arise also in RF communications applications where noise suppresses receiver gain through AGC action. Controllers with onboard A/D converters can have problems due to crosstalk between ribbon cables and footprint adapter connectors.

Target systems with high inherent-noise levels may pose noise-susceptibility problems for an emulator. Care must be taken to avoid problems with earth differentials between an emulator and a target to avoid creating earth loops which degrade signals or increase noise susceptibility.

All emulation systems tend to have problems with watchdogs and execution condition-monitoring circuitry that causes the controller to be reset whenever the emulator stops execution. In many cases, debugging is only possible if that type of circuitry is temporarily disabled.

Many controllers provide power-saving operating modes where various subsections of the device are shut down. Emulators vary widely in their support for such modes.

In general, however, it's impossible for an emulator to enter a power-down mode. Many emulators can't support low-voltage operation, although some newer offerings allow this.

A bonded-out controller is limited in providing access to internal data in internal memory and peripheral resources.

Such limits become an issue when breakpoints depend on data values, which must be accessed internally, rather than on address values, which are accessible externally.

Few bondout devices offer real-time access to internal data. Data breakpoints often involve nonreal-time execution via rapid automatic single-step. The data location is read by the emulator and compared with the target breakpoint value after each step.

A more pragmatic limitation of ICE systems is their cost. Apart from the higher initial purchase cost, it's sobering to inadvertently damage an emulator through a target fault condition, slipping with an oscilloscope probe, or plugging the emulator footprint connector in backwards!

A replacement bondout device alone costs hundreds of dollars. It's almost impossible to fully protect an emulator against such abuse, and the damage may not be isolated to the emulator.

A high-energy electrical fault may also damage the host and its peripherals. With some applications, some form of galvanic isolator between the emulator and the host is a wise investment if the emulator and computer interface configuration allows it.

BUYER BEWARE

Monitor-based debuggers provide a low-cost entry into microcontroller product development, but at what cost?

When controller resources cannot be lost or real-time is critical, the only viable solution is a full ICE system.

In the final article in this series, we'll provide an in-depth look at the features of the PDS51, Philips Semiconductor's low-cost 8xC51-family ICE development system. ❑

Graham Moss is a design engineer with the applications laboratory of Philips New Zealand, which designs and markets a variety of low-cost development tools for microcontrollers. He can be reached at graham@pds.co.nz. The laboratory's web site is at http://www.he.net/~pds/.

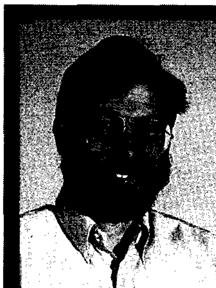
Ross McMillan and Ken Mardle are design engineers with Applied Digital Research, a New Zealand company specializing in embedded systems solutions and development tools. You can reach Ross and Ken at info@adr.co.nz.

IRS

410 Very Useful
411 Moderately Useful
412 Not Useful

Tuning Up

A Digital Zero-beat Meter



Hams communicate by tuning their radios to the same frequency. But, Ed has a lead ear. His new project listens to the transceiver's audio output and indicates which way to turn the knob for that Oh-So-Sweet same pitch.

FIRMWARE FURNACE

Ed Nisley

Adjust the transceiver frequency until the audio tone of the station you wish to zero beat is identical in tone to the transceiver's CW monitor note.

Operating Manual, 5th Edition
American Radio Relay League

No matter how much experience you gain in one field, you begin as a novice in another. Last year, I passed the exams required for an amateur radio license and became, not a novice, but a technician. The difference is important. Novice licensees must demonstrate their knowledge of Morse code.

In 1991, the FCC eliminated the Morse-code requirement from technician-class amateur radio licenses. Since then, several hundred thousand new radio operators have joined an old hobby.

Many of those hams-new codeless techs like myself-regarded Morse code as an obsolete impediment rather than a different communications mode. The enthusiastic instructors at the Raleigh Amateur Radio Society licensing class didn't change my mind overnight, but they certainly gave me a different perspective.

Although I skipped the Morse-code test during class, I wound up studying Morse over the summer and-somewhat to my surprise-passed the five-word-per-minute novice test in the fall class. While Morse code may be obsolete for heavy-duty communications, I found it a challenge and an art form.

There remained just this one, little, tiny problem. To communicate successfully, you tune your radio to the

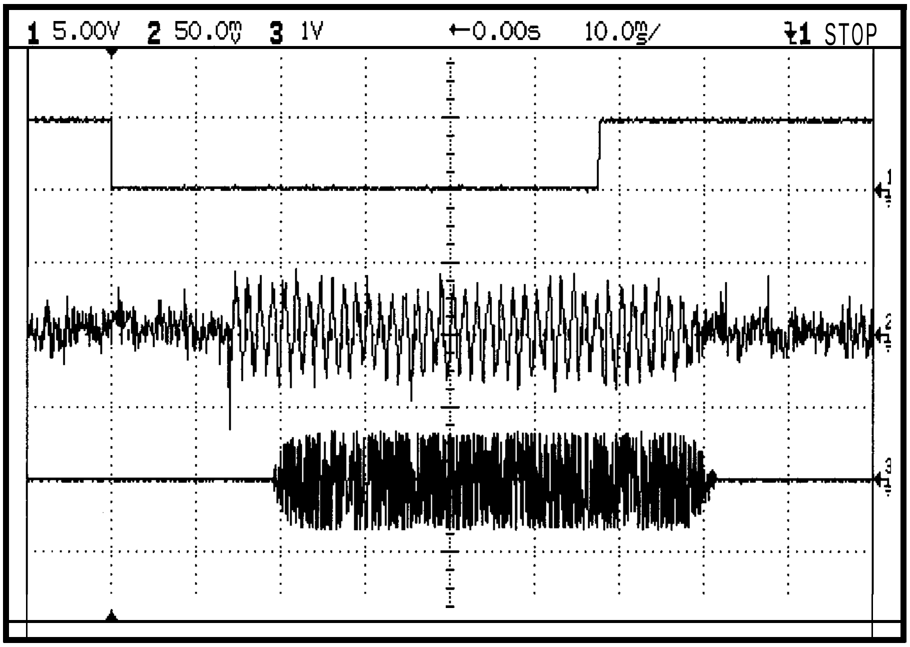


Photo 1—The upper trace is a 20-wpm dif from my electronic keyer fhaf triggers an audible sidefone and a radio-frequency burst from fhe transceiver. At fhis low sweep speed, my digital scope drastically undersamples fhe 7.125-MHz RF signal and shows random lines rather than the complete carrier envelope.

same frequency as the other operator. It seems I'm gifted with a lead ear that cannot tell when two tones sound the same.

So, I built a little widget that listens to my transceiver's audio output and tells me which way to turn the knob. A smidgen of firmware in an Atmel 89C105 1 simplifies the user interface to the point of triviality. There may be other ways to solve this problem. But, well, I'm not a novice at microcontroller projects.

Unlike some of the killer projects I've presented in these pages, Zerobeat is straightforward enough that we can begin with the fundamentals and pay careful attention to the details. As Steve pointed out in his June "Priority Interrupt" (*INK* 71), sometimes one project can illustrate how experience meets the real world. Although this may look like a simple project, the devil hides in the details.

This month, I describe the problem, show off hardware, and begin examining the audio circuitry. Next month brings the firmware that processes those signals, converts time periods into frequencies, and presents results on a moving-dot LED display.

Finally, in October, I'll cover the power-supply and the firmware preserving the CPU's internal RAM.

THE PROBLEM

Strangely enough, Samuel Morse didn't invent Morse code. His assistant, Alfred Vail, came up with the idea when he began recognizing symbols in the sounds Morse's telegraph mechanism made as it transcribed electrical signals onto an inked tape. Morse, however, had a contract with Vail that retained all credit and rights.

So it goes.

Morse code consists of pulses separated by pauses. The duration of the shortest pulse, traditionally called a dit, forms the basis of all other timings. Because pulses have no other modulation and the overall timing is entirely self-clocking, Morse-code transmitters and receivers require surprisingly little hardware.

Morse's original telegraphic application switched direct current through wires between the transmitter and receiver. Shortly after Marconi boosted Morse code into the radio era, sending wireless greetings required little more than a gated oscillator. In fact, amateurs generally refer to Morse-code transmissions as continuous wave or, more commonly, CW.

Figure 1 shows the components of a CW radio transmitter. An RF (radio frequency) oscillator sets the transmission's carrier frequency. Closing the key switch turns on a power amplifier, sending the carrier to the antenna. The switch also turns on an AF (audio frequency) oscillator that produces an audible sound, called the sidetone, for each pulse.

If I could send or receive at 20 words per minute, each dit would look like the 60-ms pulse in the top trace of Photo 1. The AF sidetone, only slightly

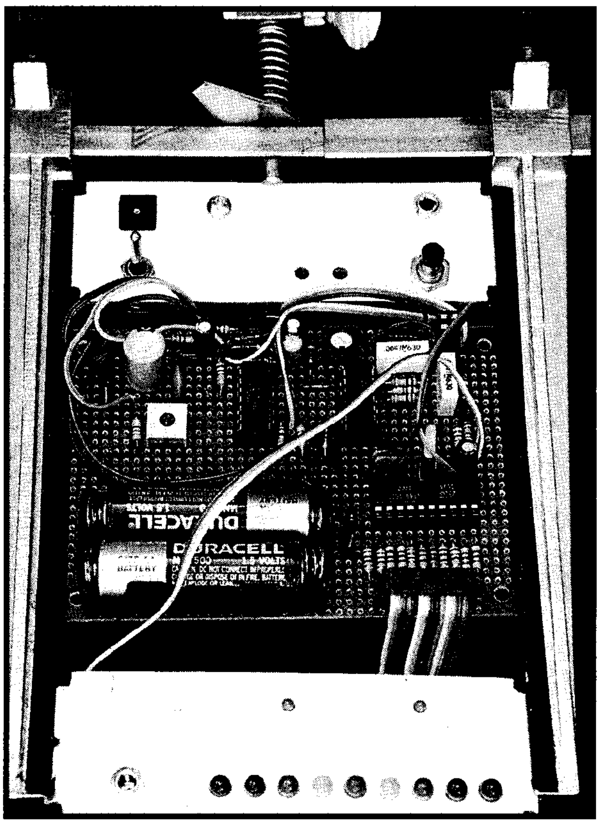


Photo 2-This circuitry converts the frequency difference between an input audio tone and a stored reference info a moving-dot LED display. The Atmel 89C1051 seems dwarfed by the surrounding park, doesn't if?

larger than the inaudible background hiss, appears in the middle trace. The nicely rounded ends of the 7.125MHz RF pulse in the bottom trace reduce the overall signal bandwidth to a few hundred hertz.

In the simple direct-conversion receiver shown in Figure 2, a nonlinear mixer combines the amplified RF from the antenna with an LO (local oscillator) signal to produce what's called an IF (intermediate frequency] signal. Although the mathematical modeling, not to mention the actual electronics, can be daunting, the mixer produces output signals at all the sums and differences of all multiples of both frequencies.

The filter eliminates all the mixer output signals except one at the difference between the two input frequencies. For example, a 7.125000-MHz RF signal mixed with a 7.124300-MHz LO produces an audible 700-Hz tone.

As you tune the receiver, the mixer output declines in frequency as the LO approaches the incoming signal, producing an audible beat note. When the two frequencies match exactly, the mixer output frequency becomes zero, hence the term "zero beat."

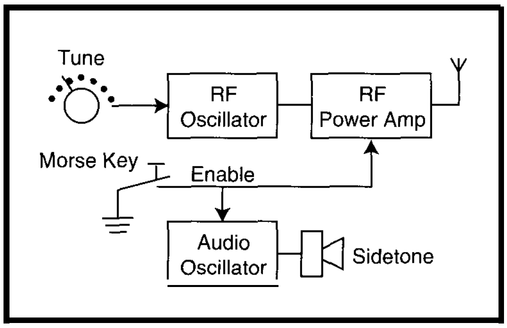


Figure 1-A Morse-code *transmitter requires little more than a radio-frequency oscillator and power amplifier controlled by a switch. A second oscillator provides audio feedback* when the *switch closes.*

The beat note increases as you continue tuning, so the zero-beat frequency occurs only when your receiver is at the transmitted frequency.

You then adjust your transmitter to the same frequency and offset your receiver to produce an audible beat. There's no connection between the transmitter's sidetone and the received audio, but I suspect in the old days hams made them equal.

Although such a simple setup delivers surprisingly good results, present-day transceivers devote far more hardware to the job. My transceiver, a Yaesu FT-767GX, has three IF stages, an assortment of filters, and according to the manual, a trio of microprocessors riding herd over everything.

The additional IF stages and filters remove the "other side" of the mixer's output in CW mode, so no audio signal emerges when the LO frequency exceeds the signal frequency. This one-sided response means that, instead of listening for an obvious zero beat, I must have absolute pitch.

This is, I suppose, progress.

Although I lack perfect pitch, I can build it into a microcontroller. One cycle of a 700-Hz tone lasts 1.4 ms, enough time for even a laggard CPU to run through a few thousand instructions. Sounds like a simple matter of software-even to me.

THE BOX

Photo 2 shows the Zerobeat hardware. For a one-off project like this, I have no qualms about soldering hookup wires directly to perf-board, but you may favor a myriad of connectors. Because the Atmel 89C 105 1 micro-controller puts all the high-speed digital electronics on one chip, component layout makes little difference.

The 89C1051 drives the row of nine LEDs as a simple, analog-style tuning indicator. When the receiver is tuned slightly below a Morse-code signal, the leftmost LEDs show how far off it is. The center green LED goes on when the frequencies match. The rightmost LEDs report when it's tuned too high.

My parts drawer supplied red, orange, yellow, and green LEDs. In retrospect, a single green LED in a sea of red would reduce the Christmas-tree effect. Analog purists may prefer a tuning meter with a real dial and needle, but LEDs show up better in the dim light found around my desk.

The 89C105 1 can handle up to 20 mA at each port pin, but no more than 80 mA total. The Zerobeat firmware ensures that no more than two LEDs go on at once, so I selected the series resistors to draw about 15 mA through each LED. If you feel industrious, you can match the intensity of the LEDs rather than equalize the currents.

The rear-panel jack connects to the transceiver's earphone output, and my headphones plug into the jack in front. Pressing the push-button switch stores the current audio frequency in the CPU's internal RAM as the reference frequency. The backup battery maintains that value in RAM when the power goes off, so you need to set the reference frequency only once.

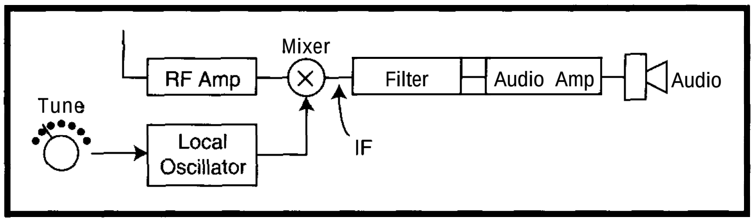


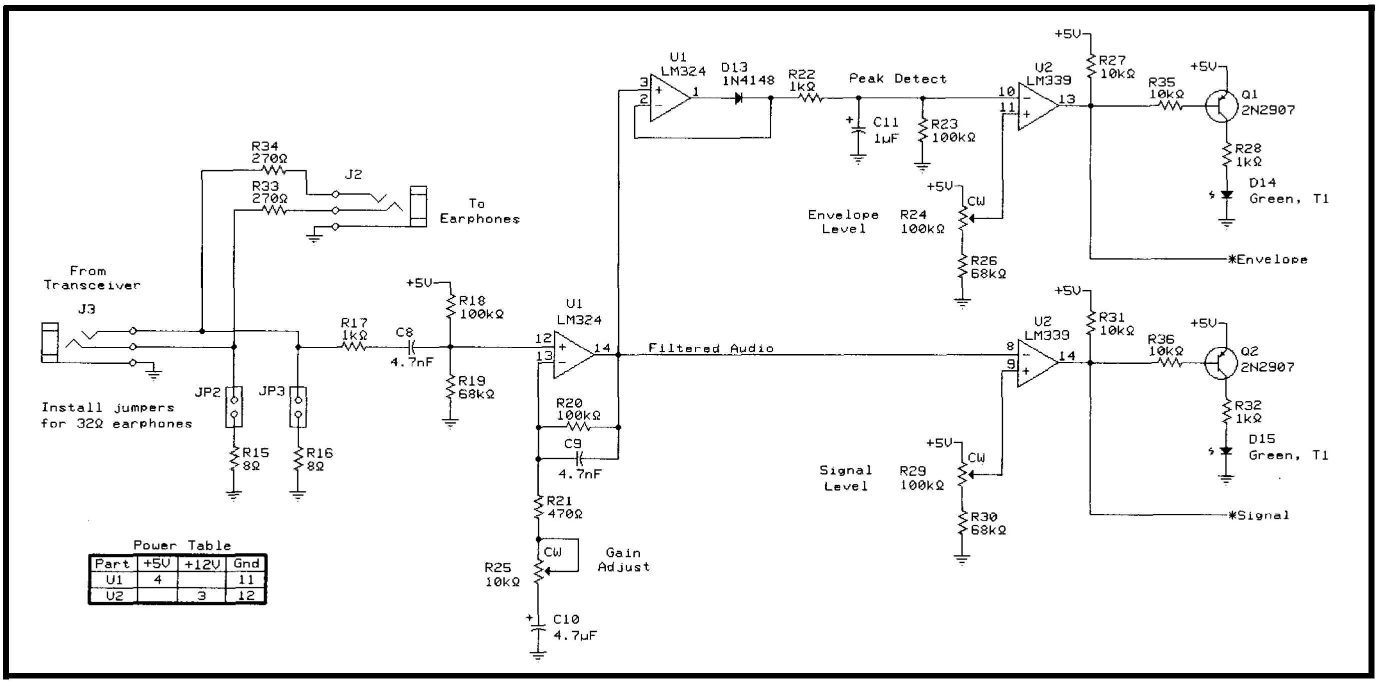
Figure 2-A simple Morse-code *receiver mixes a local oscillator* ***with*** *the received* ***RF*** *signal Tuning* ***the*** *local oscillator close to* ***the*** *input signal* ***frequency produces*** *an audible output tone ar the* difference of the two frequencies. Your brain *supplies the signal processing* and pattern *recognition.*

With the Zerobeat box perched atop my transceiver, tuning a signal involves nothing more than listening through the earphones, watching the LEDs, and turning the transceiver's knob until the green LED goes on.

How's that for a graphical user interface?

LO-FI AUDIO

Morse-code signals, unlike current multimedia productions, consist of a pure sine wave. As a result, the analog circuitry can be relatively narrowband and entirely low fidelity. A frequency response from a few hundred hertz through a few kilohertz is adequate, with no requirements for either amplitude flatness or phase linearity.



Part	+5V	+12V	Gnd
U1	4		11
U2		3	12

Figure 3—U1a stores the peak values of U1d's band-pass-filtered output on C11. The *Envelope signal goes low during each Morse-code pulse, while 'Signal toggles in sync with the audio waveform. R33 and R34 attenuate the earphone output, allowing a higher audio level from the transceiver. The rest of the schematic will be in future columns.

My FT-767GX drives either an internal speaker or external headphones. I find that headphones help my concentration, but the loudest sounds I want to hear measure about 50 mV peak. The audio circuitry must amplify and convert that signal into levels suitable for the 89C 105 1 digital inputs.

R33 and R34 in Figure 3 form simple 10: 1 attenuators that reduce the output level driving my 32-Ω phones. The FT-767GX expects to drive 8-R phones, so R15 and R16 provide suitable loads.

Remove JP2 and JP3 if your transceiver expects a lighter load. If you have 8-Ω phones, reduce R33 and R34 to about 68 Ω.

The components surrounding U1d filter the audio signal, rejecting much of the inaudible hash you see in the middle trace of Photo 1. Uld, part of an LM324 quad op-amp, provides enough gain to boost the signal up to a reasonable level. R25 sets the overall gain.

Pin 14 of U2c, part of an LM339 comparator, goes low whenever the filtered audio signal rises above the threshold set by R29. For the relatively clean tones we encounter here, this simple detection scheme works well enough. It can't handle extremely noisy Morse-code signals, but then, I can't decode them well either.

Atmospheric noise introduces occasional spikes and bursts that can confuse the frequency measurements. The diode in the feedback path of Ula detects signal peaks and stores them in C 11. R23 provides a discharge path that reduces the voltage on Cl 1 after each peak.

U2d compares that voltage against the reference set by R24. For typical Morse-code signals, U2d's ouput goes low after a few cycles and remains low slightly after the end of the pulse. Noise spikes either don't charge C 11 above the threshold or cause only a brief blip that the firmware can detect and ignore.

Photo 3 shows these circuits at work. Trace 1 is pin 14 of U1d, the op-

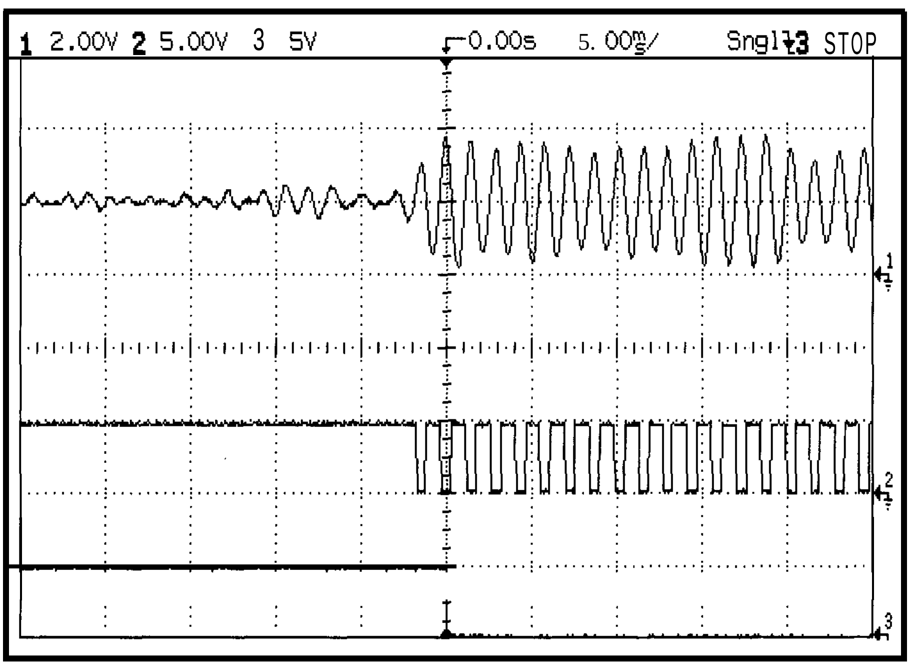


Photo 3-Trace 1 shows the filtered and amplified audio signal at U1 pin 14 and the digital outputs from two LM339 comparator sections. U2 pin 13, shown in Trace 2, produces one pulse for each cycle. When the signal's average peak level exceeds the threshold needed for good data, U2 pin 14 in Trace 3 goes low.

amp's filtered output. The low-level signals to the left of center are audible but do not trigger the comparators.

When the Morse-code pulse begins, Trace 2 goes low as pin **14** of U2c detects each cycle's peak. True zero-crossing detection would be better because detecting any other part of the signal introduces frequency errors as the amplitude varies. In this case, typical signals remains fairly constant during an audio cycle.

After Cl 1 charges up, U2c pin **13** drops, as shown in Trace 3, indicating a valid signal. If that signal remains low during the entire measurement, the CPU can be reasonably sure that the input signal represents a valid pulse.

The analog experts among you can have a great time improving these simple circuits. Nothing is particularly critical, as we are working with nice signals in a clean environment.

RELEASE NOTES

Although I don't have room for the entire schematic this month, the source and hex files are on the Circuit Cellar BBS. The Atmel 89C1051 resembles a generic 805 1 closely enough that the code will work on either one without change. You can even plunk Zerobeat into an EPROM and run it on an 8031!

Next month, we'll see how an assembly-language program can benefit from routines borrowed from a C runtime library. Homework: figure out how to convert a period in microseconds into a frequency in hertz.

PRODUCT INFORMATION

You can get blank 89C 105 1 microcontrollers directly from Atmel's distributors, although some have prohibitive minimum orders. Check your phone book for the Marshall Industries, Milgray Electronics, or Arrow/Schweber Electronics office nearest you.

The Atmel microcontroller databook provides all the specs and appnotes you need to build a simple PC parallel-port programmer.

