www.circuitcellar.com

CIRCUIT CELLAR®

THE MAGAZINE FOR COMPUTER APPLICATIONS

#126 JANUARY 2001

DATA ACQUISITION

The Geo-Sentry Vibration Monitor

Java for Embedded Sytems

A Home PCB Lab

Understanding Second-Order Systems



12 **The Geo-Sentry**
A System with Brawn
David Penrose

20 **A Class Act**
Making Your Own PCBs
Paul Verhage

28 **Designing for Reliability, Maintainability, and Safety**
Part 2: Digging Deeper
George Novacek

54 **Embedded Living**
What's New on the Home (Automation) Front?
Mike Baptiste

62 **Micro Series**
Fundamentals of Second-Order Systems
Part1: Where It All Began
David Tweed

70 **From the Bench**
Where's Waldo?
Pinpointing Location by Interfacing with a GPS Receiver
Jeff Bachiochi

78 **Silicon Update**
Sweet Sixteen
Tom Cantrell

Task Manager 6
Rob Walker
The Year in Preview

New Product News 8
edited by Harv Weiner

Reader I/O 11

Test Your EQ 84

Advertiser's Index 95
February Preview

Priority Interrupt 96
Steve Ciarcia
Electronic Evolution

INSIDE ISSUE 126

EMBEDDED PC

40 **Nouveau PC**
edited by Harv Weiner

42 RPC **Real-Time PCs**
A Cup of Java
Part 2: Standard Java in Real-Time Embedded Systems
Ingo Cyliax

48 APC **Applied PCs**
Rabbit Season
Part 5: A Board You Can Afford
Fred Eady

ChipCenter

LOGIN/REGISTER

SITE NAVIGATOR

Buy Parts Now!

- myChipCenter
- Knowlege Centers
- Guides & Experts
- Product Reviews
- Communities
- Online Tools
- Circuit Cellar
- Resources
- Real Life

SEARCH CHIPCENTER

ASK US

THE ENGINEERS TECH-HELP RESOURCE

Let us help keep your project on track or simplify your design decision. Put your tough technical questions to the ASK US team.

The ASK US research staff of engineers has been assembled to share expertise with others. The forum is a place where engineers can congregate to get some tough questions answered, or just browse through the archived Q&As to broaden their own intelligence base.

★★★★★★★★★★

Test Your EQ
8 Additional Questions

CIRCUIT CELLAR ONLINE

THE MAGAZINE FOR COMPUTER APPLICATIONS

Circuit Cellar Online offers articles illustrating creative solutions and unique applications through complete projects, practical tutorials, and useful design techniques.

This Month | Archive | About Us | Contact | Looking for More?

BREADBOARDING

by Stuart Ball

With this offering, Stuart pumps new life into the dying technique of prototyping. For electronics experimenters, the cost of making a circuit board is too high for building only one. There is another choice. One-of-a-kind prototypes can be produced without having to lay down the cash for printed circuit boards. Hand-wiring may seem laborious, but you'll end up with an effective low-cost alternative. December 2000 GO

IMPLEMENTING A SIMPLE USB INTERFACE FOR AN EMBEDDED PROCESSOR

by Stuart Allman

When connecting modules in your embedded system, decisions need to be made. The universal serial bus (USB) may help you make up your mind and ease the aches of development. With the news that Intel is phasing out the good old RS-232 and replacing it with the USB, Stuart shows us that all is not lost, because the USB should work just as well. He supplies us with all the info and leaves no room for excuses, so get ready to make a game plan for your next design. December 2000 GO

DETERMINING MEASUREMENT ACCURACY

by Hristo Stefanov

To measure different signals, you've got to define the terms of measurement. Most of us use the term "error" in assessing uncertainties. Because every measurement has some amount of error, there is always inaccuracy, which can be attributed to many factors. In this article, Hristo takes us through a series of different types of errors, from random to systematic. December 2000 GO

THROUGH THE LOOKING GLASS

by David Brobst

Taking a Look at the PIC18C*xxx* Series

If you're looking for high-end, look no further than Microchip's PIC18C*xxx* family of 8-bits. Designed with the future in mind, its direction seems clear—with increased clock speed, hardware multiplier, and higher resolution A/Ds, this new unveiling is positioned as a low-end DSP solution. With intriguing features like the table read/write capability, David shows that the PIC18C*xxx* is a valuable addition to the growing line of controllers. December 2000 GO

SIMPLE THINGS

Lessons From the Trenches

by George Martin

Prime Numbers Revisited

This month, George looks at design problems and examines where to look for the answers. He explains that, typically, he calls on a hardware engineer for logic problems and the like, and a software engineer for help with pure software tweaking. Sometimes he's led down a blind alley, but shows us that by tapping into all of the sources available, you could be surprised to find that the answers may come from the most unlikely places. December 2000 GO

RAMBLIN' MAN

Silicon Update Online

by Tom Cantrell

The name of the game is networking and automotive electronics is the target. In this article, Tom predicts a time when cars will be Internet-enabled, but the standards for such road warriors are in the limelight as well. Cars and chips together open up a wide array of possibilities, but when car manufacturers roll their own standards, you could find yourself tripped up on new wires. December 2000 GO

Mfr Data Sheets
App Notes
Ask Us

SuperSearch
Search Type: Part Number
Search for: *lm317* GO



SAVE DOWNLOAD TIME

Order your own complete set of *Circuit Cellar Online* archives.

First 6 issues July-Dec 1999 are available.

CD Includes:
All HTML files
Article PDF
Code files

Navigate and use just as you would online.