I have a small stash of 89C 105 1 chips programmed with the Zerobeat firmware. Send a check or money order (no credit cards or COD) for $10 to Pure Unobtainium [see Sources]. Outside North America, send $17 by money order or check drawn in U.S. dollars.

If you're tired of canonical raw aluminum project boxes, get SESCOM's *Constructor's Hardware for the '90s* catalog. And finally, Dunfield Development Systems produces Micro-C for a wide variety of microcontrollers.

Ed Nisley (KE4ZNU), as Nisley Micro Engineering, makes small computers do amazing things. He's also a member of Circuit Cellar INK's engineering staff. You may reach him at ed.nisley@circellar.com or enisley@ibm.net.

SOURCES

Atmel 89C1051
Atmel Microcontroller Databook
Atmel Corp.
2125 O'Nel Dr.
San Jose, CA 95131
(408) 441-0311
Fax: **(408) 436-4200**
http://www.atmel.com/

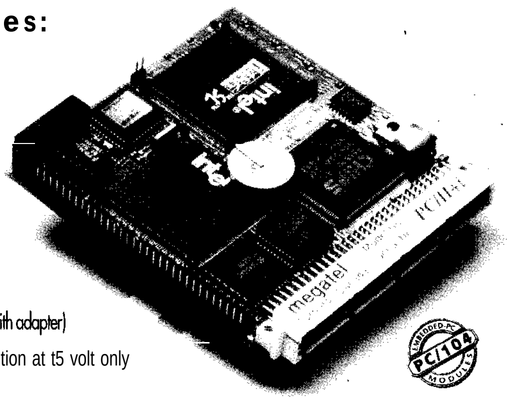
89C1051 with **Zerobeat**
Pure Unobtainium
13 109 Old Creedmoor Rd.
Raleigh, NC 27613

SESCOM ET-1
Constructor's Hardware for the 90s
SESCOM
2100 Ward Dr.
Henderson, NV 89015-4249
(702) 565-3400
Fax: (702) 565-4828

Micro-C
Dunfield Development Systems
P.O. Box 31044
Nepean, ON
Canada K2B 8S8
(613) 256-5820
Fax: (613) 256-5821
http://www.dunfield.com/

IRS

413 Very Useful
414 Moderately Useful
415 Not Useful



Nonintrusive Current Monitoring

FROM THE BENCH

Jeff Bachiochi

This series begins with winding toroidal current sensors for monitoring power consumption in the home. Jeff designs front-end signal conditioning with an expandable multiplexer for use with the low-cost Domino microcomputer.

It was the 1964 World's Fair in New York city. General Electric presented a showcase of technology, "The Carousel of progress." This was the first time I'd seen audioanimatronics.

Human robots came to life using cam-actuated hydraulics-a crude, if you will, programmable logic controller. These tireless actors performed on a revolving stage, in which the seating rather than the stage moved.

Each scene let you eavesdrop on how a typical family was affected by the technology available to them. As each decade passed, life became more comfortable, all thanks to your friends at.. .ahem.. .General Electric. GE's predictions of the future were quite good-we were going to have a cleaner, better, more efficient life.

The exhibit was quite a success. In fact, even if you missed it at the World's Fair, there's a good chance you have seen it or will see it in the future. Walt Disney World scooped it up and presents it as an attraction in the Magic Kingdom's Tomorrowland.

And, how close are we to fulfilling GE's predictions? Well.. .we are making progress in some areas. We have more electrical appliances providing many more services. And, recently, there's been an emphasis on getting appliances to operate more efficiently.

Despite the improved comfort and efficiency of home automation, we've been slow accepting it.

Utility companies are beginning to experiment with load shedding, so they can control when certain appliances can be used. You may someday be limited in just how warm or cold you can have your environment. Scary thought!

Will you or I see the day when appliances work together in a networked environment where electrical power is time-sliced into a happy average to avoid demand peaks? Probably not.

However, on an individual basis, we can control the future by paying closer attention to what goes on in our own homes. We can start today by gathering up past electricity bills and reviewing consumption.

You might notice yearly trends as seasonal shifts occur. Or, you may notice a consistent rise in costs as usage and costs per kilowatt hour go up.

What you can't see are short-term trends like day/night, home/away, or weekday/weekends. To see this kind of data requires more than just pulling out the (hopefully) paid utility bills.

NONINTRUSIVE APPROACH

Attempting to place current-measuring devices on every appliance in my house would certainly be a lengthy project, even if you didn't count the time required to string the necessary

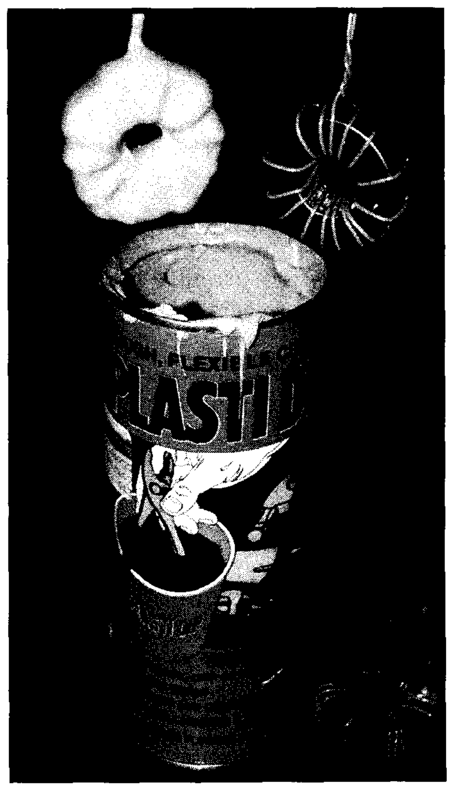


Photo I--Because the *ferrite is conductive, I coated the* toroid assembly using a plastic dip process, com*pletely* cowering *each unit.*

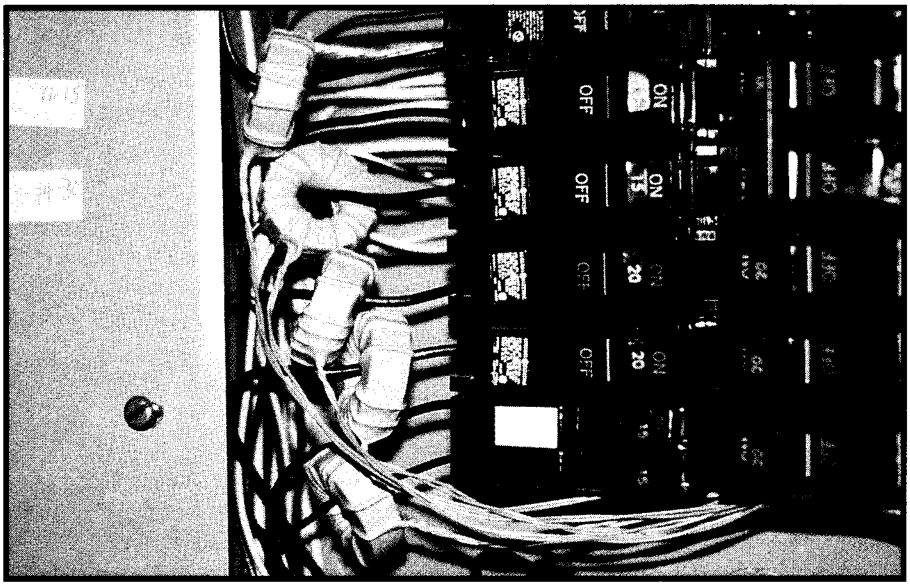


Photo 2—There is ample room for all toroids in the service panel. I labeled each with the associated breaker number for easy recognition.

web of data-acquisition paraphernalia. Although not as accurate, there's a fairly easy solution that works for everyone.

Use the breaker box in the basement as a centralized means of data collection, even though it limits us to individual circuits, rather than appliances.

Fortunately, most of us know which appliances are on each circuit. And, if not, it's easy to deduce via each breaker switch. Inside the door of every service panel is a chart of each breaker and a blank space to write in the appliances on that breaker.

How many of you have this filled in? In the past, I just looked for the tripped breaker and wasn't too concerned with what circuit was switched by which breaker. But now, I thank the electricians for doing a thorough job in filling in all those little blanks.

I hope you won't take this the wrong way, but I feel the need to remind you to use a full cup of caution if you try this project. When opening the service panel, you expose yourself to live 110-/220-V AC wiring. Turn off the mains—the 220-V legs coming in from the electric meter—and carefully turn off or remove each circuit's breaker before handling each branch. Failure to disconnect power may cost you your life.

The most widely used method of sensing current in a circuit is to put a known resistance in series with the load. This method is a bit tricky to perform in the main breaker panel, not to mention the fact that the electric code frowns on it. It also doesn't offer any kind of isolation from the power line.

Last month, I talked about some of the advantages of using ferrite as a foundation for inductors and transformers. Let's borrow the toroid transformer for this project.

As discovered over a century ago, every conductor carrying a current produces a magnetic field around the conductor. If the conductor passes through a ring of ferrite, the ferrite becomes temporarily magnetized by the field. The strength of the magnetic field is proportional to the current passing through the conductor.

When you have a multiturn secondary coil on the toroid, the magnetized ferrite induces a current in the secondary coil. This current is proportional to the first as long as the toroid core is not in saturation (i.e., exposed to more of a magnetic field than it can handle).

The big advantage of this current-sensor style is isolation. Not only are the voltages and currents small, they are in no way physically connected to the power lines.

See Figure 1 for the circuit I reproduced for each breaker in my 100-A service panel. To prevent unexpected spikes from destroying front-end components, I used a low-voltage MOV. It squashes voltages in excess of -10 V.

A Schottky diode is used to half-wave rectify the 60-Hz sine wave with as little drop as possible. The 470-µF capacitor holds the voltage to a average level, removing most of the ripple.

To match the toroid's output for the full-scale 5-V ADC input value, a variable-gain stage is set between the 1-kΩ load and the ADC. Each channel can be individually tweaked to match the

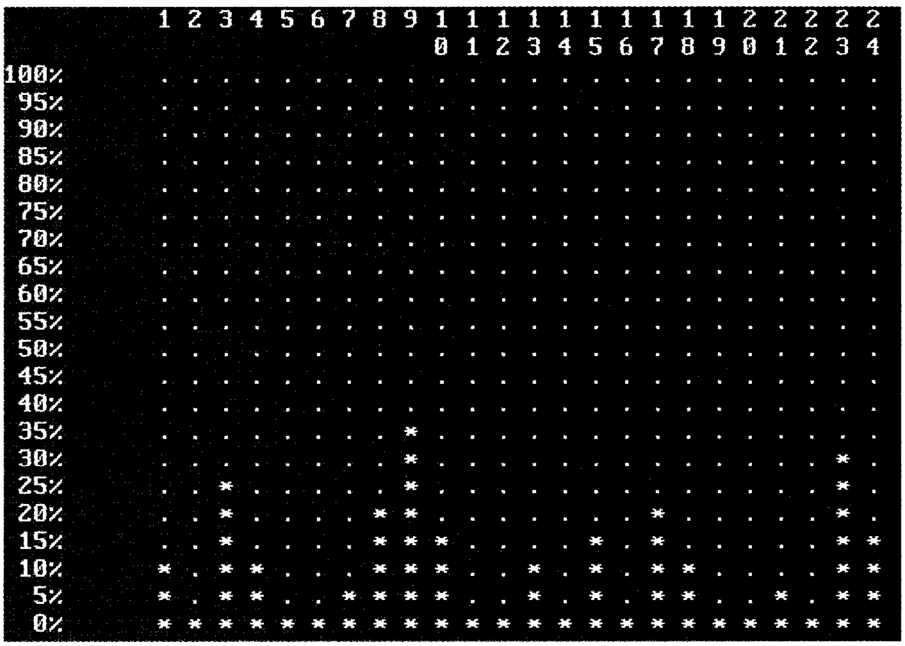


Photo 3—The real energy profile is displayed on a dumb terminal. Each breaker is labeled across the top with vertical bars indicating the percentage of maximum current presently being used.

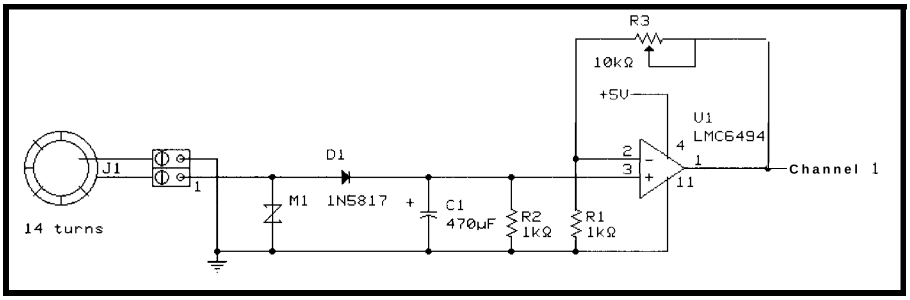


Figure 1—*Using 14 turns of 24-gauge teflon-coated wire with this signal conditioning circuitry enabled me to use the same foroid for 15-, 20-, 30-, and 40-A circuits by tweaking the gain for 5.00 V out at maximum circuit current*

capacity of the circuit breaker it's monitoring.

The toroids used for this project were hand-wound on cores purchased from Marlin P. Jones. The 0.75" ID (1.5" OD) of the toroid leaves plenty of room to pass through any size wire after a primary layer is wound.

Since the permeability of the material was unknown, I needed to experiment to determine the right number of turns for each circuit. I used 30-AWG wire-wrap wire and placed a single layer-about 120 turns-on a toroid. Now I needed a relatively high current device to use as a standard.

Janice, our managing editor, who seems to have cold feet all winter, uses an electric space heater to keep the frost off her tootsies. Well, since the snows have melted and temperature is on the rise, I didn't think she'd mind me borrowing the heater.

This little heater draws 8.8 A on our 120-V line. When I placed my hand-wound toroid on one leg of the heater, I measured about 4.8 V on my input circuit. I need about five more turns to achieve a full scale 5-V reading.

Based on a full scale, each toroid turn was producing about 40 mV:

$$\frac{V_{full\ scale}}{N_{turns}} = \frac{5}{125} = 0.04$$

Since this is based on an 8.8-A load, we need to know what each turn produces per ampere. You determine this by:

$$\frac{40\,mV}{8.8} = 4.5\,\frac{mV}{t}$$

Multiplying each breaker size by 4.5 mV, we come up with a new voltage per turn for each breaker's rating. The proper winding size, based on each breaker's rating, is calculated by:

$$N_{turns} = \frac{5\,V\,(\text{full scale})}{V\left(\frac{V}{\text{turn}}\right)}$$

As you see in Table 1, each breaker can be monitored up to the maximum allowable current, yet present only a O-5-V input for the ADC.

Each amplifier's gain is adjustable from 2 to 11 times. From this, you can see that a toroid wound with about I4 turns looks like -2.5 V with 40 A of current running through the (one-turn) primary.

With the amplifier, we can adjust it for a gain of 2 for a full-scale reading. In addition, the same toroid can be used for 30, 20, and 15 A by adjusting the gain to 2.7, 4, and 5.3, respectively.

Each toroid took about five minutes to wind. Since the number of turns is relatively small, I could deal with a couple of yards of wire without it getting all knotted up.

I had a spool of 22-AWG teflon-coated wire. This wire worked well because it is flexible and has a tough durable insulation layer. I left 30" leads on each toroid so the circuitry could remain outside the service panel.

After winding each toroid, I added a covering of tubing over the pair of secondary leads. I dunked each toroid into plastic coating material-the kind you dip tool handles in-as shown in Photo 1.

This covering completely insulates the toroid from anything it may come in contact with inside the breaker box. It gives three levels of insulation: the insulation covering the current carrying conductor, the dipped plastic, and the secondary's insulation (teflon).

Most of the general circuits in our homes are 15-A circuits, while heavier appliance areas, such as the kitchen outlets, are 20-A circuits. In my home, the dishwasher has a dual 15-A circuit, the clothes dryer takes twin 30-A breakers, and the stove uses twin 40-A breakers. The mains are 100 A, which is the minimum for most residential areas. (Prior to my addition, the house had 60-A service.)

Even though my hand-wound toroids all have an equal number of secondary turns, each has to be matched and calibrated to the circuit it will be used on. This matching and calibration is accomplished by passing a known (or measured) current through each matched toroid/amplifier circuit.

The amplifier's gain is adjusted so that a full scale (i.e., 5-V output) equals the breaker's trip-current rating. For instance, by using a space heater with a rating of 1250 W (i.e., measuring 8.8 A), the amplifier's gain can be adjusted to give 3 V (8.8/15 A) out of the amplifier.

When used on a 15-A circuit with 15 A being drawn, this amplifier's output is 5 V (i.e., full scale). This type of calibration is done for each of the 21 circuits. Photo 2 shows the service panel with the toroids installed. Now, we must collect the data.

DOMINO BUILDING BLOCK

I chose to use the Domino as the data collection's microprocessor (see Figure 2). The Domino is a complete microcontroller containing RAM, flash program storage, an optional 2-channel 12-bit ADC, 12 digital I/O bits, and an RS-232A, RS-422, and RS-485 interface-all in about two square inches.

Breaker Size	Voltage per Turn	Toroid Turns
15 A	67.5 mV/t	74
20 A	90 mV/t	55
30 A	135 mV/t	37
40 A	180 mV/t	27

Table I-For *maximum resolution, each toroid can be custom wound.*

The big advantage for me is the built-in BASIC interpreter which enables me to quickly write and debug the collection program without com-

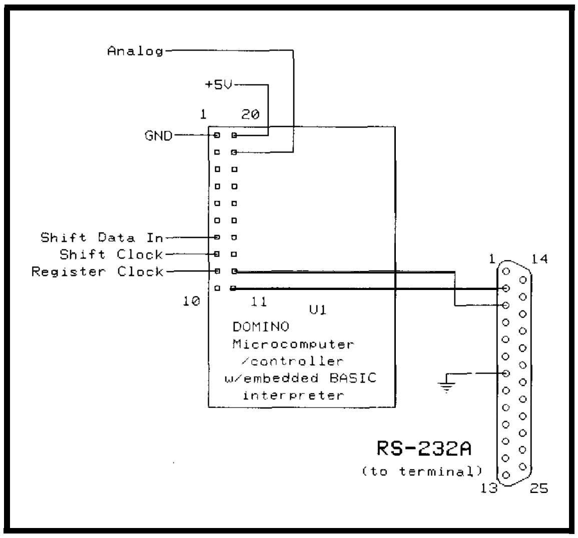


Figure 2—*The* Domino *microcontroller sequences the multiplexer, reads 12-bit samples, and outputs serial data with simple BASIC commands, all in less than 2 in.*[2]

piling or using assembly tools. However, Domino does not have a 24-channel ADC, which this project requires. (You generally won't find more than 8 channels of ADC on any micro.)

In this case, one of Domino's two channels is sufficient if I pad the front end with *4066* analog switches. These switches make for an inexpensive front end. But now, I need lots of digital outputs to control all those analog switches.

My first thought was to use three 1-of-8 demultiplexers, but I decided to go with serial-in parallel-out shift registers. Why? Expandability.

The front-end channels are built in groups of eight to a PCB. These can be stacked together to form an expanding front end (in groups of eight). Although I presently require only 21 input channels to cover all the circuits in my breaker panel, I can expand the system at any time to include more channels without requiring additional I/O pins.

Let's see how the hardware and software work together to provide expandability.

The shift registers require only three output lines to control any number of channels as shown in Figure 3. Two output lines-the clock and the data-shift the data (all 0s and a single 1 positioned to enable the required channel) through the daisy-chained registers.

The third output line clocks the shifted data into the register's output latches. Each time I want to change the channel, the shift routine shifts 24 bits out and then enables all output register latches at once.

To increase the number of channels, you need to extend the daisy chain and increase the number of shifts to cover all the registers on the chain.

The 4066 analog switches have rather poor (high) on resistances, especially at lower V_{SS} voltages. The channel's input amplifier reduces the effect of the analog switch's on resistance to the input impedance of the ADC. If you must have the circuit run entirely on 5 V, you'll pay a premium for special op-amps which swing rail to rail.

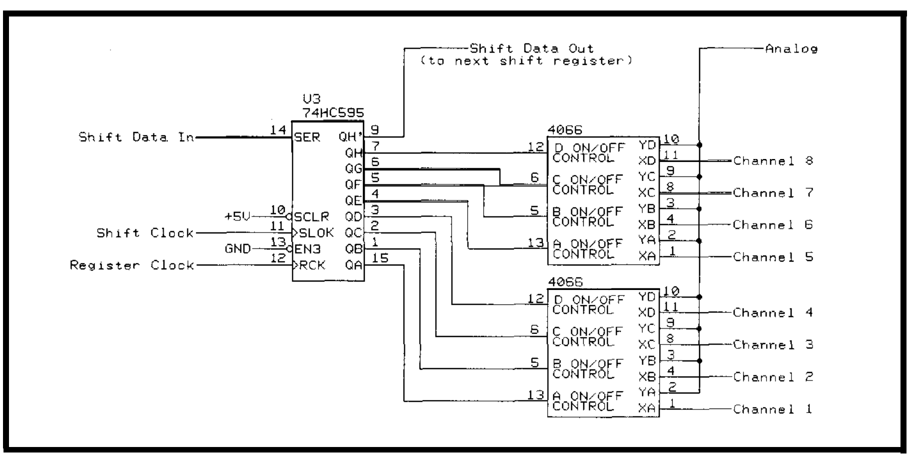


Figure 3—*The multiplexer circuitry is easily expandable. Each of the three boards contains this circuitry plus eight signal conditioners for a total of 24 monitored circuits.*

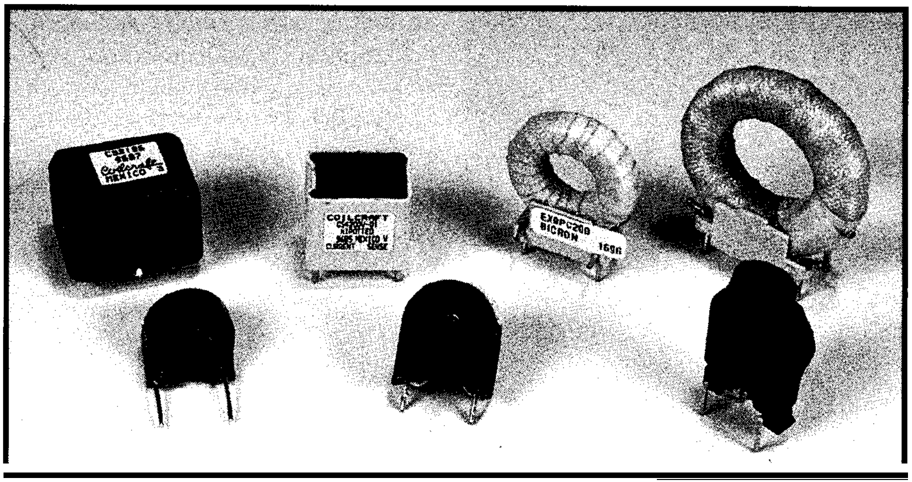


Photo 4—*Prewound coils are available from many inductor manufacturers. These are purchased by maximum current (e.g., I-5 A, I-20 A, and i-100 A).*

Data collection needn't be simultaneous for all channels. There's no real need for speed. In fact, although Domino has 32 KB of data RAM, I'll be storing about 1200 pieces of data an hour. So, I'll use an external storage device—a laptop.

The initial program written for Domino is more elaborate than necessary (to find out what's happening with the software, come back for next month's Part 2). This debug program cycles through each channel calculating the current of each probe (based on the channel's front-end gain) and displays the samples in a table directly to an RS-232 terminal (see Photo 3).

DON'T FEEL LIKE WINDING?

Prewound and calibrated toroids are available from many of the same manufacturers who make ferrite power transformers and chokes. Many of these current transformers are designed to be inserted into PC boards.

The single-turn primary is built into the part. It is designed to be placed in series with the conductor (PCB trace) which is to be monitored.

In Photo 4, you see a design kit available from Coilcraft. The kit contains both current transformers with and without a primary. (Those transformers without a primary have a hole through the center in which to pass the conductor to be monitored.)

Next time, we'll spend some time on the data-logging aspect and delve into a bit of analysis on the power-demand signature I've logged. ❑

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

Ferrite Toroids
Marlin P. Jones and Assoc., Inc.
P.O. Box 12685
Lake Park, FL 33403-0685
(561) 848-8236
Fax: (561) 844-8764

Current-Sensors Designer's Kit
Coilcraft
1102 Silver Lake Rd.
Cary, IL 60013
(708) 639-2361
Fax: (708) 639-1469

Current Sensors
Bicron Electronics
50 Barlow St.
Canaan, CT 06018
(860) 824-5 125
Fax: (860) 824-l 137

Domino
Micromint, Inc.
4 Park St.
Vernon, CT 06066
(860) 871-6170
Fax: (860) 872-2204

416 Very Useful
417 Moderately Useful
418 Not Useful

Spring Fling

SILICON UPDATE

Tom Cantrell

With spring comes trade shows—and lots of them! Through Tom, we get to walk through the Embedded Systems Conference East, Sensors Expo, and the PCI spring show. Come catch the highlights.

Once spring has sprung, trade show season kicks into high gear. I managed to hit three of them and believe me, there's plenty to report.

April started with a trip to the Embedded System Conference East in Boston to give a presentation ("Choosing An Embedded RISC"). I give an updated version of that talk and another ("Low-Cost, Easy To Use BASIC SBCs") at ESC West in September—stop by and say hi if you're there.

I then moved on to Anaheim, site of the Sensors Expo. I try to hit this show periodically, especially to monitor the fighting for trench positions on the analog/digital frontier. Besides, once you get your fill of the technical stuff, you can go to Disneyland.

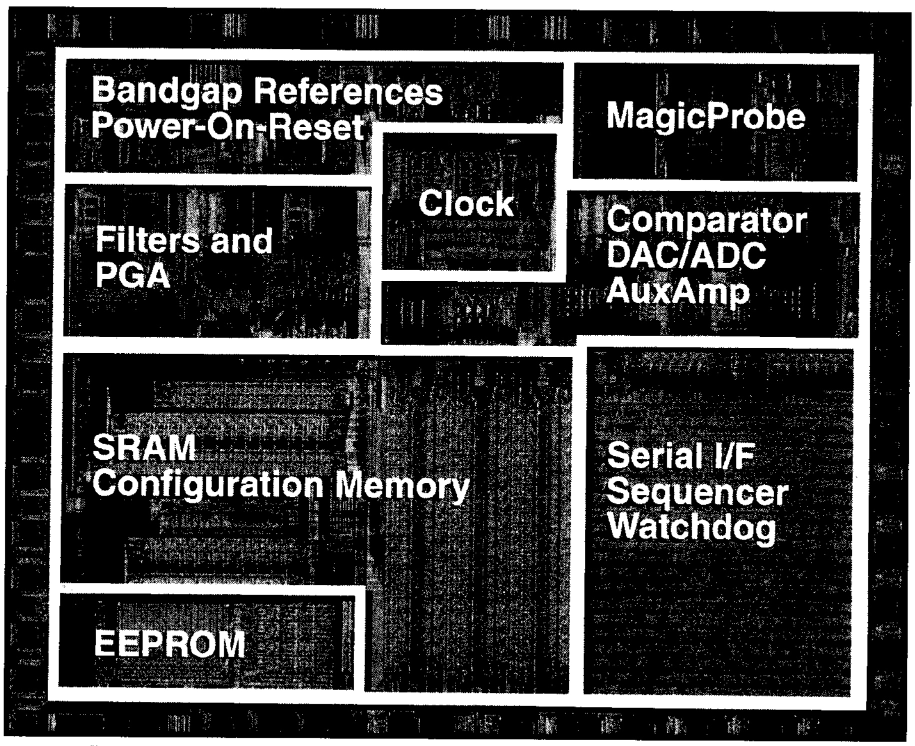
Finally home, just in time to drop in on the PCI Spring show. In case you've been on another planet, PCI is the unquestioned successor to the ISA throne, having ably dispatched lesser contenders such as EISA and MCA. Beyond the desktop (of particular interest to *INK* readers), PCI is having an impact on the embedded world as well.

WAKE UP AND SMELL THE JAVA

The Embedded Systems Conference always seems caught in an identity crisis. The exhibit floor has enough hardware to keep me happy, and several sessions have a hands-on bent. However, many classes seem targeted at professional programmers.

I'm basically a hardware guy-a lot of this stuff is over my head. However, I came away with the distinct impression that controversy is brewing in the programming world. Set the stage with the fact that the majority of embedded-systems programmers are only now migrating in mass to C, and there still is-always will be-some assembler.

Even as they barely digest C, they're worried about whether they can or



'Photo 1—*The* new IMP *50E30* combines the *conditioning front end (filter and* programmable *gain) and dual EEPROM (boot) and SRAM (in-system programming) configuration of the earlier 50E10. But, if goes digital by adding an ADC (composed of a comparator and DAC) with automatic channel scanning and limit checking.*

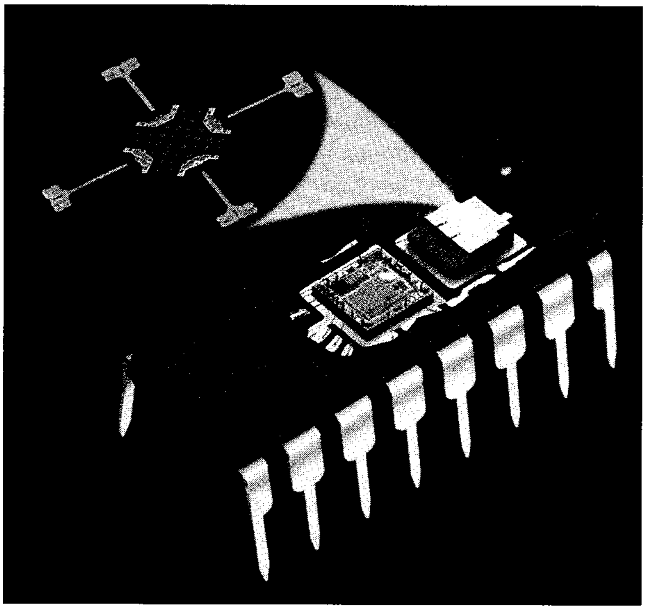


Photo 2—*Motorola, with their micromachined capacitive accelerometers, is just one of the companies pursuing a myriad of new low-G applications.*

should move to C++. Unfortunately, the purported benefits of the language and object orientation aren't overwhelmingly apparent, especially for control applications.

A cynic *(moi?)* could argue that the potential benefits of C++ are more than offset by the actual problems. Many problems are associated with the dynamic features of the language such as object binding, creation, and destruction. Besides code bloat and slowdown, there are grim prospects for writing real-time code in C++.

Part of the problem is that the term "real time" is rather loosely bandied about. Especially with the performance-at-any-price crowd, real time seems to mean "as fast as possible" or at least "faster than the competition.".

Most interpret real time in terms of meeting deadlines. Few seem to recognize the importance of what's sometimes called hard real time (i.e., the ability to handle or generate events with exact and perfectly repeatable timing).

The latest hot-button is Java, which is-best as I can tell-kind of a C++– (C+?). A bunch of the murkier stuff (e.g., pointer arithmetic, operator overloading, and multiple inheritance) is deleted. You get a leaner, meaner language, purportedly better suited for embedded and real-time applications.

However, I wonder if these HLL folks really get it when they cavalierly note that a minimum Java system requires 4 MB of RAM and may consume 30–50% of the CPU cycles handling what's left of the object orientation (a garbage collection of dynamically allocated memory). Gee, hope your mission-critical micro isn't busy housecleaning when the !@#$ hits the fan.

All in all, the whole C/C++/Java brouhaha makes me glad I don't have to program for a living.

HIT THE FLOOR RUNNING

Back on the exhibit floor, I was glad to see that at least silicon is marching on. It continues to deliver more for less across the application spectrum.

At the low-end, Microchip is putting the pedal to the metal with the popular PIC line. It seems like only yesterday I wrote "Cheap Chips-Lean and Mean PIC Machines" (INK 24) about their then-obscure and somewhat quirky chip. Since then, sales have rocketed to $250M+, and the PIC still occupies a unique price, performance, and features niche.

At the show, Microchip announced two variations of their popular 18-pin penny pinchers. The 16C710 and '711 are derivatives of the popular 16C71, which is a 1 K (x 14-bit instructions), 36-byte RAM OTP part with a 4-channel, 8-bit A/D converter.

The new parts tweak the memory complement-the '710 has 512 K x 14 bits and 36 bytes, and the '711 has 1 K x 14 and 68 bytes. Oh, and by the way, add brownout protection with low V_{CC} detection and automatic reset. Prices are $3-4 for low volume (e.g., 1000 pieces) and probably half that in high volume.

Another company that impresses me by packing a lot of value in a few transistors is Maxim. Since Maxim chips seem to proliferate like bunnies, it's hard to do more than present a laundry list of their latest parts.

Fortunately, Maxim is known, and should be appreciated, for their customer-friendly approach. Give them a call for literature, samples, and EV kits.

The popular single-supply RS-232 transceiver lineup has been upgraded with 3.3-V versions, optional 15-kV ESD (Electrostatic Discharge) protection, and automatic powerdown. The chip detects whether anything is plugged in and turns itself off if not.

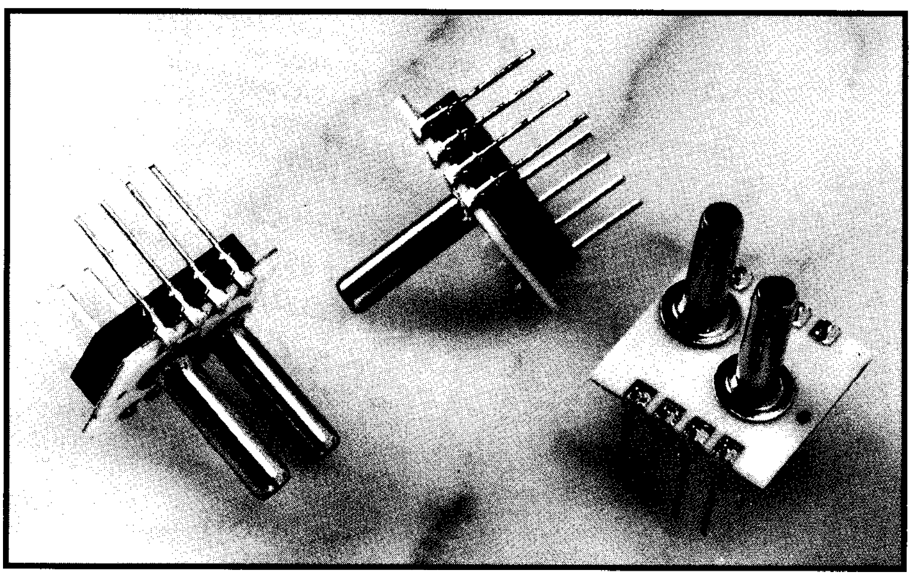


Photo 3-*These DIP-mountable pressure sensors from Silicon Microstructures feature built-in temperature compensation calibrated (by laser trimming) at the factory.*

On the RS-485 front, faster (up to 2.5 Mbps), stronger (longer cable, more nodes), lower power (20-µA) variants of the industry standard 75 17x-75 18x chips are available with 15-kV ESD protection. Maxim offers the MAX-1480 module that combines a driver, receiver, and 1500-V isolation circuitry all for under $10 in volume.

In the interface department, Maxim expands their extensive ADC lineup with a dizzying array of parts covering **8-18** bits of resolution and 50k–500M samples per second.

The MAX196 is an interesting part which at least on the surface is a rather conventional 8-channel multiplexed 12-bit ADC. However, they've integrated a programmable-gain amplifier, so each channel independently handles ±10-,±5-,10-, or 5-V inputs.

Another outfit that's rocketed to popularity in a niche is Information Storage Devices. Thanks to the ability to store multilevels (instead of just 0 or 1) in a single memory cell, their talking chips ("Talking Chips," ***INK 36)*** put low-end digital voice recorders in everything from toys to cards.

Their latest chip, the ISD33000, shown in Figure 1, runs at 3 V. It's ideal for portable applications like cellular phones. Message capacity is up to 2 min. at 8 kHz (4 min. at a lower-fidelity 4 kHz). Also, the original chip's mixed bag of control lines (admittedly, allowing push-button-only designs) is replaced with a more micro-friendly serial bus.

Having a professional interest in embedded RISC chips, I checked out the latest offerings and tried to get a sense of who's hot, who's not, and which way the wind is blowing.

One recent bombshell was AMD's decision to publicly "Dr. Jack" the 29k. Basically, management said the company is going to focus on 'x86 chips not only on the desktop, but in the embedded world as well.

Engaging in a little spin control, AMD now says they plan to support the 29k until the end of the century. It still sounds to me like a polite way of saying, "Yep, it's dead."

Competitors-like IDT, with their *29k* ***Survival*** Guide-were quick to offer their assistance to orphaned 29k

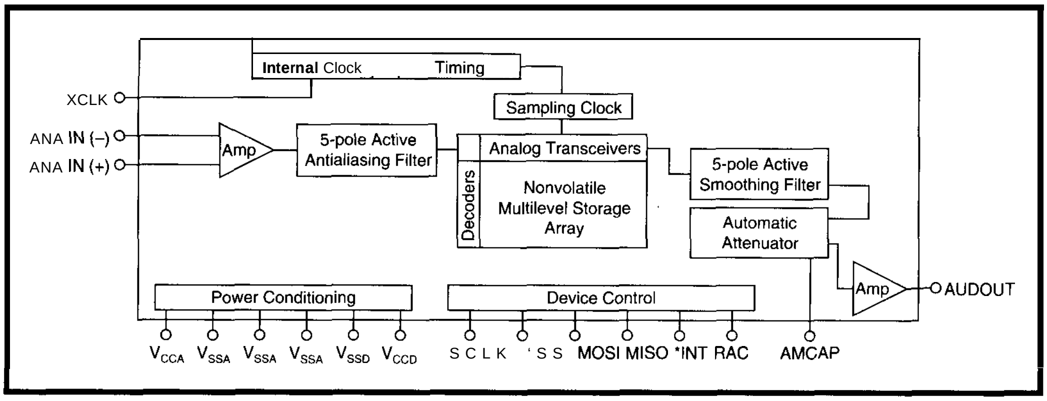


Figure l-For the latest talking chip *from ISD, capacity is up (2-4 min.) and voltage is down (3 V).*

customers. IDT continues to be one of the most aggressive embedded RISC promoters, with over 25 different MIPS-based controllers covering the spectrum from $10 to 64-bit.

While high-end RISCs have lots of sizzle, most of the steak is at the low end, with cold RISCs like the Hitachi SH and ARM starting to fill high-volume sockets. Hitachi claims over 600 design wins for the SH. ARM, which remains along with MIPS the most open and multisourced architecture, continues to show up in leading- (sometimes bleeding-) edge products like PDAs and set-top boxes.

On the other hand, prospects for the PowerPC seem decidedly murky. Perhaps this reflects the uncertainty on the desktop, with Apple on the ropes and IBM sticking with 'x86 for PCs.

Recent reorganization at Motorola seems to have pushed the PowerPC into the computer group, leaving 68xx(x) in the controller group.

Any doubts about Intel's embedded strategy weren't soothed by their absence from the show. In light of AMD's 29k move, one must ask if the '960 isn't subject to the same fate.

Is their recent introduction of a '151, a sped-up '51, tacit admission that the more ambitious '25 1 ("Plan 251 From Outer Space-Intel's 8xC-251SB," *INK 56)* is a hard sell? Has anyone noticed that the Intel embedded group is about 'x86 chips?

If there's anything approaching the controversy on the software front, it's the intricate dance being played between suppliers of CPUs and DSPs as the functions merge onto a single chip.

Each side wants to cover the other base. Witness Analog Devices who simultaneously licensed in a CPU architecture (H8, from Hitachi) and licensed out their DSP core to AMD.

I did see some software I liked. One clever tool you're likely to run into (it's licensed to a number of chip and tool companies) is Aisys's Driveway. You've probably discovered by now that, especially for the latest high-integration chips, mastering the bits-and-bytes of the on-chip I/O isn't easy.

Driveway, running under Windows, integrates a knowledge base for a particular chip's (e.g., '51s and PIC) peripherals including documentation and C drivers to configure and access the I/O. It isn't as whizzy as Java, but for ~$lOO-500, depending on version and bundling promotions, Driveway seems like a worthwhile timesaver.

Taking off from snowy New England and ending up at the Disneyland Hotel on the same day puts a special edge on jet lag. Fortunately, a lot of neat stuff at the Sensors Expo distracted me.

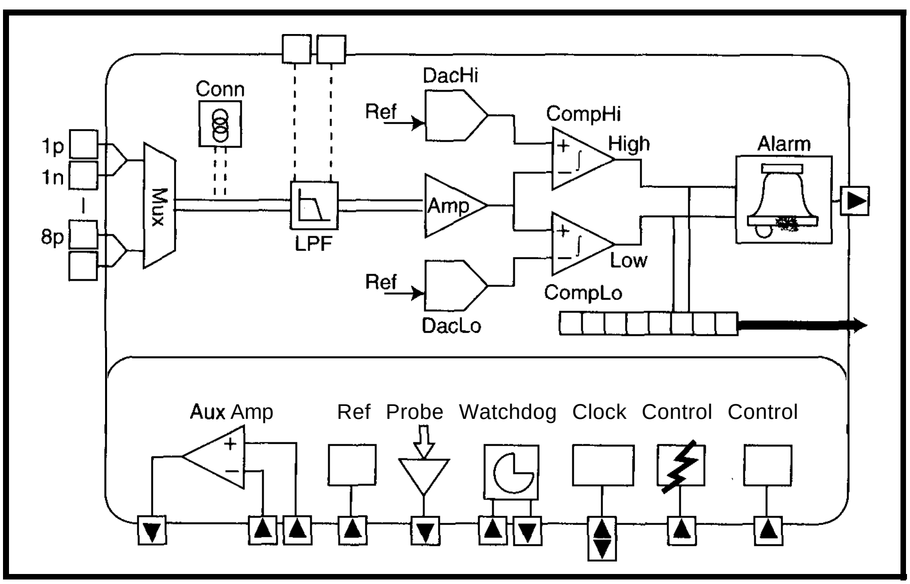
I've mentioned the digilog-merging of digital and analog-concept before. Action in the sessions and on the exhibit floor demonstrates that trend is clearly on the rise.

A good example is the latest member of IMP's EPAC family. You may remember that the earlier 50E10 ("EPAC Epoch," *INK 58)* was kind of an analog PLD with on-chip digitally programmable amps, filters, and so on. Despite the digital underpinnings, the 'El0 is still an analog-in-analog-out chip.

The new 50E30, shown in Figure 2 and Photo 1, steps further in the digital direction by incorporating an 8-bit ADC. Designed to monitor a variety [up to 16 single-ended, 8 differential) of diverse inputs at high speed, the chip also includes automatic scanning logic and comparators that signal a fault.

It also has a connectivity check mode that, by injecting a small current and measuring the voltage shift, checks the input impedance, and thus the integrity, of the connection.

On a less grand scale, the merging of digital and analog can be seen in a simple thermostat, the AD22015 from Analog Devices. As shown in Figure 3,



2—The 50E30the digilog 8-/16- (differential/single-ended) ADCpwith digital the device combines a temperature sensor (*-40°C* to +150°C with 2°C accuracy) and a resistor-programmed setpoint comparator with automatic hysteresis (-4°C).

Yes, there are lots of ways to implement a simple thermostat. But if that's all you need, few can compete with the sub-$1 volume pricing of this puppy.

Thanks to the automotive airbag arms race, accelerometers continue to be a hot item. The emergence of easy-to-use low-G units promises to expand the market into active suspension, hard-disk shock protection, physiological monitoring, elevators, and virtual reality.

For instance, AMP offers the ACH-LN-20 with a outstanding 5-V/G sensitivity. It handles high-frequency inputs (up to 5 kHz). Since it's a piezoelectric unit, it can't handle DC (i.e., 0.5 Hz minimum). So, it's best for monitoring continual vibration.

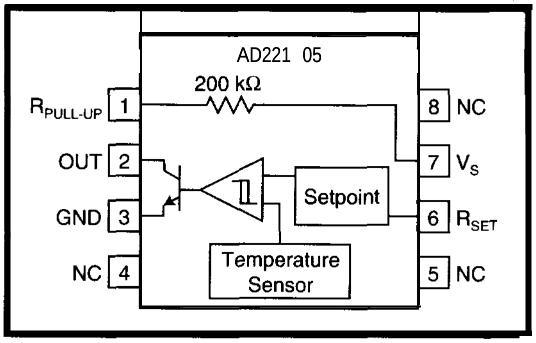


Figure 3-*The Analog Devices Thermostat IC doesn't do much, but the price is right.*

For DC applications, both Texas Instruments and Motorola have capacitive micromachined accelerometers. TI's unit achieves an admirable 1 -V/G sensitivity, and Motorola's new Senseon lineup, shown in Photo 2, integrates signal conditioning, filtering, and complete self-test capability.

Silicon Microstructures (recently purchased by Exar), an established purveyor of just such micro-friendly acclerometers ("Saab Story-A Tale of Speed and Acceleration," *INK* 57), is expanding into the pressure sensor market with the SM5600 series.

The devices, available in constant-current and constant-voltage excitation versions, feature laser-trimmed calibration and temperature compensation. Models with 100 PSI are complemented by remarkably low-range (e.g., O-O.3 PSI) units.

The dark side to digital sensors is the unwieldy plethora of standard, semistandard, and proprietary interfaces. The IEEE P145 1 "Transducer to Microprocessor" standard currently under development may proffer hope.

As shown by HP's Stan P. Woods, chairman of the IEEE working group, the 9-pin interface combines the wire-

SOURCES

PCI Show
Conference Pros
P.O. Box 2451
Saratoga, CA 95070
800-35 1-6000
Fax: (408) 526-9195
http://www.annabooks.com/

Driveway
Aisys Ltd.
P.O. Box 10041
Petach-Tikva 49001
Israel
(+972 3) 922-6860
Fax: (+972 3) 922-6863
http://www.aisys-usa.com/

SH
Hitachi America, Ltd.
Semiconductor and IC Div.
2000 Sierra Point Pkwy.
Brisbane, CA 94005-18 19
(415) 589-4207
Fax: (415) 583-4207

ARM
VLSI Technology
18375 South River Pkwy.
Tempe, AZ 85284
(602) 752-6630
Fax: (602) 752-6001

29k Survival Guide
Integrated Device Technology
3236 Scott Blvd.
Santa Clara, CA 95054
(408) 727-6116
Fax: (408) 988-3029

ISD33000
Information Storage Devices
2045 Hamilton Ave.
San Jose, CA 95125
(408) 369-2400
Fax: (408) 369-2422

MAX1480, **MAX196**
Maxim Integrated Products
120 San Gabriel Dr.
Sunnyvale, CA 94086
(408) 737-7600
Fax: (408) 737-7194

16710, 16711
Microchip Technology, Inc.
2355 W. Chandler Blvd.
Chandler, AZ 85224-6199
(602) 786-7200
Fax: (602) 899-9210

Sensors **Expo**
Expocon Mgmt. Assoc., Inc.
363 Reef Rd.
Fairfield, CT 06430
(203) 256-4700, x131
http://www.expocon.com/

ACH-LN-20
AMP, Inc.
P.O. Box 3608
Harrisburg, PA 17105
(215) 666-3500
Fax: (215) 666-3509

AD22015
Analog Devices
P.O. Box 9106
Norwood, MA 02062-9 106
(617) 329-4700
Fax: (617) 326-8703

ADS7833
Burr-Brown Corp.
6730 S. Tucson Blvd.
Tucson, AZ 85706
(520) 746-l 111
Fax: (502) 889-1510

50E30
IMP
2830 N. First St.
San Jose, CA 95 134-2071
(408) 432-9100
Fax: (408) 434-0335

Senseon, MDZ208
Motorola
5005 E. McDowell Rd.
Phoenix, AZ 85008
(602) 244-4556
Fax: (602) 952-4067

SM5600
Silicon Microstructures, Inc.
46725 Fremont Blvd.
Fremont, CA 94538
(510) 490-5010
Fax: (510) 490-1119

Embedded Systems Conference
Miller Freeman, Inc.
600 Harrison St.
San Francisco, CA 94107
(415) 905-2354
Fax: (415) 905-2220
http://www.embedded.com/conf.html

Compact **PCI**
Ziatech
3433 Roberto Ct.
San Luis Obispo, CA 93401
(805) 541-0488
Fax: (805) 541-5088

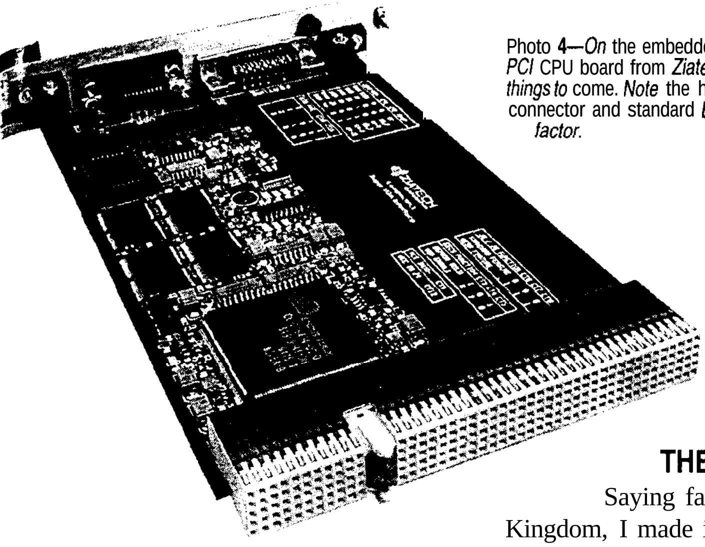


Photo 4—*On* the embedded PC front, this Compact *PCI* CPU board from *Ziatech* may be fhe shape of *things to* come. *Note* the high-rel pin-and-socket connector and standard *Eurocard (3U* and *6U)* form *factor.*

thriftiness of a serial bus with the easy access of dedicated select, trigger, and interrupt lines (see Figure 4).