(no Internet connection required)



RESOURCE LINKS

- Relays
  Bob Paddock
- Bluetooth Technology
- PDA's
  Rick Prescott

TASK MANAGER

The Year In Preview

Every journey begins with the first step. As an amateur photographer, I'm more of a fanatic planner than a random wanderer when it comes to travelling. I find out the exact times of sunset, moonrise, low tide, and such so I can be in the right places at the best times. Getting there can be half the fun, but for me, it's all about the destination.

Unfortunately, this journey through the new millennium is a rather open ended thing, so I can't exactly plan out what I want to see and where I need to be at what times. I guess I'd better enjoy it one step at a time.

As readers, you get to enjoy *Circuit Cellar* one issue at a time. Because about five months of preparation goes into each issue, I can honestly say that I've seen the future and although I don't know what kind of projects *Circuit Cellar* readers will be designing 1000 years from now, the events of the next 12 months are a bit less fuzzy.

- January: This month we're introducing the electronic edition of *Circuit Cellar* (pg 31, 96) and ending the Driven to Design (D2D) contest (pg 10).
- February: Get ready for the kickoff of the Atmel Design Logic 2001 contest and some great articles on Signal Processing.
- March: Robotics fans can look forward to information on a variety of robotics competitions and projects from them.
- April: Stop by our booth at the Embedded Systems Conference in San Francisco or our web site to see the results of the D2D contest.
- May: See how some of the Design2K winners incorporated Measurement and Sensors into their projects.
- June: Read the project abstracts for the D2D winning projects and finish your entry to the Design Logic 2001 contest.
- July: Take a closer look at the challenges of Embedded Programming and visit us at the Embedded Systems Conference in Chicago.
- August: See what kind of Internet & Connectivity projects are out there and find out about *Circuit Cellar*'s 16th design contest.
- September: Join us at the Embedded Systems Conference in Boston for the announcement of the Design Logic 2001 winners.
- October: Not everyone has "gone digital" so we'll discuss some Analog Techniques that can get the job done.
- November: If you think Wireless Communications is a hot topic now, wait and see what kind of projects *Circuit Cellar* authors are working on!
- December: In the last issue of the year, we'll show you some of the Design Logic 2001 winning projects and reintroduce the Graphics & Video theme.

There, just a little planning and I already feel better about the journey into this new millennium. So, whether you hit the ground running or are content to mosey along and take the new millennium as it comes, I wish you well on your journey.

Rob

rob.walker@circuitcellar.com

CIRCUIT CELLAR®

THE MAGAZINE FOR COMPUTER APPLICATIONS

EDITORIAL DIRECTOR/PUBLISHER
Steve Ciarcia

MANAGING EDITOR
Rob Walker

TECHNICAL EDITORS
Jennifer Belmonte
Rachel Hill
Jennifer Huber

WEST COAST EDITOR
Tom Cantrell

CONTRIBUTING EDITORS
Mike Baptiste Ingo Cyliax
Fred Eady George Martin
George Novacek

NEW PRODUCTS EDITORS
Harv Weiner
Rick Prescott

PROJECT EDITORS
Steve Bedford Bob Paddock
James Soussounis
David Tweed

ASSOCIATE PUBLISHER
Joyce Keil

CHIEF FINANCIAL OFFICER
Jeannette Ciarcia

CUSTOMER SERVICE
Elaine Johnston

ART DIRECTOR
KC Zienka

GRAPHIC DESIGNERS
Naomi Hoeger
Mary Turek

STAFF ENGINEERS
Jeff Bachiochi
John Gorsky

QUIZ MASTER
David Tweed

EDITORIAL ADVISORY BOARD
Ingo Cyliax
Norman Jackson
David Prutchi

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

ADVERTISING

ADVERTISING SALES REPRESENTATIVE
Kevin Dows Fax: (860) 871-0411
(860) 872-3064 E-mail: kevin.dows@circuitcellar.com

ADVERTISING COORDINATOR
Valerie Luster Fax: (860) 871-0411
(860) 875-2199 E-mail: val.luster@circuitcellar.com

ADVERTISING CLERK Sally Collins

CONTACTING CIRCUIT CELLAR

SUBSCRIPTIONS:
INFORMATION: www.circuitcellar.com or subscribe@circuitcellar.com
TO SUBSCRIBE: (800) 269-6301, www.circuitcellar.com/subscribe.htm, or subscribe@circuitcellar.com
PROBLEMS: subscribe@circuitcellar.com
GENERAL INFORMATION:
TELEPHONE: (860) 875-2199 FAX: (860) 871-0411
INTERNET: info@circuitcellar.com, editor@circuitcellar.com, or www.circuitcellar.com
EDITORIAL OFFICES: Editor, Circuit Cellar, 4 Park St., Vernon, CT 06066
AUTHOR CONTACT:
E-MAIL: Author addresses (when available) included at the end of each article.

For information on authorized reprints of articles,
contact Jeannette Ciarcia (860) 875-2199 or e-mail jciarcia@circuitcellar.com.

CIRCUIT CELLAR®, THE MAGAZINE FOR COMPUTER APPLICATIONS (ISSN 1528-0608) and Circuit Cellar Online are published monthly by Circuit Cellar Incorporated, 4 Park Street, Suite 20, Vernon, CT 06066 (860) 875-2751. Periodical rates paid at Vernon, CT and additional offices. **One-year (12 issues) subscription rate USA and possessions $21.95, Canada/Mexico $31.95, all other countries $49.95. Two-year (24 issues) subscription rate USA and possessions $39.95, Canada/Mexico $55, all other countries $85.** All subscription orders payable in U.S. funds only via VISA, MasterCard, international postal money order, or check drawn on U.S. bank.