One interesting part of the spec is the so-called TEDS (Transducer Electronic Data Sheet). By capsulizing the features and operating characteristics of a particular sensor, it seems to offer the potential for an industrial equivalent of plug and play.

THE MAGIC BUS

Saying farewell to the Magic Kingdom, I made it to Silicon Valley just in time for the PCI Spring show.

I'm sure you're aware of, if not using, the PCI bus in your desktop. After a rocky start, PCI quickly established itself as the PC standard bus.

It sent previous pretenders EISA and MCA to join S-100, Multibus, and ("perpetually in the") Futurebus to that great bus depot in the sky.

Maybe it's a little odd to have a show about a bus, but the vertical organization pulls all the key pieces (connector, chip, board, box, development tools, etc.) into one place. Organized by the experts at Annabooks, the extensive sessions deliver the know-how to get all the pieces working.

I don't spend a lot of time at PC or computer shows, given my propensity to stick with stuff that works. All my columns are written on a five-year-old 68k Mac, and I've got a Brand-x '386 PC that I just (reluctantly) upgraded with Windows 3.1.

However, PC and computer shows often cover technologies that spill into the embedded arena. This turned out to be the case at the PCI show. Based on the session and exhibitor mix, you could call it the Embedded PCI show.

I got the feeling that PCI bus is already thought mature in the speed-of-light desktop world. I guess everyone who needs to know how to design a PCI graphic or disk board does.

It looks like the long-term trend is toward slotless PCs with the highest speed I/O confined to the motherboard and the rest divvied up among next-generation back-panel connections like

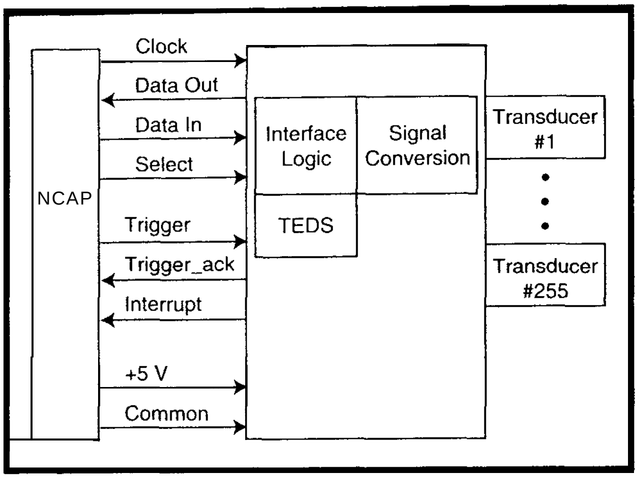


Figure 4-*The IEEE P1451 standard shows the digital connection between a Network Capable Application Processor (NCAP) and a sensor which contains its own Transducer Electronic Data Sheet (TEDS).*

USB and FireWire. You'll be hearing a lot more about them.

By contrast, embedded PCI activity is heating up on a variety of fronts. It's already penetrating the PC-In-A-Box and passive-backplane segments that rely on desktop technology. Besides accommodating standard graphics and disk adapters, PCI's speed (theoretically up to 132 MBps) is welcome in high-speed data-acquisition applications.

PCI systems go further than ISA-based PCs. ISA was technically and in a marketing sense (completely) tied to 'x86 CPUs. However, PCI has been adapted to a variety of non-'x86 chips like the X-bit RISCs (SPARC, PowerPC, MIPS, and Alpha).

Combining the embcddcd-PC concept with CPU choices spells possible trouble for the aging VME bus. There are certainly some PCI gotchas (lack of hot swap, number of slots limited to half a dozen or so without bridging), but if I were a VME supplier, I'd check over my shoulder frequently.

The most frenetic is in the small-form-factor arena where downsized variants map the PCI bus onto different form factors and connectors (see Photo 3).

One of the most interesting is the Compact PCI (see Photo 4), which uses standard 3U and 6U Eurocard format and a much hardier (gas-tight] pin-and-socket connector as opposed to the traditional desktop edgeboard.

Running out of time-and expertise-I'll hand off further coverage of this hot topic to the folks in the ***EPC*** section. I'm sure you'll see more PCI fireworks on the embedded front. •l

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 fax at (510)*** *657-5441.*

IRS

419 Very Useful
420 Moderately Useful
421 Not Useful

Photo courtesy of Octagon Systems

NPC

PC/ 104 DAC MODULE

The **PCM-D/A12-8** is a PC/l 04compatible 8-channel,12-bit D/A converter module that requires no user adjustments. Because there are no onboard potentiometers to tweak, setup of analog systems needing accurate digitally controlled voltages is quick and easy. This feature also eliminates the need for a technician or engineer to readjust or recalibrate the unit, saving you both time and money.

Based on the Analog Devices AD7237A DAC port, the device is a complete, dual 12-bit voltage-output D/A converter with output amplifiers and zener voltage reference on a monolithic CMOS chip. The singlechip design and small package size saves considerable space and increases reliability over multichip designs.

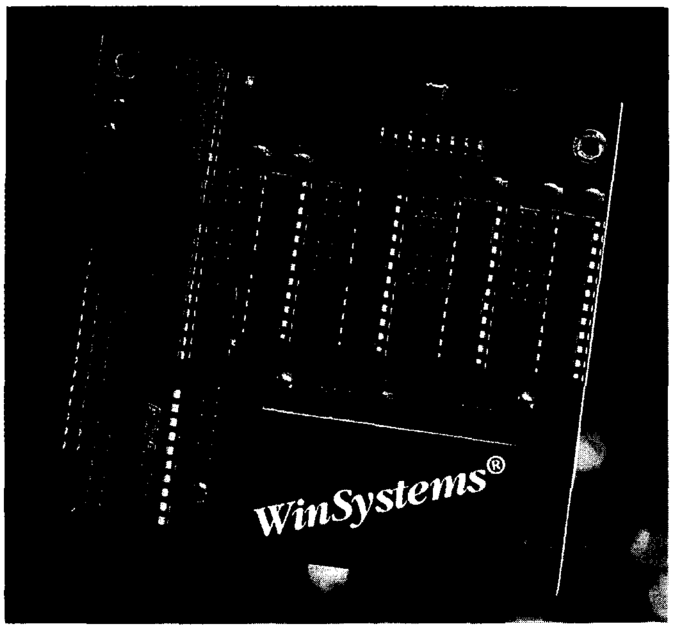
The PCM-D/A12-8 uses four AD7237As which provide eight independent channels. The output voltage ranges are 0 to +5 V, 0 to + 1 OV, and ± 10 V. All outputs are updated simultaneously. The output amplifiers are capable of developing + 10 V across a 2-k load to ground.

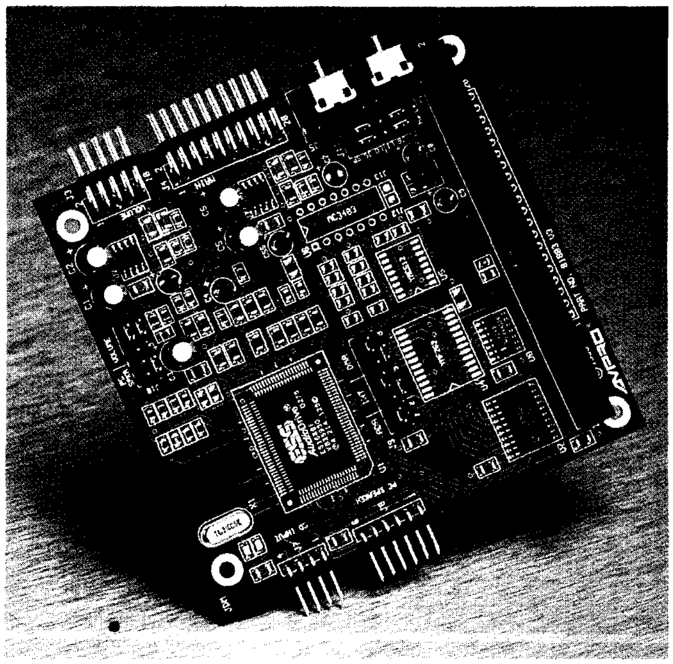
The PCM-D/A12-8 module is PC/l 04 compliantand measures 3.6″×3.8″. Itoperatesovera temperature range of –40° to +85°C and is available with or without an onboard DC/DC converter.

The PCM-D/A12-8 sells for $395. A depopulated version with only four channels is available for $295.

WinSystems, Inc.
715 Stadium Dr. • Arlington, TX 7601 l-6225
(817) 274-7553 • Fax: (817) 548-l 358

#510





STEREO SOUND FOR EMBEDDED SYSTEMS

The **Embedded** Sound Module (ESM) is a +5-V PC/104 module with full-frequency audio or music record-and-playback capability. It is ideal for audio alarms, status messages, or announcement-oriented applications such as vehicular systems, medical instrumentation, vending machines, and security systems.

The ESM includes a 16-bit ADC which samples at software-selectable rates from 5 to 44.1 kHz [CD data rates). It provides compression and decompression of WAV files. Both 8- and 16-bit record-and-playback modes are available.

The ESM accepts a number of audio sources through stereo line in, stereo CD in (connected to auxiliary in), mono microphone in, and mono PC speaker in. An onboard 6-channel mixer controls these inputs. Outputs include stereo line out, stereo amplified speaker out, and analog PC speaker out. It also contains a Yamaha OPL3-compatible 20-voice FM synthesizer. Speakers can be driven directly by a built-in 0.5-W stereo audio amplifier.

The ESM is hardware and software compatible with Soundblasterand Soundblaster Pro. Software drivers for Windows 3.1, Windows NT, and Windows 95 are included.

The ESM sells for $189 in 100 quantities. A development kit which accelerates system-integration efforts and includes the ESM, two disks, a hardware technical manual, and a software utilities manual is available for $275.

Ampro Computers, Inc.
990 Almanor Ave.
Sunnyvale, CA 94086
(408) 522-2 100
Fax: (408) 720-l 305

#511

NouveauPC

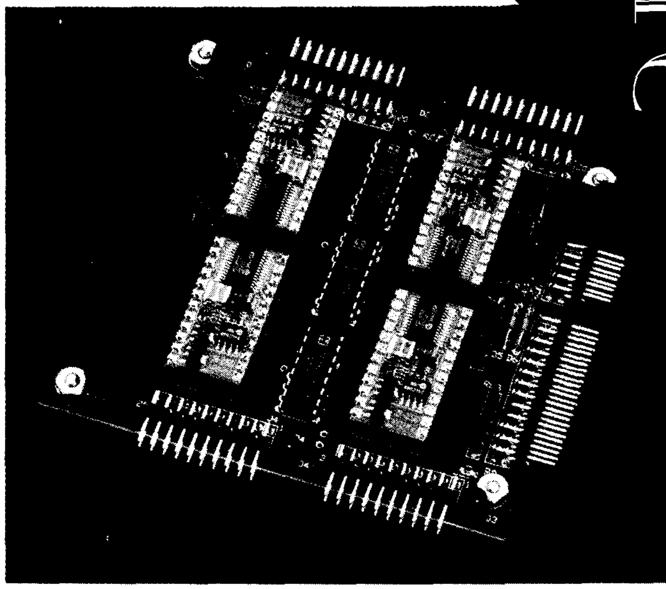
edited by Harv Weiner

PC/ 104 I/O BOARD

The PCStampII combines the Parallax Basic Stamp II (BS2-IC) microcontroller and the popular PC/104 form factor. The PCStampII makes an easy master-slave interface out of the BS2-IC and PC/104 by using up to four BASIC-programmable BS2-ICs for control.

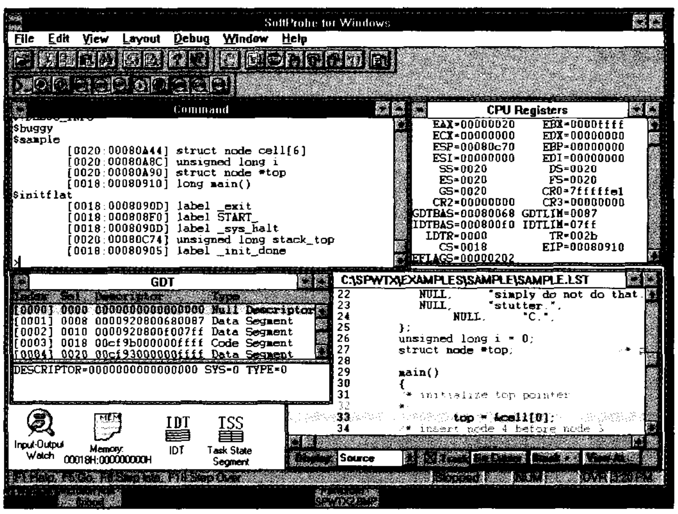
Each BS2-IC has 16 I/O lines, 50-kbps maximum speed, 2048 bytes of EEPROM space, and a maximum program execution speed of *4000* instructions per second. The PC/104 (or any master IBM PC) is used for programming, monitoring, and setting operational states of each BS2-IC. The PCStampII measures 3.6" x 3.8" and includes a 9-12-VDC power jack and PC/104 bus connector.

The PCStampII package sells for $179 and includes documentation, parallel and serial cables, and Parallax's custom software to switch between each BS2-IC. Example software shows how to control process system components (e.g., valves, pumps, sensors) and how to monitor signals from each BS2-IC.



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

#512



REMOTE DEBUGGER

SoftProbe for Windows Remote Debugger is a Windows-hosted graphically oriented source-level debugger for embedded development. A remote debugger enables you to download a program from the host PC and debug it on the target system.

Program execution on the target system is controlled by the target monitor. The target monitor communicates with the host via an RS-232 serial port.

SoftProbe comes with a number of preconfigured target monitors, as well as a target-monitor customization utility to enable users to build their own custom target monitor.

SoftProbe offers a flexible breakpoint facility and supports software breakpoints, debug register breakpoints, conditional breakpoints, and attached breakpoint actions.

The Debugger provides source-level and symbolic debugging for C. Some of the features and views provided by the software include a code window that shows code in source, disassembler, or mixed, and a memory window that shows memory in a variety of formats.

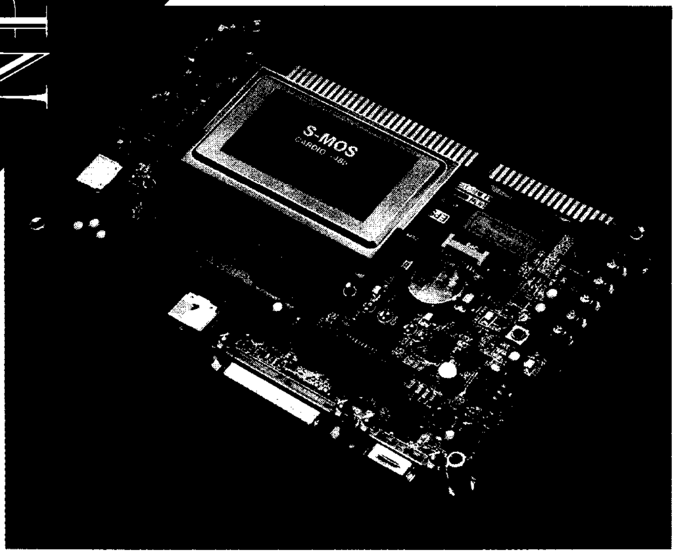
It also includes a quick-watch window to evaluate and view C expressions, a call-stack window that shows the active chain of function calls, and a locals window to show the active local variables. System-level views include GDT, IDT, LDT, TSS, and '386/'387 registers.

SoftProbe may be hosted on systems operating Windows 95, Windows NT, and Windows 3.x. The Debugger is supported by SSI's Link&Locate '386 absolute linker and locator, and it accepts files in the OMF-'86, '286, '386 boot-loadable formats along with the binary formats.

SoftProbe for Windows Remote Debugger sells **for $1795.**

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

#513



UNIVERSAL INTERFACE BOARD

A universal interface board with all prevailing, small form-factor interfaces including PC/ 104, PCMCIA, and ISA on a single board has been announced by S-MOS Systems. CardPresso is a half-size ISA board (6.7" x 4.2") that gives system designers and vendors immediate access to industry-standard interfaces, enabling the use of off-the-shelf products for embedded-systems design.

Using S-MOS **CardIO** as a separate engine, the all-on-one interface board provides a flexible and modular approach to system development and design. The end result is a significant savings in overall system size, development time, and cost.

CardPresso has two onboard PCMCIA 2.1 sockets and a standard PC/I 04 connector, a VGA CRT connection, and two RS-232C serial ports as well as the S-MOS CardIO socket. This combination gives designers the ability to customize and add value to a board or end-system design anytime during the life of the end product or product family.

For storage and memory considerations, CardPresso also has pin headers or connections for 3.5" and 5.25" floppy-disk drives and 2.5" hard-disk drives. Up to 4 MB of flash memory on CardPresso, and up to 16 MB of DRAM on S-MOS CardIO is also available.

CardPresso sells for $650 in single quantities, and the S-MOS Card10486 starts at $800 in 200 quantities.

S-MOS Systems, Inc.
2460 N. First St.
San Jose, CA 95131-1002
(408) **922-0200**
Fax: (408) 922-0238

#514

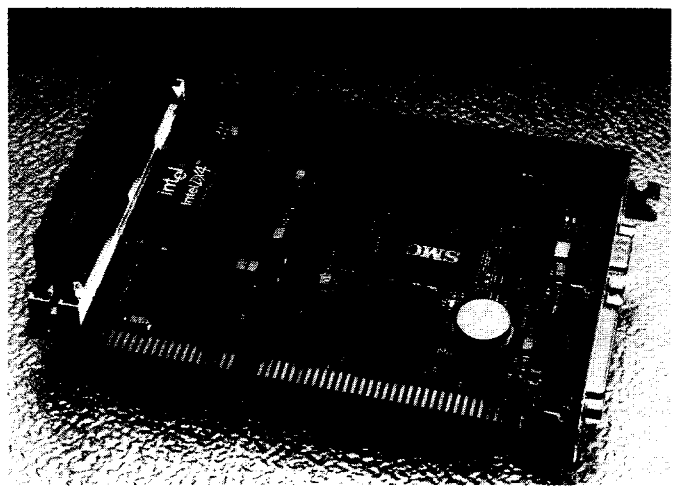
INDUSTRIAL SBC

The VIPer805 features 486DX2,'DX4 and '5x86 microprocessor designs and operates as a stand-alone unit on an ISA passive-backplane system. As the base of a variety of compact and ruggedized systems, VIPer805 is ideal in embedded applications requiring high reliability and performance. Typical applications include portable test and measurementequipment, mobile (in-vehicle) computer systems, point-of-sale terminals, and industrial portable computers.

The VIPer805 uses the plug-and-play compatible SMC FDC37C932 Ultra I/O controller chip which allows for high board-level integration. It comes standard with Local bus IDE support, keyboard and floppydisk controllers, and one parallel and two serial ports. The VIPer805 supports up to 4 MB of bootable flash memory and up to 128 MB of DRAM using two 72-pin SIMMs. Additional onboard integration is possible through the PC/104 expansion header.

The VIPer805 also includes an AMI BIOS with extended setup and power management support, watchdog timer, power-failure and low-battery detector, and support for two 2.88-MB floppy drives.

The VIPer805 in the 486-DX2/66 configuration sells for $742.



Teknor Industrial Computers, Inc
7900 Glades Rd. • **Boca Raton,** FL 33434
(407) 883-6191 • Fax: (407) 883-6690

#515

Brad Reed

$500 Internet Browser in a Box

Who hasn't become entangled in the World Wide Web? It's growing and changing every day as more people gain access. This $500 set-top box is going to launch another segment of the population. A real spider's feast.

Everyone's talking about the latest and greatest things on Internet. But, there's only one thing they all agree on. It's growing and changing rapidly.

The same is true for the Web. Hard-core surfers are embracing the best hardware and software in multimedia technology to take advantage of new developments.

While the dominant players in the browser arena are PC based and fairly well established, the face of the Web browser is still changing rapidly.

These changes will continue as more and more people venture onto the Web. But for now, only the "rich" and computer literate surf the electronic data wave.

THE MAGIC $500 MARK

Currently, there's a three-tier class distinction for venturing into cyberspace. Surfers have the computer hardware, modem, and access through an ISP. Beach bums have the hardware, but haven't made it in.

The third and largest group are the have nots. They don't have a computer, but they've heard enough that they may be interested. Still, theyaren'twilling to spend two grand to get set up. This third group is driving the much-touted Internet appliance down to a reasonable cost.

Experts agree that the price barrier for non-PC-literate to enter cyberspace is about $500. In this article, I'll explore what hardware and software it takes to get the price low enough.

In providing a $500 Internet browser, you need a hardware system with embedded software that gives everyone access to the Internet through a set-top box and a phone line. While cable modems may ultimately provide a better and higher bandwidth solution, more people have access to a telephone line than cable. Also, the local cable system may not have the infrastructure to support bidirectional cable traffic.

DEFINING THE SYSTEM

Because access to the Internet and the associated browsersoftware istightlylinked to PCs, use of an 'x86-based PC-compatible system is attractive. It simplifies the task of defining and creating the necessary hardware platform. So many software tools are readily available that the software-development task becomes much easier.

An Internet browser is somewhat unusual as an embedded system. Its performance is not so much defined by raw processing horsepower as by the speed of the communications link. It's better to save money and resources on a lower powered '386-class processor, especially if the cost differential is applied toward a higher speed modem link.

In general, mass storage is not required by a Web browser used for casual surfing. However, a small amount of nonvolatile storage should be available for bookmarks.

Adding a PCMCIA card slot is a particularly wise choice for this type of system. It saves the extra expense of a hard drive if more storage is needed to transfer data to the PC and it's difficult to purchase a hard drive that's small enough. Today, 500-MB hard drives are often the minimum.

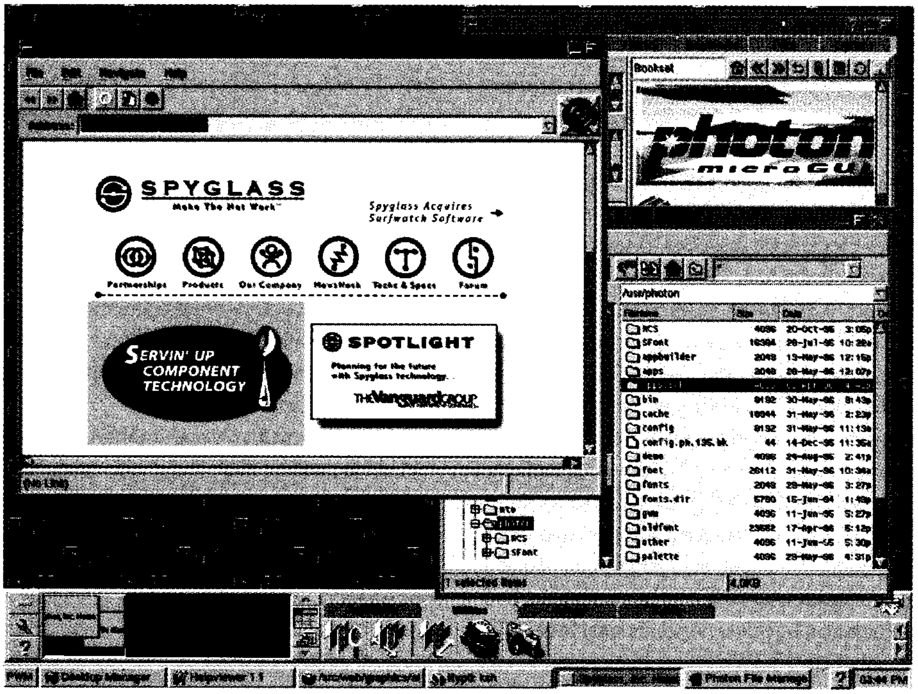


Photo 1: The **QNX** RTOS and Photon ***microGUI*** provide a real-time operating environment and complete ***graphical*** interface on an Intel **'386EX** in as little as ***1 MB*** of memory. This ***configuration shows*** the Spyglass Web browser, HTML-based Help Viewer, and ***Photon File*** Manager.

SET-TOP HARDWARE

The ultimate goal is a set-top box for the broad consumer market. Size and system cost surface as two main constraints. Let's consider a few cost-versus-feature tradeoffs.

As you move away from full PC compatibility, cost reduces. A real-time clock (RTC)-all but required in a PC—may not be necessary in an Internet appliance that can't save files.

You can also forego the keyboard and mouse controller. As a set-top box, a normal keyboard and mouse interface may not be the best choice of input device.

An infrared link alleviates a wire running across the living room from the keyboard to the box on top of the TV. And, a small, hand-held remote with a chiclet keypad certainly suffices for typing in a URL or search parameter.

While these changes may save money, software and BIOS issues can arise. Most PC-compatible BIOSs require an unexpected amount of modification to do away with an RTC and keyboard controller. While you *can* do it, it's not a task for the inexperienced or fainthearted.

DRAM cost is much less than six months ago, but the amount of DRAM significantly affects overall system cost. The amount of memory required is most directly affected by the selection of the OS and browser software.

FAST, LOW-OVERHEAD OS

By its very nature, the Web browser needs a graphical interface. Without being able to move the cursorwithinanimagemap to select a link, it's useless.

The ideal software configuration only requires l-2 MB of EPROM forcodestorageand 1–2 MB of DRAM. But, many **software** developers think megabytes are no object. So, how can you achieve this kind of svelte configuration and still have a GUI?

With Windows 3.1 -never mind Windows 95—you need a minimum of 4-MB DRAM and 8-l 0 MB of disk space. Add a Web browser, and thesystem weighs in far beyond the target configuration.

CHOOSING HARDWARE

We examined the hardware and software required for a Web browser. Now, let's look at a couple of real-world hardware platforms, configure the systems, and test surf them to see if the $500 goal is achievable.

Intel's EXPLRI board (INK71) meets the necessary criteria for a proof-of-concept platform. We want to test the software on a target platform with the same processor horsepower as the true target hardware.

The EXPLRI board is an Intel '386EX-based PC-compatible system that can run DOS and Windows 3.1 applications. It

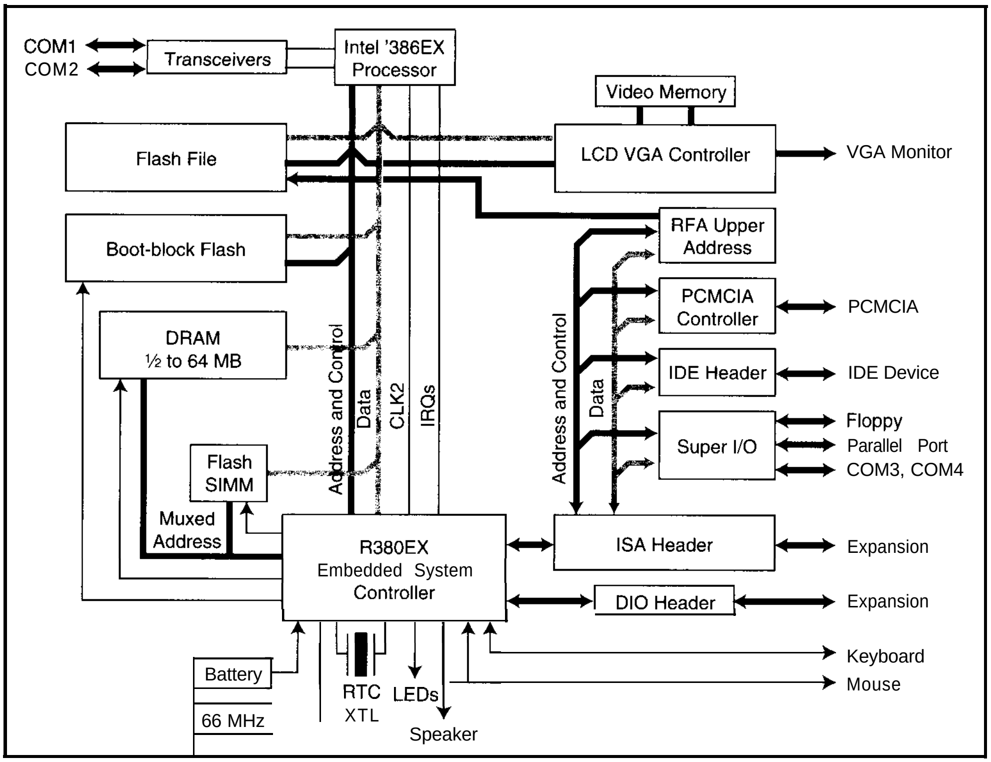


Figure 1: The block diagram of the Intel ***EXPLR2*** board shows the functional ***blocks of the*** system. The ***RadiSys R380EX Embedded System Controller provides the keyboard and mouse controller and RTC for full PC compatibility.***

ships with a 1 -MB SIMM that you can swap for a A-MB SIMM to run Windows. It even runs Windows 95 when appropriately configured with a 16-MB SIMM.

The board features a 25-MHz Intel '386EX processor and a RadiSys R300EX memory/bus controller. It also has a A-Mb boot-block flash memory, a socket for up to a 16-MB DRAM SIMM, and a single-slot Type I/II PCMCIA controller.

EXPLR1 operates as a stand-alone system with an onboard LCD VGA graphics controller, IDE controller, and PS/2-style mouse and keyboard. It has two serial ports, an RTC with 1 1 A bytes of battery-backed CMOS RAM, a watchdog timer (WDT), and a standard PC power-supply connector. Access to system bus signals and digital I/O (DIO) is provided via lwo expansion headers. And, its best feature—it only costs $386.

While it doesn't meet the diskless criterion of the end goal and has VGA rather than TV output, it still gets us in the ballpark with a small (5.5 in.') board with the right processor horsepower and starting price.

With this as the initial platform, we can turn our attention to software.

CHOOSING SOFTWARE

This next section chronicles the adventures I had with the Internet client services group at Hamilton Hallmark in configuring a workable system.

We first chose an experimental configuration to verify that the hardware had enough horsepower to be usable. EXPLRI is within normal PC configuration (if there is such a thing), so it's a relatively well understood configuration.

With an old, small AO-MB IDE drive loaded with Windows 3.1 and Netscape

QNX and Photon microGUI

QNX is a real-time, microkernel, POSIX-certified OS. It's a tiny microkernel-less than 32 KB--that manages interprocess communication between a team of cooperating processes providing higher-level OS services (e.g., networking, filesystems, and user interfaces).

Since higher-level services are implemented as processes that run in separate, MMU-protected address spaces, QNX's functionality can be scaled on-the-fly by starting and stopping processes as required (e.g., as PCMCIA modules are inserted and removed). This modular approach, depicted in Figure i, enables QNX to scale down for tiny ROM-based embedded systems-without giving up a POSIX API-or scale up into several-hundred-node distributed real-time systems.

QNX's use of the MMU to manage the OS and application processes results in a robust run-time environment. It enables QNX to be successful in mission-critical real-time applications (e.g., nuclear-reactor monitoring, medical instrumentation, and financial-transaction processing).

Photon microGUI offers a windowing system built on an architectural approach similar to the QNX OS for the Web-terminal application. Photon consists of a small, 45-KB windowing microkernel used by various optional processes to build a full-scale windowing system with surprising capabilities.

You can drag live, running applications from the screen of one network node to the screen of another-even if the two nodes use different operating systems. So, Photon's built-in windowing system in an embedded device can display within the windowing system of any network or modem-connected desktop computer.

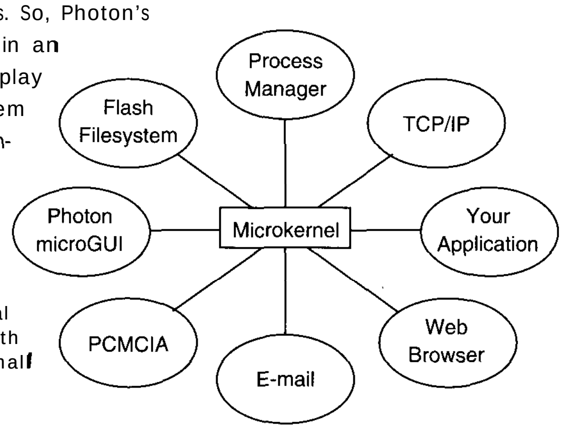


Figure i: Most ***QNX*** OS services are provided by optional modules that coopemte with each other through a small microkernel (10 *KB).*

and a relatively low-cost 14.4 PCMCIA modem, we assembled the system rather quickly. The necessary PCMCIA modem drivers and card and socket services got the system to play through the PCMCIA modem. We could have used an external modem, but then the PCMCIA socket would have gone to waste.

With this configuration, the system was surfable. But, it didn't really fall within the software parameters we had because a hard drive was needed to hold the relatively large amount of software.

As for cost, it nearly doubled the target cost. The EXPLRI board with 1 MB of DRAM was $386. We then added $80 for 4 MB, $300 for a new hard drive, $75 for the PCMCIA modem, plus power supply, keyboard, and mouse. We're at double cost without a display.

Next, please?

The aggressive innovators at Hamilton created a solution they call the "near $500 Internet appliance." The system uses EXPLR1, a hard drive, and a modem.

The system can be configured to use either a standard IDE hard drive and PCMCIA modem or an ATA hard drive in the PCMCIA socket and a standard modem. Or, if you don't have PCMCIA devices available, a standard IDE drive and modem can be used.

The software is where the real innovation starts. QNX configured a software system that uses its Photon microGUI and prototype WebGazer software.

QNX, a forward-thinking company, decided some time ago to standardize on HTML format for all their online manuals. As a result, they had a prebuilt HTML engine for displaying help files.

From there, it's relatively straightforward to build a full-blown Web browser by integrating the HTML help engine with their existing TCP/IP support. The result is the WebGazer prototype, shown in Photo 1.

While it's doubtful that WebGazer will ever be released as a commercial product, particularly in light of QNX's recent alliance with SpyGlass, it serves a valuable role in this proof-of-concept exercise.

QNX had the task of getting their OS to run on EXPLRI. Within 30 minutes, they had the necessary software installed on a 20-MB Sandisk PCMCIA memory card.

The Sandisk card contains all the components necessary to connect EXPLR 1 to a notebook computer running Windows 95.

In this arrangement, EXPLRI was working as a Web server! It went above and beyond expectations.

We were obviously on the road to getting the unit to function as a browser. The direct serial connection between the two machines gave a reliable connection and provided confidence in the serial hardware and driver support for the port.

The only pieces missing were the PPP deamon and the Watcom C++ compiler needed to compile the pieces. The 20-MB Sandisk couldn't handle the C++ compiler and PPP deamon.

To overcome this barrier, we chose a Simple Technology 170-MB Type III PCMCIA hard drive to provide greater storage yet maintain a small size. With a slight modification of EXPLR 1 's Type-II socket, the Type-III card fit.

The next and most formidable milestone was how to install the QNX software on a PCMCIA hard drive. EXPLRI doesn't have floppy support, so we couldn't just load disks. By using a notebook computer as a platform, we loaded software.

The first step in initializing the PCMCIA hard drive was to install all the software onto the notebook, including the QNX system, Photon, WebGazer, and TCP/IP.

Installation is quite straightforward, as long as you know the basic information about your notebook. The QNX installation scripts are well written. It went very well.

The QNX software works well in a multi-OS environment. The fd i s k utility provides multiple boot partitions, so DOS and QNX can reside happily on the same device during installation.

As an experiment, we tried Windows 95 with the EXPLRI board. It ran reasonably well when it was configured with 16 MB.

The process of putting Windows 95 on a 3.5" IDE device was simple compared to installing QNX on a floppiless system. The largest obstacle was installing the correct drivers. Try to get Windows 95 to install drivers for a PCMCIA device it doesn't see.

Within two days, we had it configured and operating in 16 MB of memory and with a PCMCIA 28.8-kbps modem.

The system surfed the Web, but it was nowhere near the $500 goal. We tried the configuration with 4 MB of memory, and to our surprise, it booted and loaded Windows 95.

So, it's true. Microsoft's claim that its software runs on a '386 with 4 MB of memory is real. The problem was with Netscape. The system ran, but was unusable in a real-world application. It was an interesting side trip, but we had to put the project back on course.

WEBGAZER SURFS US HOME

Once the softwarewas fully configured, EXPLR 1 with a 4-MB SIMMand the PCMCIA hard drive were connected to a 14.4-kbps modem.

The system's usability was significantly enhanced when we changed to a 28.8-kbps modem. This confirmed our hunch that the system is limited by the communications link, not processor speed.

The cost ended up well outside of the target, but the concept was proven.

The next task: how to make changes that bring us closer to the price goal? We zeroed in on three areas: program storage, DRAM, and the modem.

Grant Courville of QNX suggested we might get by with 2 MB of DRAM and 2 MB of flash and no hard drive by creating the QNX, Photon, and WebGazer system on another hostand creating an imagewhich could be burned into a flash-memory device. This change substantially reduced cost.

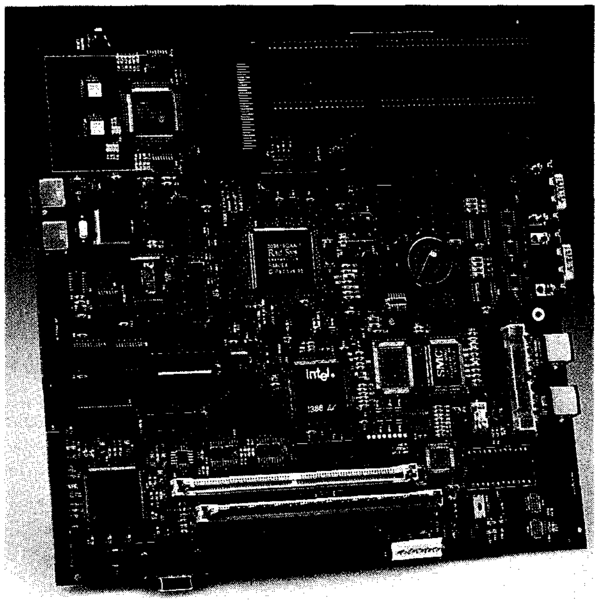


Photo 2: The ***Intel EXPLR2*** board is based on the ***RadiSys R380EX*** Evaluation board. It's uniquely designed for easy connection to test equipment. In addition to the multiple logic analyzer connection headers, the processor has pads for soldering ***in pins for connecting*** to a ***Microtek Emulator.***

THE NEXT STEP: EXPLR2

While reviewing the features of the new EXPLR2 board (pictured in Photo 2), Intel mentioned that QNX had asked to be included in the flash. While the Web browser won't be in the configuration to be shipped, between the 512-KB boot-block flash device and the 2-MB Resident Flash Array (RFA), there's plenty of room to test a full Web browser.

As you can see in Figure 1, the core of EXPLR2 centerson the 33-MHz Intel '386EX processor and the RadiSys R380EX embedded system controller. These coupled with the DRAM and BIOS in flash give you the complete core of an embedded PC.

Intel's '386EX has most PC system functions built in. It includes four of the common PC peripherals: two DMA channels, three 82C54-compatible counter/timer channels, two 82C59A interrupt controllers, and two 16450-compatible UARTs.

The R380EX rounds out the core by adding the DRAM controller supporting 512 KB to 64 MB of EDO or FPM DRAM, RTC, keyboard and mouse controller, IDE interface, and ISA controller.

To this base, EXPLR2 adds a 2-MB RFA, PCMCIA controller, VGA controller, and a Super I/O chip with parallel-port and floppy-disk controller. It has easy system expansion in the form of two ISA slots and one PC/I 04 connector.

CLOSING IN ON A PERFECT FIT

The next step in proving the feasibility **of** the browser is working with QNX to get SpyGlass running out of the RFA on EXPLR2 using only a 2-MB DRAM SIMM.

With the software in place, our attention turns back to the hardware. Three issues remain unresolved: NTSC video output, a modem, and final cost-reduction steps.

Both EXPLR 1 and EXPLR2 use the Cirrus Logic CL-GD6245 LCD VGA controller chip that drives both standard VGA displays and LCD panels. While this is a desirable characteristic for an embedded PC, it adds no value in this particular application.

What we need is the CL-GD5425-TV True-Color VGA Controller with TV output. By replacing the GD6245 with the GD5425-TV and adding an Analog Devices AD722 Video Encoder, we achieve the ultimate goal.

The final validation step will be to use a CL-GD5425-TV VGA/NTSC video board plugged into one of the EXPLR2's ISA slots to demonstrate the capability to either use a regular VGA monitor or output the NTSC video to TV.

With the current shifts in modem performance and cost, it's difficult to settle on a modem. Rather than being locked into a maximum transfer rate, we'll take advantage of the EXPLR boards' PCMCIA slots.

By using the PCMCIA slot for the modem, the user can upgrade communications by justchanging the card. Total system cost also doesn't affect modem speed.

With this final proof-of-concept hardware and software configuration, we've only have to eliminate unneeded components. Parts of the EXPLR2 reference design can be eliminated in final production. In particular, the Super I/O chip is unnecessary because its features aren't required.

ACHIEVING THE GOAL

So, you see, it's possible to build a $500 Internet Web-browser set-top box. It's technically feasible and a quick calculation based on the EXPLR2's bill of materials shows that the system comes in well under $400. Using EXPLR2 as a base design along with QNX software proves to be a viable base system for building an under-$500 internet browser. EPC

Brad Reed works as a component products application ***engineer*** at Radisys. Although he worked on embedded designs at ***Tektronix, his familiarity with embedded PC design launched*** when he joined first Microtek and then Radisys. He may be reached at *brad.reed@radisys.com.*

SOURCES

"Near-$500 Internet appliance" App. note
Hamilton Hallmark Technical Support Ctr.
http://www.hh.avnet.com/specials.html#internet

AD772 Video Encoder
Analog Devices, Inc.
One Technology Way
Norwood, MA 02062
(617) 329.4700
Fax: (617) 326.8703
http://www.analog.com/

CL-GD6245 LCD VGA, CL-**GD5425-TV**
Cirrus logic, Inc.
3 100 W. Warren Ave.
Fremont, CA 94538
(510) 623-8300
Fax: (5 10) 226-2 180
http://www.cirrus.com/

EXPLR1 and EXPLR2
Intel Corp.
500 W. Chandler Blvd.
Chandler, AZ 85226
(800) 5484725
Fax: (708) 296-3699
http://www.intel.com/

Photon **microGUI, WebGazer**
QNX Software Systems Ltd.
175 Terence Matthews Cres.
Kanata, ON Canada K2M 1 W8
(613) 591-0931
Fax: (613) 591-3579
http://www.qnx.com/

R380EX Embedded System Controller
RadiSys Corp.
15025 SW Koll Pkwy.
Beaverton, OR 97006-6056
(503) 646-I 800
Fax: (503) 646-I 850
BBS: (503) 646-8290
http://www.radisys.com/

I R S

422 Very Useful
423 Moderately Useful
424 Not Useful

Scott Baisch

Kevin Smith

WinLight

Part 2: The Application

Programming for WinLight or **Windows** *requires development tools and techniques we may not be so familiar with. Here's a real-world clock application program and the tools needed to write, test, and debug it.*

Last *Embedded* PC (INK 71), we discussed various concepts behind developing an embedded graphical system requiring only 256 KB of ROM or flash memory and 256 KB of RAM.

In this installment, we take WinLight a step further, showing you how it significantly **alters** one's approach to design. We look at the fundamental difference between an application for WinLight/Windows versus a nongraphical program.

We'll show how to develop a WinLight/Windows application for an embedded system. Although there are basic differences between Windows and Winlight, remember the structure and content of a WinLight application is essentially the same as that of a Windows program.

Consequently, a program that runs under WinLight also runs under Windows and appears the same on the display. The reverse may not be true since a program written specifically for Windows could call on features not supported by the much smaller and efficient Wintight.

PROGRAMMING GUIs

Programming applications for WinLight, Windows, or almost any environment pro viding a graphical user interface (GUI) differs greatly programming applications that run directly on a text-mode operating system such as MS-DOS.

In the latter program, calls to the OS are made as needed. These calls are for services, such as those required to output data to the display, request keystroke input, write and read data to and from disk storage, and so on.

If you're used to thinking of an application in this manner (i.e., it **calls** the OS to get inputs from other devices or users), then an eventdriven graphical program looks a little strangeat first. Several excellent books written on Windows programming explain the concepts in far more detail than is practical to do here.

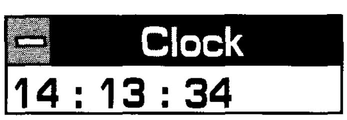


Figure 1: Although it doesn't look ***like a*** typical window, ***the*** clock window is real and comes ***complete*** with ***a*** command ***menu.***

In a graphical environment like Windows or Wintight, the application program runs in a window or in multiple windows. While WinLight or Windows displays the windows in which the different parts of the application appear, it's the application that determines the look and attributes of its windows.

Another seemingly bizarre concept is that the OS makes calls to the application. That's right-Wintight requests the application to do certain things. These requests are most often in response to input from a user or some other device in the system.

Why this shift? Because WinLight is a multitasking OS. When WinLight receives input, such as from a keyboard, it determineswhichrunningapplication (i.e.,which specific window) the input belongs to and notifies that window's application of the event. It's up to the application to perform specific actions relating to the event.

Listing 1: The *WinMain* function is the **entry** point in any **WinLight** or Windows program. Without *WinMain*, the operating **system** never loads your program.

```
#include <windows.h>
#include <dos.h>

// Borland and Microsoft use slightly different structure names
#if __BORLANDC__
  #define DOSTIME struct dostime_t
#else
  #define DOSTIME struct _dostime_t
#endif

// determine the width of font character via this formula
#define GetSysFontCharWidth() (LOWORD(GetDialogBaseUnits()))

// prototype all functions to make sure they are used correctly
void DisplayCurrentTime(HWND hWnd);
long FAR PASCAL _export ClockWndProc (HWND, UINT, UINT, LONG);

// WinMain is the entry point called by WinLight/Windows.
// When WinMain returns, the app is terminated
int PASCAL WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    LPSTR lpszCmdParam, int nCmdShow)
```

This OS-to-application interaction differs greatly from the traditional technique of putting the application in a loop polling for a specific event. WinLight passes these event notifications to the application via messages. You see this message-passing mechanism in a clock application program. Note how the application program communicates with the OS.

CLOCK APPLICATION PROGRAM

This sample program, while small, includes all the basic elements of a WinLight application. It is written in Borland C and has been tested with Borland C/C++ 4.5 and Microsoft Visual C++ 1.0. When the program runs under Windows or Wintight, the user sees the clock shown in Figure 1. The time display updates every second.

Listings l-6 show the C code for this program, stored in a file named CLOCK. C. The code should be entered into a single source file in the order shown.

Listing 1 shows the header files that are needed. A `#define` statement determines the size of the system font provided by WinLight/Windows, the function prototypes, and the `WinMain` function. The comments identify these lines of code.

All WinLight/Windows programs must contain a `WinMain` function as the established entry point for the program. The variable names offer clues to their purpose or content. For example, `lp` is a long pointer, `sz` is a zero-terminated string, and `h` is a handle. This conventional notation in Windows programming is useful when reviewing someone else's code.

The next part of our clock program declares several variables and a window class structure. Before a window can be created, its window class must be established. The window-class structure, shown in Listing 2, is filled in and then registered by means of a call to `RegisterClass`.

A window is always based on a window class. If the class is not already registered, a call to `RegisterClass` is necessary.

Registering the window allows WinLight to identify the window for the purpose of passing messages to it. Remember, WinLight passes events and requests to the applicationvia messages. These messages must find the appropriate window.