Direct subscription orders and subscription-related questions to Circuit Cellar Subscriptions, P.O. Box 5650, Hanover, NH 03755-5650 or call (800) 269-6301.

Postmaster: Send address changes to Circuit Cellar, Circulation Dept., P.O. Box 5650, Hanover, NH 03755-5650.

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

The information provided by Circuit Cellar® is for educational purposes. Circuit Cellar® makes no claims or warrants that readers have a right to build things based upon these ideas under patent or other relevant intellectual property law in their jurisdiction, or that readers have a right to construct or operate any of the devices described herein under the relevant patent or other intellectual property law of the reader's jurisdiction. The reader assumes any risk of infringement liability for constructing or operating such devices.

Entire contents copyright © 2001 by Circuit Cellar Incorporated. All rights reserved. Circuit Cellar and Circuit Cellar INK are registered trademarks of Circuit Cellar Inc. Reproduction of this publication in whole or in part without written consent from Circuit Cellar Inc. is prohibited.

READER I/O

BY GEORGE...

I'm about to read the December article of George Novacek's new series and I just wanted to say that I'm glad to see him back!

His previous three-part series ("The Joys of Writing Software", *Circuit Cellar* 120–123) was really on the nose. Until a couple of months ago I was a software tester (IBM: terminal emulation products, published programming APIs). One of his articles, directed to the function (attitude) of testing, was really right on target…based on my experience.

I also learned new things from the other two articles. Now I am on the development team (analyzing reported bugs, fixing bad code) for the products I used to test.

Novacek really does have broad, in-depth experience, and clearly knows what he is talking about. I'll be reading the current series to pick up more ideas to apply to my job! Yep, I can take lessons from hardware and think software.

Thank you for a great magazine that gets read each month (unlike some other of my subscriptions).

Steve Modena

HOOKED ON CONTESTS

I just finished reading the December *Circuit Cellar* cover to cover and wanted to comment about Steve's "Priority Interrupt".

I think it's great that *Circuit Cellar* continues to do the micro contests. I've debated entering contests in the past, but finally I have committed myself to do the Zilog Z183 contest.

I finished wire wrapping my board (also soldering the 100 pin sample chip on a PCB adapter underneath a microscope) and wrote my first assembly program for the Z183 chip—the infamous "Blink the LED" program that every EE writes first when working with a new processor. I tried it out and it worked! I started late, but think I can make the January deadline. This is my first experience with a Zilog chip, it's not like programming a PIC that's for sure.

Anyways, back to the contests, this contest has given me the incentive to use a new microprocessor. I also plan on entering the Atmel AVR contest in February. I use FPGAs now and am dying to use one with a built in micro. The AVR processor will be nice to have under my belt as well. Hopefully I can do that along with building a fire-fighting robot for the Trinity contest (your magazine has also hooked me on that one as well). Right now, it sounds like a good application for the Z183.

So, thanks for publishing such a great magazine. I'm hooked for sure.

Bruce Pride

Editor's note: In the November issue, the new product listing for Allison Technology contained the incorrect product information. The proper information is listed below:

HAND-HELD DIGITAL OSCILLOSCOPE

The **HH972** is a hand-held digital storage oscilloscope and in-circuit component curve tracer. The device provides two useful test instruments in one package the size and weight of a digital voltmeter and at a price more in line with that of a DVM than an oscilloscope or a component curve tracer.

The hand-held is equipped with many features. It is capable of auto-ranging in oscilloscope mode and displays I/V curves for in-circuit component testing. The device has a wide 5 MHz bandwidth (digital sampling at 20 Ms/sec). Powered by an internal 9-V battery or AC wall adapter the unit utilizes a high contrast, backlit, 150 × 100 dot LCD display with wide angle visibility. Amplitude and time resistor values can be display on the screen. The device is able to take 280 V true RMS AC or ±400V DC maximum input voltage.

With all these features the device is well suited for a wide variety of applications. Its small size and powerful digital storage oscilloscope capabilities make it an natural choice for both bench and field service applications. Its low cost also makes it suitable for educational and hobby use.

The device is priced at **$259.00**.

Allison Technology
Phone: (281) 239-8500
Fax: (281) 239-8006
Web site: www.atcweb.com

FEATURE ARTICLE

Design2K Contest WINNER

David Penrose

The Geo-Sentry

A System with Brawn

Looking for a monitoring system that's all ears? Post the Geo-Sentry on watch and listen up as David covers the design details of his Design2K-winning project that senses ground vibrations and displays events on a fluorescent display screen.

This project could have been called the "ear-to-the-ground" project with 20 ears. The ears are small seismic sensors that sense minute vibrations and convert them to electrical energy. The sensors are integrated into a real-time monitoring and recording system where sampled inputs are recorded into a 3000-entry circular buffer at up to 60 samples per second. The buffer is emptied by separate processes that display the events on a 4 × 20 fluorescent display or print them. The system can send an X10 command or sound an alarm based on an event exceeding a threshold. Or it exports the raw event data to a PC for storage and analysis.

Although the buffer is 3000-events long, it would soon fill if all data was sampled and recorded at the maximum rate. The system provides a complete set of filters that allows you to define what type of data is stored. It can store by sensor: every event at the 60-Hz rate, every event summarized once per second, every event that trips an alarm threshold, or every X10 command read from the X10 interface.

A similar set of filters exists to define what gets sent to the fluorescent display and what gets passed to the printer port. The operator interface also can name each sensor so that the data is reported in a meaningful fashion. It can set alarm thresholds for each sensor, associate X10 commands with alarms, and so on. All data is time-tagged by a real-time clock to 60-Hz resolution with year, month, day, day of week, hour, and minute recorded.

All this functionality is accomplished with one microprocessor (Philips' 80C451), a SmartWatch from Dallas Semiconductor, and one RAM and PROM chip. It is a powerful, compact system that can handle many different real-time monitoring and recording tasks. Photo 1 shows the processing and display unit together with one of the assembled sensor units. Figure 1 presents an overview of the project's components.

HARDWARE

Although Philips discontinued it, the 80C451 still is one of my favorite microprocessor chips. The 80C451 works like the baseline 8051 microprocessor; so, your hardware and software

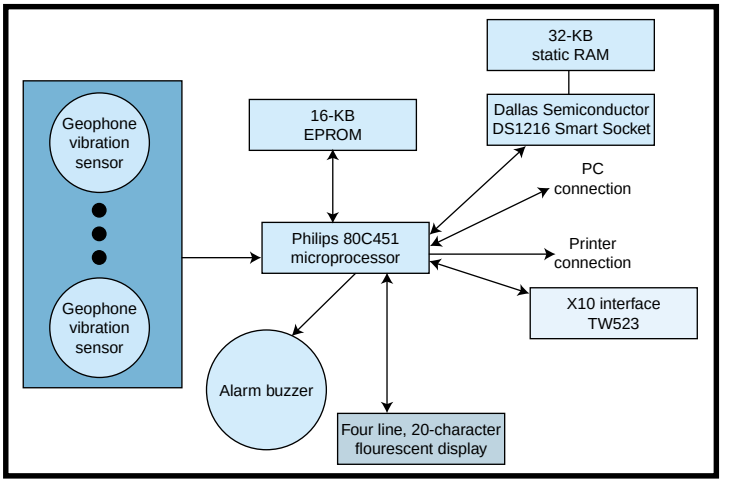


Figure 1—*The Geo-Sentry block diagram shows the vibration-sensing array. There are up to 20 remote sensors whose outputs are processed by the 80C451 microprocessor. The 80C451 records all vibration events and formats the data for display, printing, or generation of X10 events.*

knowledge of the 8051 system is applicable. The 80C451 has seven fully functional ports. Each port has 8 bits (in the PLCC package), with a total of 56 lines to interface to the outside world. You still have pins to play with after using some external memory.

The Geo-Sentry can be implemented with the standard 8051, but if you want to process 20 external inputs, multiplexing chips need to be added. Modifications to the software probably would be less than 10 or so lines to address the multiplexor. This is a great example of the '51 manufacturers maintaining almost perfect compatibility with this family of processors (see Figure 2).

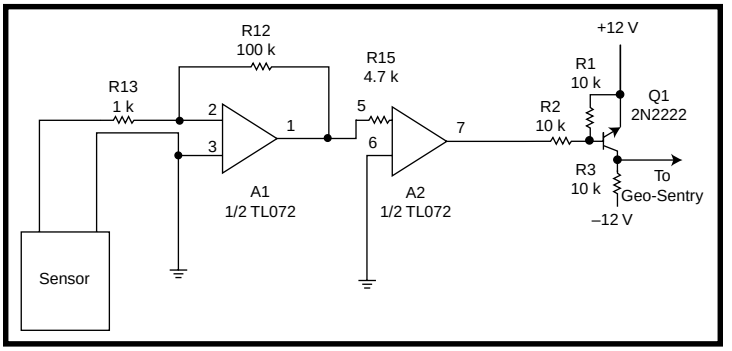


Figure 2—*The Geo-Sentry sensors are shown here. The output of the geophone sensor is converted to a square wave by the A1/A2 op-amp. The transistor then drives the long line to the Geo-Sentry. The ±12-V power supply is decoupled by 10- and 0.1-µf capacitors at both positive and negative supplies.*

A driving requirement of building the Geo-Sentry system was that it has to be fast and have nonvolatile storage. The number of sensors being examined and the sampling rate dictated the speed requirement. The nonvolatile requirement ensures that no data is lost because of a power outage and that the user settings are preserved without power. The storage also has to allow unlimited writes and allow for asynchronous filling and emptying. A battery-backed FIFO chip could have been used, but it's expensive and would require additional memory for parameter storage or battery backup to the microprocessor onboard memory.

I used the SmartWatch instead. This socket allows a 32-KB static RAM chip to be plugged into it. The socket provides RAM battery backup and has an integrated real-time clock chip. This one socket solved all the memory problems and as a bonus provided the real-time clock function.

You can never be too rich nor have too much memory. When I began the Geo-Sentry, I knew that my dreams for programmability and functionality were going to chew up a lot of program storage space. To combat this loss, I configured the system to use external EPROM and currently have it wired to accept up to 32 KB. The EPROM and RAM are integrated through a 74L-