The next piece of our program, shown in Listing 3, does four things:

- defines the window width so it holds at least 10 characters of the currently used system font
- defines the window height to be twice that of the title bar
- creates the window with a call to `CreateWindow`, passing the attributes needed to display the window
- calls several functions to create the window on the display and insert the current time in it

`ShowWindow` and `UpdateWindow` are provided by Wintight, while `DisplayCurrentTime` is implemented in the sample program. Recall that `DisplayCurrentTime` was prototyped in Listing 1. We'll discuss this function more later.

So far, our clock program starts at its entry point and makes calls to register, create, and display its window before placing the current time in it. However, within one second, the displayed time is out-of-date unless it's updated regularly.

The message loop and window procedure come into play here. The message loop, shown in Listing 4, retrieves messages from WinLight that are destined for one or more of the windows previously registered by a program. Although our clock program has only one window, we must use this standard mechanism for getting messages from Winlight.

The message loop retrieves messages from WinLight and sends them back again via a message structure pointed to by `&msg`. This passing back may seem unusual. However, it's essential for directing messages to the appropriate window when there are multiple windows on the display.

Listing 2: Your program must call *RegisterClass* after defining a window. Failure to do so means no event messages get back through to this part of your program.

```
static char szAppName[] = "Clock":
WNDCLASS  wndclass;
int       width:
int       height:
HWND      hWnd;
MSG       msg:

if (!hPrevInstance){
  // set up a window class with standard APPLICATION icon.
  // ARROW cursor, and WHITE background
    wndclass.style              0;
    wndclass.lpfnWndProc       = ClockWndProc;
    wndclass.cbClsExtra        = 0;
    wndclass.cbWndExtra        = 0;
    wndclass.hInstance         = hInstance;
    wndclass.hIcon             = LoadIcon(NULL, IDI_APPLICATION);
    wndclass.hCursor           = LoadCursor(NULL, IDC_ARROW);
    wndclass.hbrBackground     = GetStockObject(WHITE_BRUSH);
    wndclass.lpszMenuName      = NULL:
    wndclass.lpszClassName     = szAppName;

    RegisterClass(&wndclass);  // register the clock window class
}
```

Listing 3: Here's how the typical *WinLight/Windows* program does some essential things: defines the width and height of the window, creates it, and displays it with some pertinent data to display in it.

```
// window will be wide enough to hold 10 characters
width = GetSysFontCharWidth() * 10;
// window will be twice as tall as the caption (title) bar
height = GetSystemMetrics(SM_CYCAPTION) * 2;

// create the clock window
hWnd = CreateWindow(
  szAppName,                    // registered class name
  szAppName,                    // window caption (title)
  WS_OVERLAPPED|WS_SYSMENU, //  window style
  0,                            // initial x position
  4,                            // initial y position
  width                         // initial width
  height,                       // initial height
  NULL,                         // no parent window
  NULL,                         // no menu
  hInstance,                    // our hInstance
  NULL);                        // no create parameters

  ShowWindow(hWnd, SW_SHOW); // flag the window as visible
  UpdateWindow(hWnd);           // force window to be painted NOW
  DisplayCurrentTime(hWnd);     // immediately display current time
```

Also, GetMessage permits WinLight to suspend the application, giving other applications an opportunity to run. Within DispatchMessage, Winlight passes the message back to our program by calling its window procedure.

At this point, it's worthwhile to say that a full discussion of the windows procedure is beyond the scope of this article. But, it's a rule that every WinLight/Windows program has at least one window procedure.

That is, it must have a procedure that can be called to receive and process the messages passed back by WinLight or Windows. Such messages can indicate a wide range of events to the program, including the user pressing a key, touching a sensitive screen, or moving a mouse.

Listing 5 shows the window procedure of our clock. This procedure simply tests for the receipt of any of three WinLight messages: WM_CREATE, WM_TIMER, and WM_DESTROY. It takes the appropriate action when any of them is received.

When WinLight initially sends the WM_CREATE message during the call to CreateWindow, the window procedure makes a call to SetTimer. This call causes WinLight to send a WM_TIMER message every second.

Although we set the timer to 1000 ms, WinLight is a cooperative multitasking system, and if busy, may take a few milliseconds to get around to sending a WM_TIMER message. If a program needs finer resolution than this, some method other than a timer must be used.

The WM_TIMER message calls DisplayCurrentTime, included in Listing 6, to update the time shown in the window displayed onscreen. The displayed time updates on a regular basis, so the window provides an effective clock.

As a side note, most window procedures receive and process a WM_PAINT message from WinLight notifying the application of a need to redraw (i.e., repaint) the window. However, since our clock program receives WM_TIMER messages on a regular basis, it ignores any WM_PAINT messages.

Listing 6 shows that DisplayCurrentTime gets the time from DOS, stores it at currentTime, converts it to a displayable string, gets a device context (DC) for the clock window, and calls TextOut (provided by Wintight) to display the string in the clock window.

DC is similar to a file handle. It's a token that tells the OS where to send output. A

Listing 4: Every *WinLight/Windows* program must employ a message loop so it can receive messages. Messages are the only way the program *determines what the user is doing or what other events are taking place that should affect its operation.*

```
// GetMessage retrieves next msg and returns zero when a quit msg
// is received
while (GetMessage(&msg, NULL, 0, 0))
  // pass msg into WinLight for processing, ClockWndProc is called)
  DispatchMessage(&msg);
```

listing 5: The window procedure, often called the "winprock," is where the program decides what to do in response to those messages trapped by the message loop. For every window that a program creates and displays, there must be a window procedure.

```
long FAR PASCAL _export ClockWndProc(HWND hWnd, UINT message, UINT
                                      wParam, LONG 1Param)
{
  static UINT timer; // remember timer id so we can kill it on exi

  switch (message) {
    case WM_CREATE:
      // set timer to generate a WM_TIMER msg every 1000 ms
      timer = SetTimer(hWnd, 1000, 1000, NULL);
      break;
    case WM_TIMER:   // l-s timer goes off. update display
      break;
    case WM_DESTROY:// when clock window destroyed, release timer
      KillTimer(hWnd, timer);
      PostQuitMessage(0);// tell msg loop to terminate clock app
      return 0;     // don't pass msg to DefWindowProc

  // pass message to default window procedure for handling
  return DefWindowProc(hWnd, message, wParam, lParam);
}
```

program can have multiple DCs active at one time, each directed to a different window.

In a WinLight program, the key to outputting data to a window is to first obtain a DC to that window using its handle, hWnd. After the TextOut function returns, the DC must be released.

Looking back over this sample program, there seems to be a lot more code than you'd expect for a program that simply displays the time. But as you can see, much of it-perhaps 80%—is the overhead required to create a simple window.

Keep in mind that WinLight returns significant benefits for the overhead included in any program. In addition to protected-modeoperationand multitasking, WinLight provides built-in support for such things as:

- separate windows
- multiple fonts
- text output
- graphical output
- command buttons, scroll bars, and so on
- command menus
- mouse and other pointing device

As a result of all this user-interface support, you can have a clean and familiar-looking application in WinLight or Windows without creating all this functionality yourself.

WINLIGHT APPLICATIONS

In developing an embedded WinLight application, some basic steps minimize development time. First, write your code as if you were writing a Windows application. Simply use the editor integrated into the programming environment installed on your desktop system.

Perform the compile-test-and-debug cycle until your application runs as designed. You can do all this on your desktop development system, running the application and debugger under Windows.

Next, run theapplication under WinLight on your desktop system using the Borland Turbo Debugger. Ensure that your program runs correctly under WinLight. At this point, you can place the application in your target system along with WinLight and do any final testing and debugging.

Currently, the preferred development environment is Borland C since WinLight fully supports the Borland Turbo Debugger for Windows. TheMicrosoftC/C++ graphical debugger can be used when debugging under Windows.

The preceding clock example can also be compiled under Microsoft C/C++. In the future, WinLight will support other languages such as Visual Basic, FoxPro for Windows, and Borland Delphi.

Whether you choose Borland or Microsoft C/C++ tools, use the built-in graphical debugger under Windows. When debugging under WinLight on the desktop system, it's best to use the character-based Turbo Debugger for Windows from Borland, which is virtually the same as the DOS version.

For optimum use of Turbo Debugger, we strongly recommend the use of a monochrome video card installed in your desktop system with a monochrome monitor connected to it.

The Turbo Debugger is a Windows application that outputs its video to the monochrome monitor, leaving the VGA monitor of your development system dedicated to displaying your running WinLight application.

After your application is debugged and running properly on the desktop system under WinLight, place it in the target embedded system. If the target hardware contains sufficient resources, run the Turbo Debugger along with WinLight and your application.

If the target system is small, but includes a serial port, you can use a serial connection to the desktop system while running Turbo Debugger in remote mode. With this arrangement, Turbo Debugger runs on the desktop under Windows while WinLight, the application, and a small piece of debugger kernel run on the target. The serial port and cable link the two systems.

If the target system doesn't include a serial port, serial debugging is not an option and debugging is reduced to the `printf` method. With `printf`, the application writes messages to a disk file residing on a RAM or flash disk. This amounts to virtually no debugging capability at all.

For this reason, inclusion of a serial port on the target hardware is strongly recommended.

This feature, along with sufficient memory for Winlight, the application, ond the debug kernel, provides full debugging capability.

Listing 6: The *typical* program implements several, *if not many,* private *functions.* These functions, like this one, *are* unique to *the operation of the particular program.*

```
void DisplayCurrentTime(HWND hWnd)
{
  // currentTime is static so it's in DGROUP, not onstack
  static DOSTIME currentTime;
  char timeString[20];
  HDC hDC:

  _dos_gettime(&currentTime);  // ask for the current time

  // convert time into a string
  wsprintf(timeString, "%02u:%02u:%02u", currentTime.hour,
            currentTime.minute, currentTime.second);
  hDC = GetDC(hWnd);             // obtain a DC for this window
  // output the time string at position (3,2)
  TextOut(hDC, 3. 2, timeString, lstrlen(timeString));
  ReleaseDC(hWnd, hDC);          // release the DC
}
```

WINLIGHT DRIVERS

The WinLight keyboard driver is Windows compatible. Source is provided with the WinLight Software Developer's kit. The WinLight mouse driver is also compatible, and source code is provided. Armed with the source code, it's fairly simple to derive drivers for other pointing devices such as a touch screen or other nondesktop devices.

WinLight also includes source code for a display driver. This driver is not compatible with Windows. But, Windows display driversarealsoincompatiblewith WinLight. Creoting o WinLight **display driver** for non-standard video hardware is not difficult.

MS WINDOWS, ROM VERSION

Developing your application for WinLight has the added benefit of being fully compatible with the ROM version of Microsoft Windows in case you later decide to run under this OS.

The ROM version of Windows, like WinLight, provides a graphical, protected-mode, multitasking environment. It is well suited to diskless systems since it runs from ROM, RAM, or flosh memory.

Unlike WinLight, the Windows ROM version supports the entire Windows API, including (but not limited to) such features as DOS boxes, DDE, OLE, clipboard, multimedia, multiple document interface, True Type fonts, metafiles, and the help viewer.

If your application requires the support of ony of these features, then Windows ROM version may be your platform of choice. However, consider its hardware requirements when making this decision.

ROM/RAM requirements for Windows ROM version may be as high os 4 MB, depending on the operating mode. This high demand for memory **contrasts** with the 512-KB requirement of WinLight.

APPLICATION CALLING YOU?

We've shown you the basic structure of a WinLight (and thus a Windows) application and provided you with some idea of the development process that best suits this type of program. Our example clock pro gram is exactly that-an example.

Applicationsforembedded systemsvary widely in their purpose, complexity, and system requirements. Still, each can be approached using the guidelines we've presented.

Programming for Windows or WinLight is not a formidable task-just one differing from what you may be used to ond requireing a different set of development tools and techniques. EPC

Scott Baisch has over J 0 years of sales and marketing experience in the software industry. He joined *Datalight* in J 995 after ***working with*** Walker, Richer, and Quinn and Microsoft. He may be reached at *scottb @datalight.com.*

Kevin Smith has *18* years of *software* develop ***ment experience. He joined*** *Datalight* ***in*** J 993 ***and was most*** recently VP of Engineering. Kevin may be reached at *kevins@tgi.net.*

BIBLIOGRAPHY

J. Conger, *The* Waite Group's Windows API *Bible*, Waite Group Press, 1992.

P. Norton and P. Yao, **Windows 3.1** Power *Programming* Techniques, Bantam **Computer Books, 1992.**

C. Petzold, Programming Windows 3. *1*, Microsoft Press, 1992.

SOURCES

WinLight
Datalight, Inc.
188 10 59th Ave., NE
Arlington, WA 98223
(360) 435-8086
Fax: (360) 4350253

Microsoft Windows, ROM Version
Microsoft Corp.
1 Microsoft Way
Redmond, WA 98052
(800) 426-9400

IRS

425 **Very Useful**
426 Moderately Useful
427 Not Useful

John Kates

Precision Timing with PC/104

Real-time *applications need accurate* timekeeping. *John shows* how *affordable* precision time is available to *portable* and *mobile systems* bused on PC/ 7 04. No *more* external *synchonized* clocks with bulky *parallel* interfaces.

Red alert! A massive explosion with the characteristic signature of an underground nuclear weapons blast is detected. It could be a violation of a weapons-control treaty, depending on the explosion site.

Suppose countries F and G-bitter enemies-are surrounded by countries B (an existing nuclear power), A, and C, as shown in Figure 1. Seismic data can determine whether the nuclear blast occurred in B (i.e., no treaty violation) or in F or G.

To make matters worse, some of F or G's neighbors aren't reliable monitoring locations because of relationships with F or G. So, three monitoring stations are set up: site E in country E and sites D1 and D2 in country D.

Time-of-arrival triangulation can determine the location of the blast. But for it to work, the clocks used for triangulation at sites E, D1, and D2 must agree to better than a millisecond.

Many real-time applications just like this scenario require accurate time. To backtrack to fault locations, a power-distribution utility measures the time when voltage or current transients occur at different sites. Lightning strike locations are triangulated by measuring the time when E- or H-field transients occur at distributed detectors.

To achieve accurate time and maintain an accurate local clock (i.e., a synchronized clock), you need:

- a clock, usually implemented as a digital counter
- a frequency source (oscillator) to drive the clock
- a reference to periodically compare the local clock to (either for real-time clock adjustments or for postprocessing correction)
- a disciplining mechanism to keep the clock time synchronized to the reference
- a time-stamp mechanism to get clock readings close to a point in time to be measured
- a controller for smoothly coordinating these features
- an interface from the clock and controller to the host computer system

In this article, I'll tell you about the necessary system components for precision timing. I'll discuss how specifications affect cost and provide examples of PC/ 104 precision-timing implementations using commercially available components.

CLOCK

For precision timing, the clock needs resolution in the range of hundreds of nanoseconds to milliseconds. Data may be in calendar/watch or binary format.

Calendar/watch format (i.e., years, month/day or day of year, hours, minutes, seconds, milliseconds, microseconds) is commonly transferred in binarycoded decimal (BCD). Binary format usually measures time in second of year and microsecond of second.

The calendar/watch style makes more sense to humans, but binary is easier for significant real-time computations.

FREQUENCY SOURCES

Frequencysourcesareavailable in wide ranges of calibration accuracy, stability, power consumption, and size.

Calibration accuracy is the maximum frequency error at a specified temperature (usually 20°C). The error caused by calibration accuracycan becorrected byalgorithms that measure the error of the free-running (i.e., undisciplined) oscillator compared against a time reference. The algorithms then apply an equal but opposite discipline (i.e., correction).

Stability refers to how much frequency changes over a temperature or power-supply voltage range. Because the operating environment of an oscillator is often harder to control than its power-supply voltage, temperature effects typically determine stability.

Stability is usually specified as ppm (parts per million). For example, a temperature-compensated crystal oscillator maychangea maximum of ± 1 ppm over a 0-50°C temperature range.

A nontemperature-compensated AT-cut oscillator might be specified as ±35 ppm over the temperature range. The stability over a narrower temperature range (e.g., 20–35°C) for inexpensiveoscillators may be ten times better than the wider range.

The open-loop stability and accuracy of an inexpensiveAT-cutcrystal oscillator may beadequate in high-precision applications as long as the referenceoccursoften enough and there's a good disciplining mechanism. The time between references is called the "open loop" or flywheeling time.

A rule of thumb for specifying frequency-source stability based on open-loop requirements is:

$$\text{req. stability} = \frac{\text{max. allowableerror}}{\text{max. time between refs}}$$

For example, if the maximum allowable error is 1 µs, and the time between references is 1 h (because the clock is in a submarine, for example), the stability requirement is:

$$\frac{1\ \text{ms}}{3600\text{s}} = 2.6 \times 10^{-10}$$
$$= 2.6 \times 10^{-4}\ \text{ppm}$$

To avoid overspecifying the oscillator, be realistic about the temperature changes encountered while running open loop. The narrower the open-loop temperature range, the less expensive oscillator you need. Or, consider an alternative reference to allow more frequent comparisons.

In the submarine example, an ovenized crystal unit might meet the 2.6 x 10^{-10} requirement over a 20–30°C range. Overspecifying the temperature range or open-loop time might demand a $5000 rubidium oscillator with a 50-in^3 volume and a 30-W power requirement where a $400 ovenized crystal oscillator might otherwise suffice.

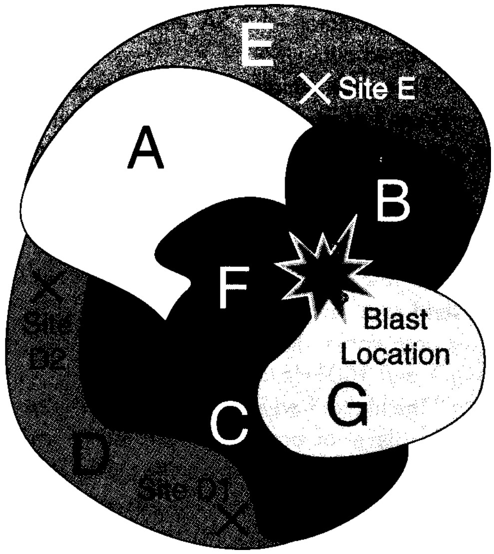


Figure *1:* Which ***country*** exploded the nuclear device ***—country B,*** F, or G? Precision timing can answer the question.

TIME REFERENCES

Referencescan be based on a local time over some area, or they may need to be based on UTC (Universal Coordinated Time). UTC is based on atomic clocks, with leap seconds added every 18 months or so asdictated by **astronomical** measurements.

The selection of a reference depends on the absolute accuracy required, the physical location of the clock being disciplined, the maximum open-loop time, security, and cost. The maximum open-loop time is sometimes based on a tradeoff of clock stability versus cost.

The references for standards and off-the-shelf equipment are GPS, time codes, and NTP (network time protocol). UTC time is distributed worldwide by the GPS system, which has 28 satellites-plus backups-in nonsynchronous orbit.

A GPS receiver at an unknown position must receive signals from four satellites to establish its position and UTC time. Once the position is known, most GPS receivers can be placed in a known-position mode to provide accurate time when only one satellite is received.

The GPS receiver corrects for the propagation time from the satellite to the antenna. The user tells the receiver the additional error due **to** antenna cable length so the receiver can compensate. Figure 2 shows a block **diagram** of a precision clock using GPS as a reference.

GPS ON PC/l 04 BUS

Modular GPS receivers in the $300–500 price range like the Trimble SV6-CM3, the Motorola VP Oncore, and the Rockwell Microtracker LP are available packaged as Satpak PC/l 04 modules by Zeli Systems. The power requirements are about 175 mA at +5 V.

The Satpaks include keep-alive power for the GPS receivers so they can lock onto satellites within 30 s of +5-V power application. The modular GPS receivers output one reference pulse per second (pps).

The PC1 04-SG interfaces directly to a GPS receiver on a Satpak through a TTL-level serial I/O port. It shares data with the user system through dual-port memory on the PC 104-SG. Alternatively, the user communicates directly with the receiver and initializes the major (i.e., days through seconds) time on the PC 104-SG when the receiver is tracking.

The second approach requires that the selected GPS-receiver 1 -pps pulse be "on time" (i.e., on exact UTC second boundaries). Of the modular receivers mentioned, only the Trimble SV6-CM3 meets the on-time restriction.

Some precision-timing applications are on mobile platforms like aircraft or land vehicles. Mobile platforms may have the GPS signal temporarily interrupted when a plane banks **steeply or** a vehicle is between tall buildings.

Depending on the application and environment, it's necessary to assume the open-loop time is longer than 1 s in case the antenna is obscured. Historically, other radio time references like WWV shortwave, WWVB longwave, LORAN, and the GOES satellite system acted as references.

However, GPS is cheaper, available worldwide, more accurate, and less susceptible to interference. And, it has simple, cheap, and easy-to-install antennae.

A fringe benefit of GPS is navigation data (i.e., position and velocity) which may be useful for portable or mobile systems. The accuracy of the 1-pps reference-pulse output is usually better than 1 µs.

U.S. Defense deliberately introduces random errors (called Selective Availability or SA) in the GPS signals that add time errors of up to 200 ns. (Recently, government decided to discontinue SA in 1998.)

While SA is still in effect, UTC accuracies below 100 ns can be obtained. By averaging 1-pps references over 18 h or longer-which, in turn, requires a very stable oscillator-the SA effects can be averaged out.

Time-code signals are analog sine-wave signals that are amplitude modulated with pulses carrying time information. The Inter-Range Instrumentation Group (IRIG) defines many time codes.

The most common (IRIG-B) is based on a 1-kHz sine wave, which makes it easy to send over phone lines or record on VCR sound tracks. IRIG-A and IRIG-G have 10- and 100-kHz carriers.

The higher frequency carriers enable a given level of accuracy to be met more easily. Modern synchronizable clocks like the JXI2 PC104-SG get about 300-ns accuracy from IRIG-B and slightly better from IRIG-A or IRIG-G.

NASA has a time code similar in performance to IRIG-B called NASA36. The security community uses the XR3 (250-Hz carrier) or 2137 (1-kHz carrier) codes.

Time-code frames repeat the encoded time once per second (IRIG-B, NASA36, XR3, 2137), 10 times per second (IRIG-A), or 100 times per second (IRIG-G). By using the carrier wave as the reference, 1000–100,000 reference times per second are available. Figure 3 shows a block diagram of a precision clock using time-code input as a reference.

High time-code accuracy requires delay correction from the time-code source to the synchronized clock. The delays include propagation delay that equals the distance from the source divided by the transmission speed in the wire (or air for radio).

Phase shifts at the time-code carrier frequency are introduced if the time-code source must drive a reactive load like an unterminated cable. A tiny phase shift of 1° at 1 kHz corresponds to over 3 µs of added delay.

JXI2's PC104-SG allows the user to specify a propagation-delay value in the range of ±0–99999.9 µs. The correct value is determined approximately by the calculations above. It is obtained by comparing a pulse output from the synchronized clock with a reference-time pulse from a portable atomic clock or a GPS receiver.

Time-code sources come from other synchronized clocks which may be referenced to GPS or another time-code source. JXI2's board-level synchronized clocks can generate time codes from their clocks.

Time codes are easy to distribute long distance over wires, compared with GPS signals which must be distributed over coaxial cable with maximum cable runs of 100 m.

National defense and aerospace facilities with heavy demand for precision timing often broadcast time codes on VHF or UHF radio frequencies for easy reception by mobile-instrumentation vans.

NTP statistically averages many jittery readings of time references over a network.

A local clock is the real-time clock of the workstation or PC. Each clock is controlled by a program that communicates with other clocks using a network protocol. One clock on the network may be a "time

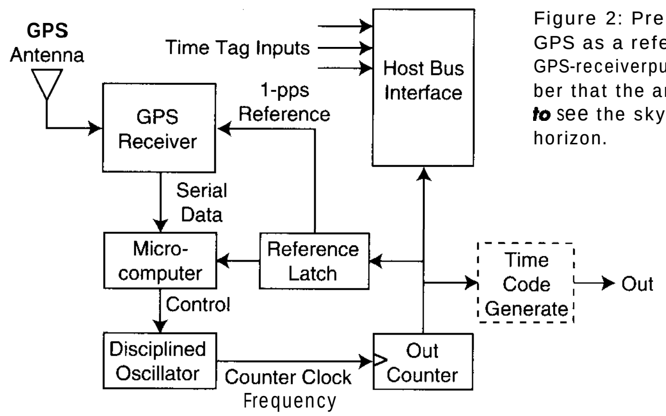


Figure 2: Precision timing using GPS as a reference uses a *1-pps* GPS-receiver pulse output. *Remember* that the antenna must be able *to* see the sky to within 10" of *the* horizon.