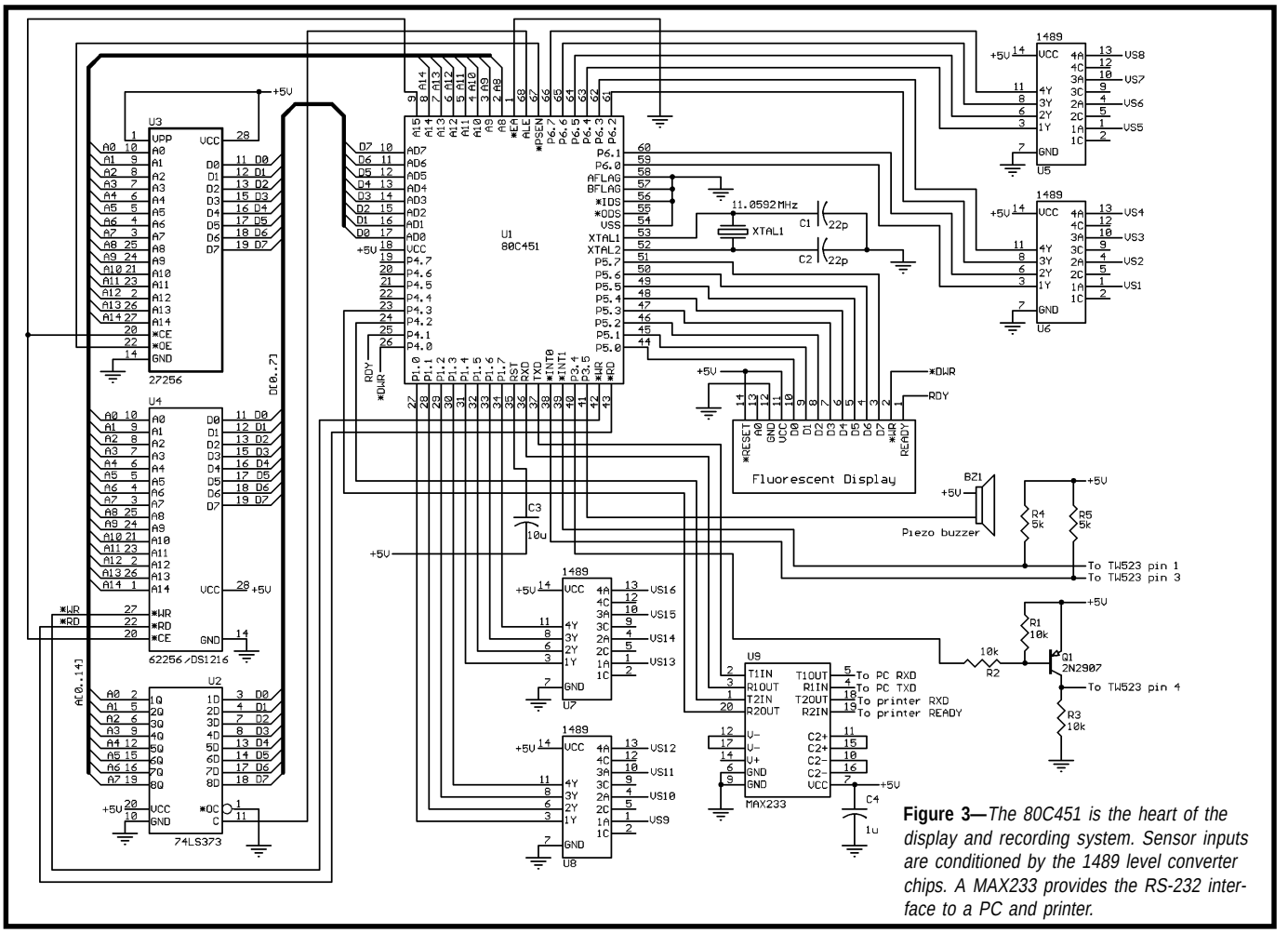


Figure 3—*The 80C451 is the heart of the display and recording system. Sensor inputs are conditioned by the 1489 level converter chips. A MAX233 provides the RS-232 interface to a PC and printer.*

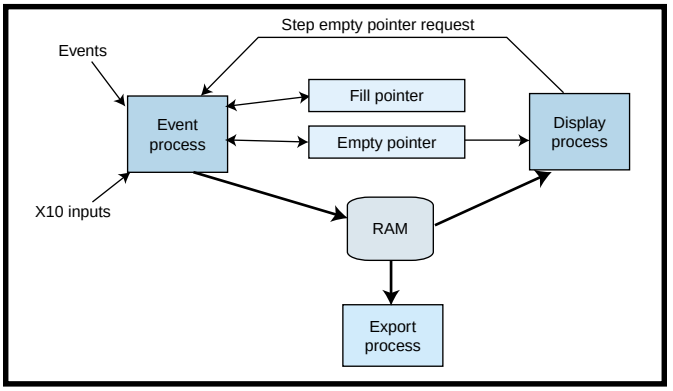


Figure 4—*The RAM buffer is at the center of the storage and display process. The process on the left fills data and manages all pointers. The process on the right displays data from the buffer. And, at the bottom is the export process that clears the buffer when done.*

S373 address latch, a standard interface for the '51 family of processors.

I chose Geosource's geophone sensors. These seismic sensors consist of a magnet suspended in a coil and are packaged in a metal container with two leads coming out of the top (see Photos 2 and 3). Figure 3 shows the components on the circuit board.

Geophone sensors were designed to do acoustic or seismic surveying of potential oil fields. In application, some seismic energy is pumped into the ground, and then an array of these sensors intercepts the returning energy as it reflects off different strata. The arrival times and energy levels are then interpreted to build a picture of the underground area. My application is not this sophisticated.

I placed an amplifier and a comparator next to the sensors and turned the analog output into a digital pulse stream. I was only interested in the number of vibrations per second rather than the levels of the vibration. This sacrifices some of the power of these units, but gives me more than enough information to detect footsteps or vehicle motion in a large area surrounding each sensor. The processor examines the bitstream from each sensor to determine if there has been a change since the last sample. Because this sampling is performed at 60 Hz, it picks up all normal vibrations.

I planned to deploy the sensors over a large area, hence each would drive a long length of wire. To keep things simple and cheap, I used 1489 RS-232 receiver chips for the processor and placed a transistor driver in each sensor to generate the ±12-V signal expected by these receivers. More sophisticated approaches were possible, but this simple approach works well. The outputs of the 1489 receivers are standard TTL and are connected directly to port pins on the 80C451.