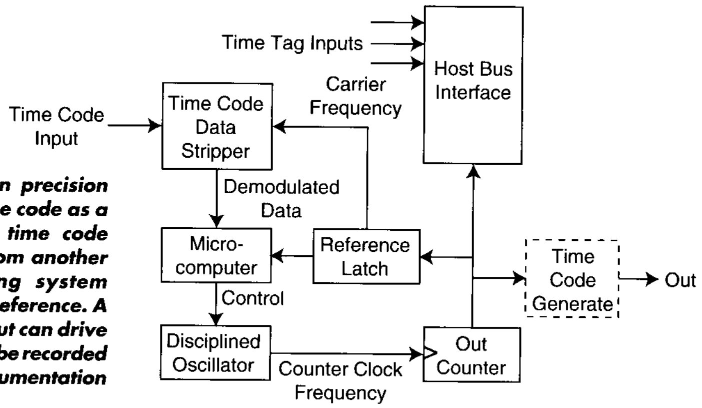


***Figure 3:** When precision timing uses time code as a reference, the time code often comes from another precision-timing system based on GPS reference. A time-code output can drive a slave clock or be recorded on VCRs or instrumentation tape recorders.*

server," synchronized to an off-network reference like GPS. Alternately, you can access a remote time server over the Internet.

The accuracy of NTP time is limited by the real-time clock resolution—good on workstations, terrible on PCs—network latencies, and, of course, the accuracy of the time server. Lightly loaded local networks can expect accuracies in the low-millisecond range.

DISCIPLINING MECHANISMS

The simplest disciplining mechanism, called "jamming," sets the clock to the reference time each time a reference point is available—like setting your PC clock's time from your wristwatch. Jamming is usually unacceptable because it causes positive or negative jumps in time.

Another mechanism uses count swallowing or doubling. Periodically (e.g., 1000 times per second in the case of the PC104-SG), a count is added or skipped in the counter. The total number of adjustments per second multiplied by the adjustment size applies the desired correction based on previous reference measurements.

Time jumps in the positive direction still occur, but they're very small—just 100 ns. Time jumps in the negative direction do not occur because a swallowed count causes time not to advance for 100 ns.

By adding or skipping a 100-ns count 1000 times per second, it's possible to compensate for 100 µs of oscillator error every second. This compensation is 100 ppm—enough to compensate for even a very inexpensive crystal oscillator.

Even the small steps in time caused by the count-swallowing mechanism may be too high for some applications. A voltage-controlled oscillator allows the frequency to be tightly controlled.

For example, if the oscillator has a ±3-ppm control range for 0–5 V, and a D/A converter has a 1.25-mV output resolution, the frequency may be controlled with an accuracy of:

$$\frac{6\text{ ppm}\times 1.25\text{ mV}}{5\text{ V}} = 1.5\times 10^{-9}$$
$$= 1.5\times 10^{-3}\text{ ppm}$$

Frequency-control resolution is limited by the oscillator stability over the time interval, the D/A resolution, the D/A error, and the noise floor of the control voltage as seen by the oscillator.

Calculating the correct control voltage may require averaging reference measurements over long intervals.

TIME STAMPS

Once a clock with a reference and disciplining mechanism is implemented, it's necessary to obtain clock readings to go with user data points. The first approach usually considered runs a program on the host computer that reads the current time from the synchronized clock.

A real-world event typically triggers the host's time request. The accuracy of software-triggered time stamps is limited by unpredictable delays due to interrupt latencies and OS scheduling priorities.

External time tags provide a more accurate method. A logic pulse (i.e., the external input) causes the clock time to be recorded whenever a specific edge (positive or negative) occurs.

The record is stored in hardware registers accessible to a host computer, or it may be buffered internally by a microcomputer. Either way, when the record is ready for the host computer to read, a status bit is set and an interrupt occurs if enabled by the host computer.

Newer synchronized clock interfaces can support multiple external events. For example, a PC1 04-SG option allows three independent external time-tag inputs. The *TAG8 option on the JXI2 VME-SYNCCLOCK allows eight independent events.

The minimum time spacing between events depends on how fast the host computer reads the time tags, freeing the tag registers to accept a new reading. JXI2 recently introduced a FIFO mechanism for its VME product that allows time tagging of any or all of the 16 event inputs every microsecond.

The recorded event data requires 64 bits of FIFO storage for each microsecond with any combination of events. The FIFO stores the tags until the host empties it.

Other applications generate periodic rates synchronized to the reference and generating pulsesatuser-programmedclock times. Rates are generated by feeding the disciplined counting frequency to dividers.

Time-match pulses are generated by the output of timecomparator logic. One side of the time-comparator logic connects to the clock. The inputs to the other side are controlled by the user.

CONTROLLERS

The internal controller for synchronized clocks is a microprocessor. In most modern designs, many time-critical or timing-specific functions are performed inside an accompanying FPGA which typically also contains the counter.

For users choosing the host processor to control disciplining and processing reference measurements, JXI2 supplies stripped-down PC1 04SG units providing a binary counter and two sets of registers. One set captures the clock time of reference pulses, and the other captures event times.

HOST INTERFACES

The PC 104-SG provides the clock, disciplining, event tagging, and bus interface on a single PC/104 module. It accepts time-code inputs and a 1-pps time pulse from a GPS receiver simultaneously, providing redundancy in case one input fails. It allows multiple external-event inputs and provides heartbeat rates and user-specified match time pulses.

It accommodates onboard-DIP package-sized oscillators which are available with stabilities down to ±1ppm. For tighter oscillator stability, a 1 O-MHz ovenized crystal or rubidium oscillator can be mounted in the PC/I 04 enclosure and the output fed to the PC 104-SG.

PORTABLE PRECISION

Thanks to recent advances in FPGAs and small modular GPS receivers, it's possible to implement precision timing using PC/I 04 modules that previously required external synchronized clocks with bulky parallel interfaces.

The PC/I 04 modules accept a variety of references and frequency sources. They provide the host with a variety of outputs based on the synchronized clock. Affordable precision time is here for PC/104 portable and mobile systems. PCQ.EPC

John Kates has designed board-level precision-timing products since 1982. Currently, he is the president of JXI2 located in El Paso, TX. You may reach John at jkates@whc.net.

SOURCES

PC 104-SG
JXI2, Inc.
2308 Lake Omega St.
El Paso, TX 79936
(9 15) 856-0241
Fax: (915) 856-0245

VP **Oncore**
Motorola, Inc.
4000 Commercial Ave.
Northbrook, It 60062
(847) 714.7329
Fax: (847) 714.7325

Microtracker **LP**
Rockwell Semiconductor Systems
431 1 Jamboree Rd.
Newport Beach, CA 92660
(714) 221.6996
Fax: (714) 221.6375

SV6-CM3
Trimble Navigation, Inc.
645 N. Mary Ave.
Sunnyvale, CA 94088.3642
(408) 48 I-6940
Fax: (408) 48 I-6020

Satpak PC/ 104
Zeli Systems
3233 Pagosa Ct.
El Paso, TX 79924
(915) 751.3222
Fax: (915) 751.3222

IRS

428 Very Useful
429 Moderately Useful
430 Not Useful

Applied PCs

Fred Eady

Multiaxis Stepper Motors Continued

Fred walks us from his high-school slide-rule days into the modern world where milling machines use embedded intelligence coupled with motors and encoders. Take a turn at the wheel of this hands-on application.

Over the past few weeks, I've put in quite a bit of shop-floor time researching for this article. I had a great time, but I'm still digging metal out of the soles of my shoes.

During my stints in the various machine shops, I was privy to some incredible equipment and wild stories.

Get this. Sometime way back in Machine Tool B.C. (Before Computers), engineers experimented with ways to automate a manually controlled, three-axis milling machine. They wanted a circular cut-a bit more complex than a straight one.

Their embedded PC was a slide rule. All the coordinate calculations for the circular cut were resolved-slide rules don't compute-and listed in order by hand.

Of course, the intent was to prove in theory that, with the addition of a precision motor and encoding device attached to each axis, any manual cut could be automated and repeated with great accuracy. (took Ma, no hands!)

To test their theory, it took four engineers. One engineer read the x and y coordinates while two others turned handcranks positioning the x and y surfaces.

Once the correct coordinates were dialed in, the fourth engineer manually engaged the z axis and thus the cutting tool. You can figure the rest. A few hundred verbal xs and ys later, a circle was born.

On hearing this tale, I realized I haven't heard or used the term "slide rule" in quite some time. In my teens, I didn't go anywhere without one!

I owned a top-of-the-line 10-incher, a nifty little 6-incher, and a roundy-round

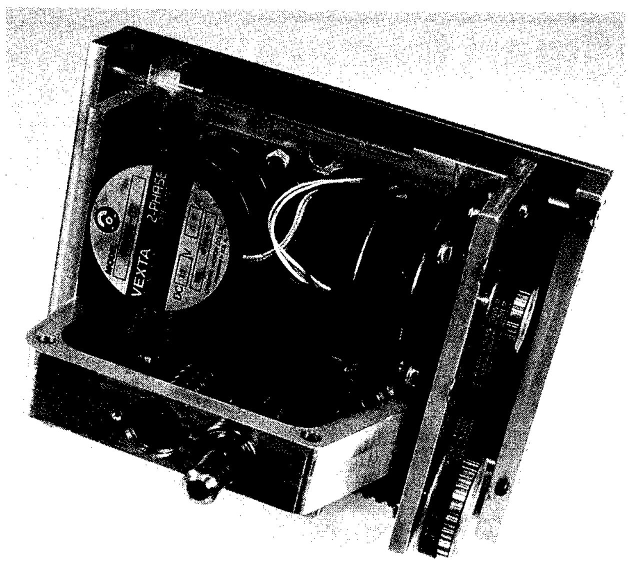
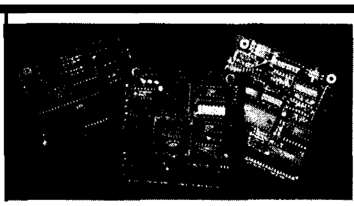


Photo 1: This industrial positioner was fabricated in a local machine shop and is capable of moving *the* spindle in 0.0001" increments.



one. I kept the 6-incher and/or the roundy-rounder in my pocket at all times. You never knew when the urge to calculate would hit. Was I destined for embedded PCs or what?

Today, in **Machine** Tool A.D. (All Digital), our slide rules are '386- or '486-based embedded engines. And those four engineers.. .they've been replaced by motioncontrol software.

However, their simple idea lives. All modern milling machines use embedded intelligence, motors, and encoders to control motion along multiple axes.

MOTION-CONTROL PROCREATION

In my shop-to-shop travels, I saw lots of evidence that motion-control applications were finding homes in embedded PCS.

For the purposes of this article, I could have borrowed a 4-ton milling machine capable of milling a l-ton piece of steel. But, I could see myself washing dishes in the Circuit Cellar cafeteria for life, too!

What to do? It was fourth down, 99 yards to go, and I needed a real working motion table. Bing!

Why not have one of those gray behemoths spit out an itty-bitty copy of itself for this story! As someone ancient and Biblical once said, "So it is written, so shall it be done."

CAD programscadded, CAM programs cammed, indexers indexed, and more aluminum chips found their way into my soles. The resulting mechanism is portrayed in Photo 1 -a precision industrial positioner capable of independently **or** simultaneously indexing x and y axes at an accuracy of 0.0001 "per step. The proud parents of this little monster are a Bridgeport three-axis mill and his beautiful '386 embedded motherboard.

I think that makes me an uncle-which puts the little x/y positioner as a nephew. Since family should have only the best, I decided to give my nephew the best motion-control system that money could buy.

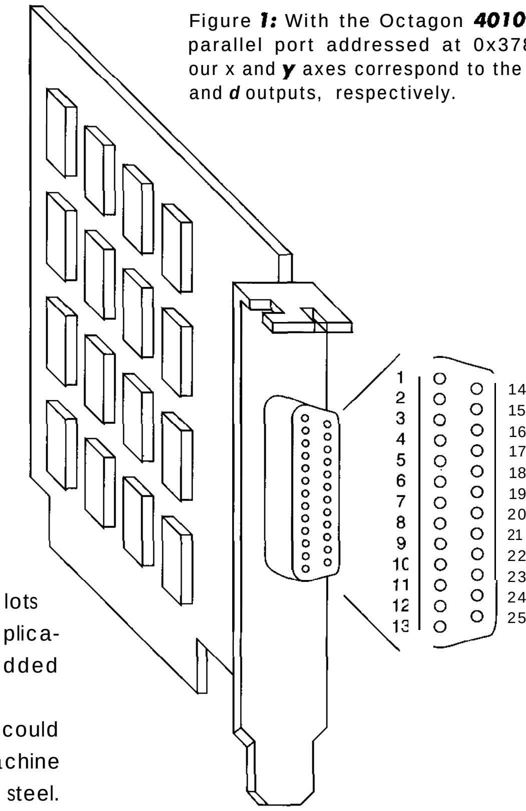


Figure **1:** With the Octagon ***4010's*** parallel port addressed at 0x378, our x and ***y*** axes correspond to the c and ***d*** outputs, respectively.

IF I ONLY HAD A BRAIN

Last time (INK 71), I introduced the Octagon 4010 embedded-PC engine, Indexer LPT software, and the Dragon Driver stepper driver. Using that hardware plus my new nephew, I'll now take a closer look at the theory behind the motioncontrol software running in the Octagon 401 0—and play with the new baby, too.

The Octagon 4010 executes the Indexer LPT program. Indexer LPT's sole purpose is to interpret a set of motioncontrol commands and pass the resultant movements to the Dragon Driver stepper driver.

These movements are either linear or angular in nature and are more commonly known in the motion-control field as linear and circular interpolation. Indexer LPT uses a best-fit strategy to accomplish linear and circular interpolation.

Stepping motors attached to the Dragon provide the motion that does the work. The internal electronics and firmware also support linear and circular interpolation.

Listing **1:** Assuming a ***preset home position, with these three commands Indexer LPT uses a best-fit algorithm to simultaneously move the x and y stepper motors.*** The result is a smooth ***positive linear motion at*** 45" relative to the x-axis.

```
home:c              ;home the x-axis
home:d              ;home the y-axis
move:c,5000,d,5000;move the x- and y- axes 5000 steps
```

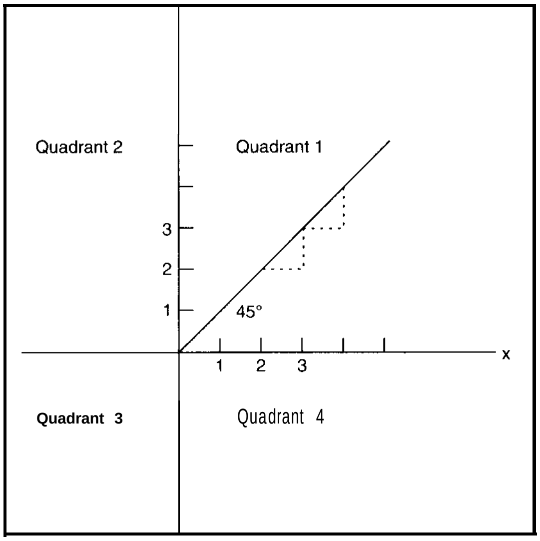


Figure 2: This is an exaggerated depiction of jogging the x- andy-axes. Note ***the*** x andy ***jogs*** for a 45" angle are identical in measure.

LINEAR INTERPOLATION

Before we talk about interpolation, let's set up the playing field. Our little industrial positioner can independently or simultaneously move its spindle along xy axes. This limits us to a 2D coordinate system.

We all know this 2D system as the rectangular-or Cartesian-coordinate system. The rectangular positioning system is most common in motion control because it's easy to understand.

The alternate positioning system is polar. (Our positioner doesn't have a third axis, so I won't discuss the polar coordinate system.)

Basically, using the Cartesian system, the motion-control programmer selects an x- and y-zero-reference point and instructs the program to position the spindle about the reference point. With the ability to command angular or linear motion, just about any 2D shape can be drawn or cut.

Suppose we wanted to move our industrial-positioner spindle from home to a positive x position 0.5" away. Assuming the spindle is at rest at location (x0,y0), the command is move : c, 5000.

The "c" corresponds to our positioner's x-axis and the "d" represents the positioner's y-axis since the Octagon 401 O's parallel port is addressed at 0x378. This axis assignment is hardcoded within Indexer LPT. Take a look at Figure 1 for clarification. More information is given in Table 1 in my last article (INK 71).

Only the x-axis motor moved since no y-axis motion was commanded. To perform the same operation on the y-axis, simply substitute "d" for "c". No interpolation is needed for this move.

Pretty **straightforward,** huh? Let's look at this from another angle.

Suppose we wanted to direct the spindle to return home and move within quadrant 1 at a 45" angle in a positivexand positive y direction for the same number of steps. Assuming a set-home command placing x and y home positions at (x0,y0) was issued, the relevant Indexer LPT commands are shown in Listing 1.

Let's go back to our four engineers turning handcranks. If the one at the helm of the x-axis changes his position, creating a jog in the x direction, and then the y-axis engineer jogs in the y direction, the end point is reached, but the path isn't linear.

Obviously, very small jogs produce the most accurate linear motion, and to make

Listing 2: This C code snippet computes the x and y coordinates for a range of ***1°*** to 45" on the circle segment in Quadrant ***1.***

```
#include <stdio.h>
#include <dos.h>
#include <math.h>
#include <conio.h>
// Here's one way to find x and y coordinates on an arc or circle
// in a 2D  Cartesian-coordinate  plane.
void main0
{
  int degrees:
  double X,Y,radians,r;
  r = 5;                                  //Radius is set for 5 steps
  clrscr0;
  for (degrees = 1; degrees <= 45; ++degrees){
    radians = degrees * 0.017453293: // Convert A degrees to radians
    X = r * cos(radians);            // Calculate x coordinate
    Y = r * sin(radians);            // Calculate y coordinate
   printf("Angle = %d X = %lf Y = %lf\n",degrees,X,Y);//Print results
  }
```

Angle = 01	X = 4.999238	Y = 0.087262	Angle = 24	X = 4.567727	Y = 2.033683
Angle = 02	X = 4.996954	Y = 0.174497	Angle = 25	X = 4.531539	Y = 2.113091
Angle = 03	X = 4.993148	Y = 0.261680	Angle = 26	X = 4.493970	Y = 2.191856
Angle = 04	X = 4.987820	Y = 0.348782	Angle = 27	X = 4.455033	Y = 2.269953
Angle = 05	X = 4.980973	Y = 0.435779	Angle = 28	X = 4.414738	Y = 2.347358
Angle = 06	X = 4.972609	Y = 0.522642	Angle = 29	X = 4.373099	Y = 2.424048
Angle = 07	X = 4.962731	Y = 0.609347	Angle = 30	X = 4.330127	Y = 2.500000
Angle = 08	X = 4.951340	Y = 0.695866	Angle = 31	X = 4.285836	Y = 2.575190
Angle = 09	X = 4.938442	Y = 0.782172	Angle = 32	X = 4.240240	Y = 2.649596
Angle = 10	X = 4.924039	Y = 0.868241	Angle = 33	X = 4.193353	Y = 2.723195
Angle = 11	X = 4.908136	Y = 0.954045	Angle = 34	X = 4.145188	Y = 2.795965
Angle = 12	X = 4.890738	Y = 1.039558	Angle = 35	X = 4.095760	Y = 2.867882
Angle = 13	X = 4.871850	Y = 1.124755	Angle = 36	X = 4.045085	Y = 2.938926
Angle = 14	X = 4.851479	Y = 1.209610	Angle = 37	X = 3.993177	Y = 3.009075
Angle = 15	X = 4.829629	Y = 1.294095	Angle = 38	X = 3.940054	Y = 3.078307
Angle = 16	X = 4.806308	Y = 1.378187	Angle = 39	X = 3.885730	Y = 3.146602
Angle = 17	X = 4.781524	Y = 1.461859	Angle = 40	X = 3.830222	Y = 3.213938
Angle = 18	X = 4.755283	Y = 1.545085	Angle = 41	X = 3.773548	Y = 3.280295
Angle = 19	X = 4.727593	Y = 1.627841	Angle = 42	X = 3.715724	Y = 3.345653
Angle = 20	X = 4.698463	Y = 1.710101	Angle = 43	X = 3.656768	Y = 3.409992
Angle = 21	X = 4.667902	Y = 1.791840	Angle = 44	X = 3.596699	Y = 3.473292
Angle = 22	X = 4.635919	Y = 1.873033	Angle = 45	X = 3.535534	Y = 3.535534
Angle = 23	X = 4.602524	Y = 1.953656			

Figure 3: This resulting table is generated by the C program in Listing 2.

a smooth line, both axes must move simultaneously toward the end point. This precision jogging and simultaneous motion is referred to as linear interpolation.

The Indexer LPT software precisely calculates a series of very tiny single-axis movements which appear to be a perfectly straight linear motion. The best-fit strategy controls the axes to traverse the best-fit linear path to the destination.

In our example, the x- and y-axes are stepped simultaneously at a 1: 1 ratio for 5000 steps. This 1: 1 step ratio produces a 45° linear motion outward from the point of origin, (x0,y0).

Mathematically, linear interpolation can be thought of as an angular trig function applied against a right-angled triangle.

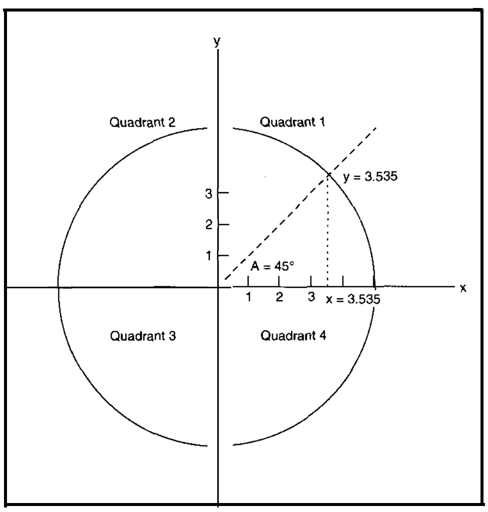


Figure 4: The trigonometric *functions* $x = r \times \cos(45)$ and $y = r \times \sin(45)$ yield these specific points on the *circular section* in Quadrant 1.

The x-axis is the triangle's adjacent side. The y-axis is the triangle's opposite side.

The tangent of our angle (in this case, 45°) equals the opposite side divided by the adjacent side. I retired my slide rule, but with just a couple of keystrokes on one of those scientific silicon rulers, you'll find that the tangent of 45" is 1.

This falls in line with our example, as we are moving x and y at a 1: 1 ratio. Figure 2 is an exaggerated representation of our explanation of linear interpolation.

CIRCULAR INTERPOLATION

Working with circular interpolation involves much of the same math we used to find points in the Cartesian plane using linear interpolation. The basic formula for finding a point on an arc or circle in a rectangular x/y plane is:

$$r^2 = (x - a)^2 + (y - b)^2$$

where *r* is the radius of the circle or arc, xequals the x-axis coordinate, yequals the y-axis coordinate, *a* is the x-axis offset, and *b* is the y-axis offset.

Again, using this equation against a right-angle triangle, x is the adjacent side while y is the opposite side. The formula requires that the radius and at least one side be known to determine the remaining coordinate. The radius, a starting point, and an ending point are normally known.

Another way to solve for x and y coordinates in an arc or circle is:

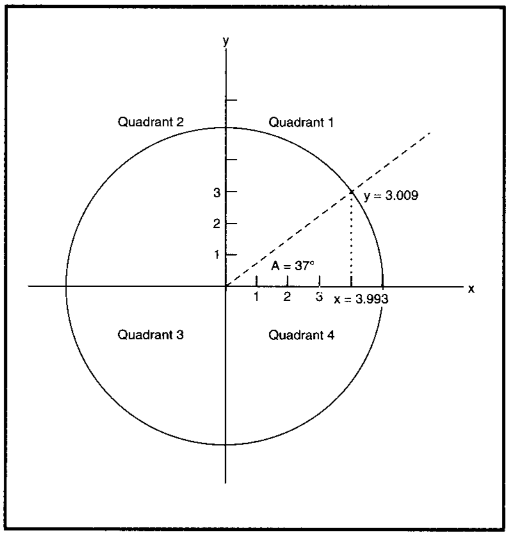


Figure 5: Using the functions shown in Figure 4, this set of points exists when the angle is at 37°.

$$x = r \times \cos A$$
$$y = r \times \sin A$$

where *x* equals the *x*-axis coordinate, *y* equals the *y*-axis coordinate, *r* is the radius of the circle or arc, and *A* is the angle opposing the opposite side.