The Geo-Sentry includes an interface to accept the outputs from a standard TW523 X10 interface and also to provide command inputs to this interface. Inside the Geo-Sentry, three port pins are dedicated to this function. One pin is the X10 zero crossing, which is a square wave following the 60-Hz power line signal. The second line is a receive line that only requires a pull-up resistor to drive a port pin. The third line is the transmit line that requires a transistor interface to drive the X10 interface.

BRIGHT DOESN'T ALWAYS MEAN SMART

The fluorescent display was selected to provide maximum visibility. This display is bright and comes with a number of filters to modify the apparent color. The Geo-Sentry currently uses a dark blue filter. The interface to this display is different enough from an LCD to cause a headache. It doesn't allow the simplified four-wire interface. It also responds differently to control codes than the standard LCD.

One nice feature is that the fluorescent display automatically scrolls data when the last character of a line is written. It takes a lot of software to accomplish this same function on an LCD. The timing for reads and writes is about the same as for LCDs, because the unit has a small processor, too.

SOFTWARE

Unfortunately, the simplicity of the hardware is not duplicated in the software. The software is essentially running two separate processes and integrating them through the RAM buffer (see Figure 4). One set of processes is examining the sensor array at the 60-Hz rate and looking for changes of state in the sensors. When a state change is detected, this process builds a 10-word entry that identifies the sensor, count of vibration events, and time of occurrence.

Because different types of events can be recorded, this format was designed to be flexible at the expense of storage space. The 8 bytes of time completely define the year, month, day, hour, minute, second, and hundredth of a second. The count word is provided because you can configure the event storage to happen only once per second, in which case many vibrations may have occurred.

This storage format also records any X10 commands that have been assembled from the X10 input line. The input line is read at each zero crossing of the 60-Hz power line frequency. Because the processor only interrupts on a high-to-low transition of this signal, I added an 8-ms timer to generate the second crossing.

Within 1 ms of these crossings, the data line is checked, and any detected X10 events are assembled and stored.

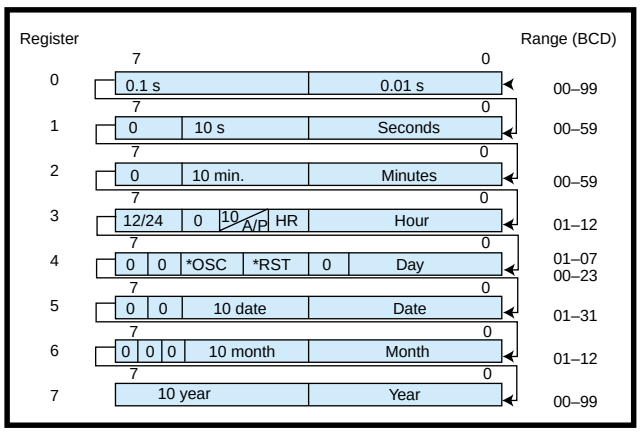


Figure 5—*This is taken from a Dallas Semiconductor application note. It shows the eight words of clock data containing the year, month, day of week, hour, minute, second, hundredth of a second, and operating mode.*

This provides a monitoring function for X10 transmissions and a foolproof readout of the X10 commands sent by the Geo-Sentry. Each X10 event transmission through the TW523 interface results in the command also being interpreted as an input available on the TW523 output line.

The second process is examining the circular buffer to determine when data is available for processing. This ensures that the buffer never fills without the data being processed. As data becomes available, it is processed to decide if it qualifies for display, alarm, X10 transmission, or printer output. After processing, a flag is set to allow the first process to advance this "empty" pointer to make sure that the two processes stay synchronized. The first process is interrupt-driven by the 60-Hz input signal from the X10 or from an 8-ms timer running in Timer0. The second process runs as a polling loop.

The great differences in processing time determine the buffer between the storage of events and action on these events. An X10 transmission is a lengthy process with two full words of command required and each word requiring about 1 s to transmit. Transferring the information to a printer is a slow process also, because the printer is connected through a slow serial interface.

I considered leaving the high-speed serial port available for the PC interface and providing a bit-serial output driven by software for the printer. With the high-speed serial port, a command will output printer data through the standard serial port, not the dedicated printer port.

The high-speed serial port is required for data export to the PC. Exporting is the only process that removes data from storage. All other processes leave the data intact until it needs to be written again. When you select export, the data is formatted and sent to the PC as text lines. After the data is exported, the circular buffer is cleared.

While resting, the display presents the current date and time on line one and the number of events on line two (see Photo 4). If an event that generates a display entry occurs, that entry is written to the last line, which raises all other lines by one. A 4-s timer is started at this point to delay the redisplay of time and event count.

As long as events continue to occur, this timer will continue being reset. While events are occurring, the display shows four lines of event data. Following a series of events and after the timer expires, the last two lines show the most recent data and the first two show the time and event count.



Photo 1—*The Geo-Sentry and one of the seismic sensors are displaying a vibration event. The date and time plus the number of stored events are displayed on lines one and two. Lines three and four show the time of the event and the name of the sensor that triggered the event.*



Photo 2—*The geophone sensor is the metal cylinder. The PC board has the dual op-amp, some resistors, a transistor, and decoupling capacitors.*

While these two cooperating processes are occurring to store and retrieve data, a part of the polling loop is examining the input line from the PC. All commands are single-character action requests. These requests are used to establish operating parameters for the unit. Parameters are stored in the battery-backed SRAM and are initially set to default conditions.

During boot, the software checks two cells in the SRAM for a defined bit pattern. If this pattern is not found, the software assumes that the memory is not initialized properly and will perform a cold start operation. A cold start clears memory pointers and flags and then establishes a default set of operating parameters. You can trigger the fresh start with the F command.

These parameters define what data is to be recorded, displayed, and printed. Data to be displayed, printed, or alarmed must be recorded first. The software makes sure these data selections are matched to recording requests. If there is a mismatch, you are warned and the recording flags are set automatically. It is possible to record data without any display, print, or alarm request because the data is still available for exportation.

Exporting is triggered by the X command, and all other processing is terminated. The process begins with the oldest data and outputs all valid data. A counter indicates the number of valid data points. It's shown on line two of the display to also indicate the number of events stored by the unit. The export data as well as the print data is presented in an expanded format where the year, month, day, day of week, hour, minute, second, and hundredth of a second are displayed. The normal display shows current time to the nearest second.



Photo 4—*The display is resting and displaying the last two lines of event data. In this case, it's an X10 event addressing house code G, unit 8, with an on command followed by an off command.*

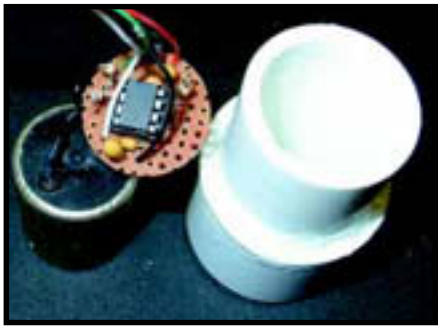


Photo 3—*The sensor and circuit board fit into a section of 1" PVC tubing with two end caps sealing the top and bottom. It's a tight fit.*

I assumed that the export data would be used for analysis and the precise time as well as the complete time would be required to identify data stored in the unit. At the end of the export operation, the circular buffer is reset and the event counter is zeroed. Also for these operations, you can use the kill command (K), which clears memory without exporting data.

Alarm sensitivities and alarm actions can be adjusted by the sensitivity (S) and alarm (A) commands. The S command allows you to specify how many vibration transitions occur 1 s before an alarm is declared. After an alarm is declared, the buzzer is turned on for <1 s and the event qualifies to send an X10 command.

The X10 command can be established by the A command. This allows you to link specific sensor alarms to sets of X10 house and unit codes. The software sends an on command to the selected house and unit code when the alarm occurs. Then, it begins a timer that sends an off command. An alarm doesn't have to trigger an X10 command, and the buzzer can be disabled. The alarm is stored as an information event for the export operation.

Sensor names are definable. All messages except for the high rate 60-s events are displayed, printed, and exported with a text name to define the sensor. In the default condition these names are Sensor 1, and so on. With the N command, you can assign a unique name that is 10-characters long to each sensor. Displays and other data are then more meaningful.



Photo 5—*The back view shows the RJ-45 connectors used for sensor inputs, PC connection, and X10 interface. The power connector is at the right.*

Three commands, examine (E), write (W), and memory (M), are useful only to the developer or technician. These allow you to examine a memory location, write data to a memory location, or perform a memory test and identify limits of memory. The E and W commands are designed so that both onchip and external memory can be read or modified. The distinction between them is that you enter two hex digits to access onboard memory and four hex characters to access external RAM. The commands are designed so that a carriage return at any time will step to the next cell location and perform either the E or W command depending on which was used last.

Setting the clock requires an easy interface. Setting a clock that has year, month, day, day of week, hour, minutes, seconds, plus fields to select 12- or 24-hour format involves a lot of data. I wanted a fast, easy way of doing this with a text interface (see Figure 5). I developed the logic that displays the current clock time with all of the fields displayed and lets you put the cursor below a selected field and then increase or decrease that field. The clock (C) command performs these actions.

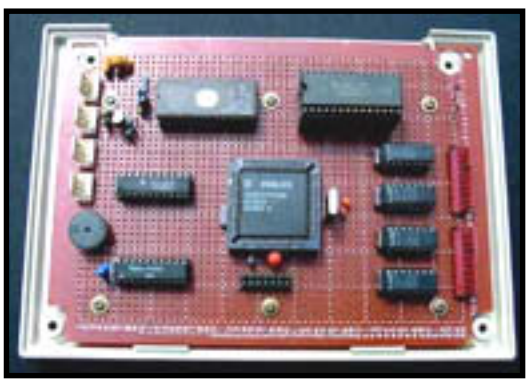


Photo 7—*The connectors are unplugged. Checkout is easier if the board can be worked on without wires getting in the way.*

When the C command is entered, the software displays the current date and time. This display is updated in place at 1-s intervals and the cursor is positioned initially at the first field, which is the current month. An operator prompt informs you that the different fields can be selected by inputting a greater or less than symbol, which will move to the next or previous field. The U or D command will then step up or down the field with wrap around at either end of the range. This is a foolproof method.

I developed the unit using a small hand-held terminal instead of the PC interface. This terminal is a four-line, 20-character display with a unique set of positioning codes that correlate with different rows and columns on the display. For the PC interface, I used the VT100 positioning commands because this mode was available in Hyperterm. This meant that I had to switch between these two modes to provide a meaningful display. The terminal, or T, command toggles between the VT100 mode and the handheld, or H, mode each time the T command is entered. The prompt will be "S-" (standard) or "H-" to identify the operating mode.

INTEGRATION

Everything on the Philips 80C451 works like an 8051, except when you use Port 6. This port was designed either for a Centronics interface or interface to another computer, so it has a few control lines that must be established. Lines AFLG, BFLG, and *IDS must be grounded if you expect to use the Port 6 pins as normal I/O. The specification sheet explains the additional function of these port pins well, but I learned the lesson of grounding these three pins a number of times anyway.