Listing 2 assumes a radius of 5″ and a positive angular displacement of 1–45° with all movement in quadrant 1 in a positive *x* and *y* direction. Figure 3 represents the run of the program in Listing 2.

Note that at the 45° mark, *x* and *y* are equal in value, as discussed in conjunction with Listing 1. Figure 4 represents the points found at 45°, and Figure 5 depicts points found at 37°. Both Figures 4 and 5 use the radius of 5″ from Listing 2.

MAKING A POINT

The idea here is to show you how interpolation is accomplished if you're developing your own embedded motion-control software. Indexer LPT doesn't necessarily use the algorithms I've introduced here.

One motion-control program I saw used these formulas in a repetitive form and stored the results in an array. Once all of the points had been computed, another part of the program translated the point-to-point movements into digital form.

Essentially, the digital information was an array of step and direction bits stored in sequence to excite the motor-driver circuitry. These bits were clocked out at a desired rate to obtain the end result—controlled linear and circular motion resulting in the motion-control programmer's machined part.

The program was written in BASIC and was compact enough to fit in the Octagon 4010 storage array. In fact, the parallel port was used for output, and all of the computations and I/O were done within BASIC. A keyboard and display were the only necessary peripherals. A perfect application for the Octagon 4010.

Some great embedded software tools ease the task of rolling your own motion-control code. But, why would you do that?

Ready for this? Using Indexer LPT, you never have to make a single interpolation calculation. As a matter of fact, you don't even have to write a single line of code...of any kind. The math is done for you.

You may choose a high-level programming language to assist in some automation routines, but that's optional, too. Let's take a closer look at some of the more commonly used Indexer LPT commands.

INDEXER LPT COMMANDS

We have already experienced the `move` command, so let's talk about it first.

- `Move`—The syntax for `move` is:

```
move:<axis>,<steps>...
  [,<axis>,<steps>]
```

This command simultaneously moves up to six axes for a specified number of steps. `move` generally implements rapid traversal of an axis.

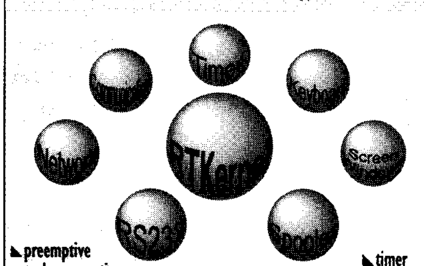
In a real-world example, `move:c,-5000,d,5000` moves the *x*-axis –5000 steps and the *y*-axis +5000 steps. The movement automatically interpolates.

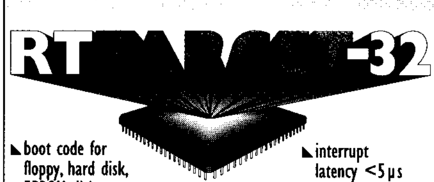
- `Circle`—The syntax for `circle` is:

```
circle:<direction>,<axis>,
  <steps to cp>,<axis>,<steps
  to cp>
```

The `circle` command uses the designated axes to traverse the "best fit" circle around a specified center point. Direction can be clockwise or counterclockwise.

Assume the *c*- and *d*-axes (*x* and *y* Cartesian) were at (*x*0,*y*0) when `circle:ccw,c,0,d,200` was executed. This code





centers the circle at the rectangular coordinate (x0, y200) with a radius of 200. The circle is drawn in a counterclockwise direction.

- Home-Executing the home command moves the selected axis to a home reference position established by set-home.

The result of home :d is that the y-axis moves to a predetermined home position. This command is effective only if a previous set-home command has been issued.

- Jog-The syntax for jog is:

```
jog:<axis>,<steps>
```

This command moves the selected axis a number of steps from the current position.

For example, if the c (x-axis) motor was at point 300 before execution of jog :c,- *100,* the resultant motor position would be 200.

- Set-home-The syntax for set-home is:

```
set_home:<axis>
```

This command establishes the home-reference position for the selected axis. For example, the current motor position of the y-axis is referenced as home following the execution of set_home:d.

In total, Indexer LPT has over 50 commands. And, it also provides the ability to communicate with high-level languages such as C and Pascal.

PROGRAMMABLE MOTION

Indexer LPT operates as a character device named "motor." Since motor is accessed much like a file, any programming language that manipulates files can communicate with Indexer LPT.

Listing 3 shows how a C program can be written to control motion via Indexer LPT's mot o r device. Two file pointers, one for read and one for write, are opened for the motor device. Once a valid motor device is sensed, it's simply a matter of reading and writing character strings.

Reads gather information such as motor position, program settings, and status, while writes are actually the Indexer LPT ASCII command strings and parameters. You see-Indexer LPT, and thus your motion-control system, can be programmed.

listing 3: High-level languages such as C can be *used* to send commands and *retrieve* ***status from Indexer LPT.***

```
/* Indexer LPT uses 1 file ptr for reads and another for writes */
//include <stdio.h>
main0
{
  FILE *fpr,*fpw;
  char instring[80];
  long position:
  /* If "motor" cannot be opened in read mode, it is invalid */
  if( (fpr = fopen("motor", "r")) == NULL){
    printf("cannot open motor device\n");
    exit(1);
  }
  fpw = fopen("motor", "w"); /*Open file ptr to motor for write */
  fputs("set_home:c\n", fpw ); /* Set home ref. pt for x axis */
  fputs("move:c,1000\n", fpw );/*Move x-axis motor +1000 steps */
  fflush(fpw);                /* write output stream to "motor" */
  fgets(instring, 80, fpr);   /* Read position from the mailbox */
  printf("%s", instring);     /* and print */
  position = atol(instring); /* Convert ASCII pos. to numeric */
  printf("%ld\n", position): /* and print */
  fputs("home:c\n", fpw );    /* Send x-axis motor home */
  fflush(fpw):
  fgets(instring, 80, fpr);   /* Read pos. again and print */
  printf("%s", instring);
  fclose(fpw);                /* Close file ptrs and exit */
  fclose(fpr);
  exit(0);
```

STEPPING OUT

Embedded motion control is an exciting and highly productive application of our present embedded-PC technology. Unrelated gears, pulleys, motors, and software can be transformed into an intelligent electromechanical system.

The Octagon engine's ease of use is complemented by the superbly engineered Indexer LPT software. By adding the universal Dragon Driver, we effortlessly turned motors and ballscrews on our small-scale positioner.

Debugging newly developed motion-control hardware and software takes time and money. By selecting and combining proven hardwarecomponents, thedesigner can bypass lost time due to new product development and testing.

Developing motion-control softwarecan be time consuming and expensive, too. Most of the time, the application is written for a particular hardware configuration or task.

With a well-thought-out package like Indexer LPT, if the product specification or task changes, the designer simply specifies another Indexer LPT-compliant component without regard for the software.

If stepping motors are in your future, contact the companies listed as sources. You'll find that, collectively, they offer a wealth of motion-control products and expertise. APC.EPC

Fred Eady ***has over 19years experience as a systems engineer. He has*** worked with computers ***and*** communications ***systems large and small,*** simple and ***complex. His forte is*** embedded-systems ***design and communications. Fred may be reached at edtp@ddi. digital. net.***

REFERENCES

M. lynch, "Understanding CNC Motion Types," Modern Machine Shop, 99-I 04, 1996.

E. Oberg, F.D. Jones, H.L. Horton, and H.H. Ryffel, Machinery's *Handbook,* Industrial Press, NY, 78-81, 1992.

SOURCES

Indexer **LPT**
Ability Systems
1422 Arnold Ave.
Roslyn, PA 19001
(215) 657.4338
Fox: (2 15) 657-78 15

Dragon Driver
General Controls
2350 Brickvale Dr.
Elk Grove Village, IL 60007
(708) 595-2 152
Fax: (708) 595.2271

Octagon 40 10
Octagon Systems
6510 W. 91st Ave.
Westminster, CO 80030
(303) 430-I 500
Fax: (303) 426-8 126

IRS

43 1 Very Useful
432 Moderately Useful
433 Not Useful

CONNECTIME conducted by Ken Davidson

The Circuit Cellar BBS
300/1200/2400/9600/14.4k bps
24 hours/7 days a week
(860) **871-1988—Four** incoming lines
Internet E-mail: **sysop@circellar.com**

I'm a bit cramped for room this issue, so I can only do a one-message thread. The one I chose deals with whether it's necessary to provide some kind of venting for a battery sealed in a water-tight box. There is even a reference back to a thread in last month's column.

Venting

Msg#: *3836*
From: Janusz Suchorolski To: All Users

I wonder if there is an "official" way for providing a vent to a sealed battery enclosure. Say I need to use a lithium battery inside a watertight box. I keep hearing people saying "you need to vent it." However, I could not locate an off-the-shelf gadget or valve that might do the job. Adding such a vent should not jeopardize the watertightness as well. Any suggestions?

Msg#: 4177
From: Pellervo Kaskinen To: Janusz Suchorolski

I have been thinking this venting business. I expected some more messages about it, like the definition of the actual location of the vent.

Are you talking about a package *containing* a sealed battery? And should the venting be on this outer case, rather than on the battery?

You shall not tamper with the original battery case, no matter what. And especially not most of the lithium-based ones. They have to remain metal-to-metal hermetic seal-protected, otherwise they will most likely explode. The batteries are manufactured in extra dry atmosphere because of the danger of a lithium / water reaction.

Normal "vents" are not venting anything, except when due to some other failure the internal pressure would otherwise cause the whole case to burst. It is deemed safer to have just a small metal, plastic, or rubber dot burst instead of the whole can.

Look at the aluminum cans of electrolytic capacitors. In the large cans, you may find a rubber dot on the top cover, others have a scored cross at the bottom. This weaker place is designed to burst before the pressure can become so high that the whole can explodes like a grenade.

As it sounds, your references to the venting needs may be interpreted to relate to your outer case. If anything develops an excessive pressure inside, the case may or may not hold it. If it does, you get everything on your face next time you open it. Maybe this is what they mean with the venting need.

You could avoid the problems by making the opening a gradual process-enough small screws or by providing one needle valve-type opening. A high-pressure test chamber we have offers access through a 7″ diameter lid. But there is a thick chain on the lid, attached to a conical-tip screw that has to seal the "vent" hole during operation. The chain is so short that the screw has to be completely removed before the lid can be loosened. Thereby, the pressure is released through the small hole rather than the big lid.

OK, making a watertight vent may not be too relevant, if a battery bursts inside the case. What do you care about watertightness *after* that damage? But if you do, the device you need to consider is called a check valve. It allows flow in one direction only. It is available from several valve and fitting manufacturers. Skinner has two models you might find usable. Parker and Cajon come to mind as other sources worth checking.

Msg#: 4198
From: Janusz Suchorolski To: Pellervo Kaskinen

"Thou shall not tamper with sealed batteries!" Period!

Never in my mind did I think about causing any intentional "venting" of the battery's capsule, especially lithium or NiCd! That was probably due to a not-so-fortunate choice of words in my previous postings.

What I want to achieve is a sort of "preventive" venting of the sealed outer case (battery's compartment) once there's a hydrogen buildup inside it over a long period of time (from chemical self-reaction inside the battery), or a battery leak.

A check valve is probably the correct wording for such a gadget. Thanks! I will investigate it further with the companies you've mentioned (Skinner, Parker, and Cajon).

By the way, Sensym's Duane Tandeske makes some source suggestions in one of their app notes: Del Technical Engineering and Balston Filter Products. The first one's product is “negative” (too big), and I’m still waiting for an answer from Balston. However, what I need is a small-size vent (you’ve named it “needle valve-type opening,” I call it a “nipple”). Main features needed: to be small, provide ventilation, and protect from moisture.

But then again, a big question remains. Do I really have to vent a sealed enclosure that has two or four size C or D lithium batteries inside? What supports or rejects a need for that? What do IS, UL, FCC, CSA, DIN, etc. specs have to say in cases like this to meet and pass their requirements and testing?

Msg#: 3854
From: Ken Simmons To: Janusz Suchorolski

Let’s see what I can think of.

“Slitted” rubber sheeting clamped between support plates where the slit is tightly closed until venting is necessary. The slits would be really small, say no more than ⅛″ long. This would work in a nonimmersion applications where all you want to provide is protection from rain or splashed water.

A one-way pressure valve with the proper “popping” threshold (1 or 2 PSI?) can be used in immersion application as the valve is hermetic until it pops. Of course, it should be able to overcome any outside water pressure or it’ll be useless.

That’s all I can think of offhand.

Msg#: 3983
From: Janusz Suchorolski To: Ken Simmons

I’m glad you’ve answered, Ken. A flat rubber plate punctured with a needle is fine for a home project, but I am in search for the “real” thing.

My search is still on, due to the simple fact that I’ve seen such pressure-relief valves in the past, but I have not tried to source manufacturer(s). Now, that I need it, the pressure is on. :–)

Speaking of which, at least one submersible pressure transducer has something similar to allow for a barometric pressure compensation. It seems very close to what I would like to investigate further, but the vent’s I/O had to stay floating above water level all the time.

A prospective pressure-relief valve should be reasonably small, easy to install, watertight (i.e., operate without a problem when submersed under 50″ of water, approximately 2 PSI), and available off the shelf.

> Where I work (Boeing Military], our PWAs are
> encapsulated with either silicone or urethane resin
> (Hysol, PC-18M) so they can handle salt-fog, dust, and
> humidity without failing.

[Editor’s Note: *This quote is from the “Bus bar” thread featured in last month’s ConnecTime* (INK *72).]*

Quoting your reply from some time ago, I wonder if you’ve come across a manufacturer of small pressure vents in the past? Maybe there is some sort of “briefing skin” contraption that does a pressure relief/compensation job both ways, plus withstands a couple of feet of water (without humidifying the inside)?

Another chance is that I might be OK without venting at all. The size of the battery compartment will exceed the battery’s volume by at least factor of three. My understanding is that by having all that extra volume, I’ll allow the worst-case scenario (i.e., a lithium battery explosion) to be well encapsulated within the box. That extra, initially empty volume space should prevent a rupture of the box and keep an exploded view of the lithium battery well inside.

Is there anybody else with an expertise in encapsulation, venting, or pressure compensation? Anything else that needs to be added to this from an intrinsically safety point of view?

Msg#: 4022
From: Ken Simmons To: Janusz Suchorolski

> I’m glad you’ve answered, Ken. A flat, rubber plate,
> punctured with a needle is fine for a home project, but I
> am in search for the “real” thing.

Well, I gave you my best shot. :-)

> A prospective pressure-relief valve should be
> reasonably small, easy to install, watertight, and
> available off the shelf.

I think you can find such critters commercially. A rating of 2 PSI doesn’t sound too unreasonable as a “popping” value for screw-in relief valves.

> Quoting your reply of some time ago, I wonder if you’ve
> come across a manufacturer of small pressure vents in
> the past?

I’m sorry, I don’t do any kind of procurement or design suggestion, so I can’t offer anything offhand.

> Another chance is that I might be OK without venting
> at all. The size of the battery compartment will exceed

CONNECTIME

product is "negative" (too big), and I'm still waiting for an answer from Balston. However, what I need is a small-size vent (you've named it "needle valve-type opening," I call it a "nipple"). Main features needed: to be small, provide ventilation, and protect from moisture.

But then again, a big question remains. Do I really have to vent a sealed enclosure that has two or four size C or D lithium batteries inside? What supports or rejects a need for that? What do IS, UL, FCC, CSA, DIN, etc. specs have to say in cases like this to meet and pass their requirements and testing?

Msg#: 3854
From: Ken Simmons To: Janusz Suchorolski

Let's see what I can think of.

"Slitted" rubber sheeting clamped between support plates where the slit is tightly closed until venting is necessary. The slits would be really small, say no more than ⅛″ long. This would work in a nonimmersion applications where all you want to provide is protection from rain or splashed water.

A one-way pressure valve with the proper "popping" threshold (1 or 2 PSI?) can be used in immersion application as the valve is hermetic until it pops. Of course, it should be able to overcome any outside water pressure or it'll be useless.

That's all I can think of offhand.

Msg#: 3983
From: Janusz Suchorolski To: Ken Simmons

I'm glad you've answered, Ken. A flat rubber plate punctured with a needle is fine for a home project, but I am in search for the "real" thing.

My search is still on, due to the simple fact that I've seen such pressure-relief valves in the past, but I have not tried to source manufacturer(s). Now, that I need it, the pressure is on. :-)

Speaking of which, at least one submersible pressure transducer has something similar to allow for a barometric pressure compensation. It seems very close to what I would like to investigate further, but the vent's I/O had to stay floating above water level all the time.

A prospective pressure-relief valve should be reasonably small, easy to install, watertight [i.e., operate without a problem when submersed under 50" of water, approximately 2 PSI), and available off the shelf.

> Where I work (Boeing Military), our PWAs are
> encapsulated with either silicone or urethane resin
> (Hysol, PC-I8M) so they can handle salt-fog, dust, and
> humidity without failing.

This quote is from the "Bus bar" thread ConnecTime72).JK

> Maybe there is some sort of "briefing skin" contraption that does a pressure relief/compensation job both ways, plus withstands a couple of feet of water (without humidifying the inside)?

Another chance is that I might be OK without venting at all. The size of the battery compartment will exceed the battery's volume by at least factor of three. My understanding is that by having all that extra volume, I'll allow the worst-case scenario (i.e., a lithium battery explosion) to be well encapsulated within the box. That extra, initially empty volume space should prevent a rupture of the box and keep an exploded view of the lithium battery well inside.

Is there anybody else with an expertise in encapsulation, venting, or pressure compensation? Anything else that needs to he added to this from an intrinsically safety point of view!

Msg#: 4022
From: Ken Simmons To: Janusz Suchorolski

> I'm glad you've answered, Ken. A flat, rubber plate,
> punctured with a needle is fine for a home project, but I
> am in search for the "real" thing.

Well, I gave you my best shot. :-)

> A prospective pressure-relief valve should be
> reasonably small, easy to install, watertight, and
> available off the shelf.

I think you can find such critters commercially. A rating of 2 PSI doesn't sound too unreasonable as a "popping" value for screw-in relief valves.

> Quoting your reply of some time ago, I wonder if you've
> come across a manufacturer of small pressure vents in
> the past?

I'm sorry, I don't do any kind of procurement or design suggestion, so I can't offer anything offhand.

> Another chance is that I might be OK without venting
> at all. The size of the battery compartment will exceed

> the battery's volume by at least factor of three. My
> understanding is that by having all that extra volume, I'll
> allow the worst-case scenario (i.e., a lithium battery
> explosion) to be well encapsulated within the box.
> That extra, initially empty volume space should
> prevent a rupture of the box and keep an exploded
> view of the lithium battery well inside.

Given this new information, I really don't think you'll need a relief valve. You might want to consider putting a partial vacuum in the battery compartment, given the stated volume.

If the compartment is totally sealed, maybe even a slight pressurization is in order, along with a higher-value pressure-relief valve, to guarantee watertightness (i.e., a 2-PSI initial pressurization and a 4-PSI poppet valve).

Msg#: 4019
From: Lyndon Walker To: Janusz Suchorolski

> A prospective pressure-relief valve should be
> reasonably small, easy to install, watertight, and
> available off the shelf.

I missed your first message, so this may be way off the mark, but most (all?] 4x4 pickup trucks have differential breather vents which are closed so the truck can cross deep water but open when the air pressure inside the axle goes up. Perhaps try a parts dealer? They're about ½″ diameter and 1 ½–2″ tall.

McMaster-Carr lists some pressure-relief valves, but the trip points are fairly high-IO+ PSIG.

Another thought may be to try the pneumatics people like Clippard Minimatic (513/521-4261) or Bimba.

Msg#: 4082
From: Janusz Suchorolski To: Lyndon Walker

Thanks Lyndon, will try to contact manufacturers you've mentioned and report the results.

Msg#: 3909
From: Jan Verhoeven To: Janusz Suchorolski

> I wonder if there is an "official" way for providing a
> vent to a sealed battery enclosure.

If the manufacturer sealed it, vent-proof, then he will have a firm reason why *not* to place a vent on it. I would have more faith in the know-how of the maker.

> Adding such a vent should not jeopardize watertightness
> as well. Any suggestions?

Normal liquid acid/lead batteries use vents so that the oxygen and hydrogen can escape during the charging phase. Gel cells contain no vents at all and still they use the same chemistry. The developer made arrangements such that venting was not necessary anymore.

Msg#: 3919
From: Pete Chomak To: Jan Verhoeven

FWIW, all of the gel ceils I have seen still have a pressure-relief valve, usually just a rubber piece with a tiny hole in it.

We *invite you to call the Circuit Cellar BBS and exchange messages and files with other Circuit Cellar readers. It is available 24 hours a day and may be reached at (860) 871-1988. Set your modem for 8 data bits, 1 stop bit, no parity, and 300, 2200, 2400, 9600, or 14.4k bps.*

ARTICLE SOFTWARE

Software for the articles in this and past issues of *Circuit Cellar INK* may be downloaded from the Circuit Cellar BBS free of charge. It is also available on the Internet at **http://www.circellar.com/**. For those with just E-mail access, send a message to **info@circellar.com** to find out how to request files through E-mail.

For those unable to download files, the software is also available on disk. Software for issues prior to 1995 comes on a 360-KB IBM PC-format disk, one issue per disk. For issues from 1995 on, software comes on a 1.44-MB PC-format disk, with three issues per disk. Disks cost just $12 each. To order Software on Disk, send check or money order to: Circuit Cellar INK, Software On Disk, P.O. Box 772, Vernon, CT 06066, or use your Visa or Mastercard and call (860) 8752199. Be sure to specify the issue numbers with your order. Please add $3 for shipping outside the U.S.

434 Very Useful 435 Moderately Useful 436 Not Useful

PRIORITY INTERRUPT

How to Not Take It in the Chops*

Way back in *INK* 2, *I* described the relationship between the core audience of a magazine and publication revenues. Think of carefully breaking an egg in the center of a spinning turntable. At the center is a well-defined and boundaried area (the yolk) surrounded by an expandable unboundaried area (the white). Metaphorically speaking, the yolk is the core readership, while the white is the general-interest audience.

The rotation rate of the turntable represents the complex compromise of editorial, reader, and revenue objectives. When the rate of spin is reasonable, following a well-directed path, the yolk remains intact and the white expands out uniformly. An unbalanced relationship risks spilling everything.

INK is at a critical juncture. While you've known it all along, the commercial and advertising communities have finally recognized that *INK* is a serious source of technical wisdom and, more importantly, a place where their customers hang out. Usually these success stories or questions about the future are not discussed with the readers. Instead, one day your trusted and favorite magazine shows up on your doorstop looking like a trade magazine or other "sold its soul to the devil" rag. Many of you certainly remember when *BYTE* spun up the platter and all us techies spilled off the edge.

I don't want that to happen again. At the same time, I can't ignore the fact that we have created a vehicle where techies are taken seriously again, I'd like to use that prestige to press issues of collective importance. Raising circulation increases both our prominence within the industry and secures our ability to continue providing what you've come to respect and enjoy.

Some publications simply add warm bodies with a mailing address. It is expedient but also the worst thing I could do to you. For a price, Ed McMahon and his crew will find 10,000 or 25,000 new subscribers for us. But, given how these promotions are sold, we might be swamped by 5000 new readers screaming for video-game reviews. You never know what you'll get or how you have to change editorial to accommodate the unknown.

As I contemplated the horrendous and costly task of doing mass mailings to find more people like us, I came to the undisputable conclusion that while none of us can identify a thousand potential new readers, there are thousands of us who know at least one new right person. Similar minds will keep the platter speed in check.

I need your help in finding the right kind of new subscribers (sorry, Mr. McMahon). When you find that person, have them fill out the subscription card in the magazine (or any vehicle with the same info) and put your name and subscription ID where indicated. For every new subscription, I'll extend your subscription three issues free. Find a cooperative computer club and get 12 subscriptions? You get an additional three years-free!

In addition, to help those of you who don't dare show your own *INK* copy around because you don't want to risk losing it, I'll be sending an extra magazine especially labeled as a give-away copy to a group of you each month specifically for that purpose. Pass it around and get credit for the subscriptions.

Success is a hard thing to pass up. I also remember that it took all of you to get us here. Help me find the right people that will allow us to stay the course. You have an opportunity to participate in controlling the platter speed. The last time we were on this merry-go-round, we got screwed.

Steve

steve.ciarcia@circellar.com

*For our international readers, "chops" is slang for "jaw".