The SmartWatch needs careful watching. It uses a technique called "phantoming." Normally all access to the socket passes directly to memory, and the clock chip is the phantom behind the RAM. When it's necessary to talk to the clock, the D0 bit becomes a serial bitstream into and out of the clock.

To be able to use this one bit to communicate with the clock without getting mixed up in memory accesses requires a special sequence of 64 bits be sent to the socket. Dallas Semiconductor stated this long bitstream is a string that's unlikely to occur during normal operation of the circuit (see Figure 6). To read or write the clock, the datastream on D0 is sent to any memory address that will enable the SRAM. After the datastream has been written, the clock is open and current reads or writes will return/send data from/to the clock on D0.



Photo 6—*This view of the open case shows the connectors from the processing board to the offboard components. The display connector is at the center of the photo and the sensor inputs are at the right. The left connectors provide power and interface signals to the PC, printer, and X10 interface.*

This clock data is 8 bytes of data with date and time packed as BCD digits. This means that 64 reads or writes must occur to clock this data out of or into the clock chip. The safest process is to disable interrupts during this process, because any other reads or writes to the memory will look like a clock read, confusing both the interrupt and clock software.

A careful examination of the datasheet and application notes for the SmartWatch will save you from problems during development. I want to note caution about closing the clock during program initialization. I didn't do this, and randomly encountered all

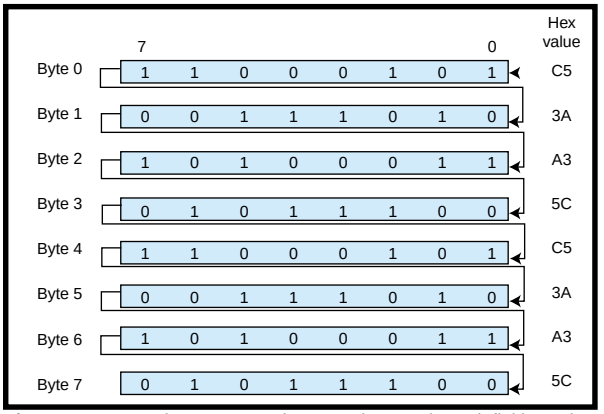


Figure 6—*Here's the SmartWatch comparison register definition. The pattern recognition in hex is C5, 3A, A3, 5C, C5, 3A, A3, 5C. The odds of this pattern being accidentally duplicated and causing inadvertent entry to the SmartWatch are less than 1 in 10^{19}. This pattern is sent to the SmartWatch LSB to MSB. This 64-bit pattern opens the clock for reading or writing.*

sorts of clock upsets. The clock may be open performing a read or write when the processor is shut off. Because the clock is backed up and still running, it waits for more bits. During initialization, all accesses to RAM talk to the clock. This can cause problems.

Dallas Semiconductor recommends that during initialization, read at least 65 bits from the clock before doing anything else. I now read nine words and the clock hasn't failed me again.

CONSTRUCTION

I planned to bury sensors in my yard, so I wanted a watertight, inexpensive enclosure. One-inch PVC pipe is tight for the sensors, but I bored the inside so they fit well. Each section of PVC pipe has one cap attached with PVC cement making a permanent joint.

I placed the cap on the other end and drilled a hole through the side at the top of the pipe. This hole allows the power and signal lines to pass through. I cut a number of round circuit board sections and mass assembled the single 8-pin op-amp on these boards together with some decoupling capacitors, the level setting resistors, and drive transistor (see Photo 2).

I then attached the sensor to the circuit board with flexible wire and inserted the assembly into the pipe. I placed the top cap and added a bead of Goop around the seam and the hole the wires pass through. This allows easy disassembly of the units while still providing a good seal. Connection to these units is made with the water-resistant connectors developed for phone lines. These connectors contain a small amount of grease to exclude moisture.

Even though the sensors connect to the back of the case via eight conductor cables, I split the cables to individual runs as soon as possible to avoid cross talk on the lines (see Photo 5). The connectors made interfacing to the processor box easy, but you're still tempting fate by running signal lines in this cable with pairs twisted together. I haven't encountered problems with short cable runs. Try to run sensor cables away from sources of noise.

I constructed the prototype using point-to-point wiring on a prototyping board (see Photos 6 and 7). The PLCC socket has many pins close together, so be careful to avoid shorts. I managed more than once to lose count while I soldered wires to this socket and had to move a lot of connections. Fortunately, the result is a reliable unit.

I use connectors between the main board and off-board elements. This takes longer, but makes a more attractive and easier to maintain unit. Be careful assembling these connectors and solder them rather than crimp.

The processor board doesn't use much power, but the fluorescent display is power hungry. I have a power brick that supplies 5 V, ±12 V to the unit. The processor case gets warm after extended runs, but not hot enough to worry about part reliability.

I have an interesting story about the brightness of this display. I have a magnifying desk lamp above my workbench. One day, I worked during the afternoon and left the unit running. Later in the evening, I noticed the display was being magnified by the desk lamp and projected on the ceiling in about 1" characters. Focus could be varied by just moving the lamp closer or further from the unit. There may be an opportunity in here somewhere.

OPERATION

I planted only five of the sensors so far. One of the sensors was under the brick walkway leading to the front door. Note that I have a two-story house without gutters. When a rainstorm hit, all of a sudden that sensor went wild. It turns out that the water from the roof runs off in a strong stream and hits the ground about 6' away from the sensor. Consequently, the sensor went off constantly until I raised the sensitivity of the unit.

NEXT

With an outdoor project such as this one, there is limited time when I can make good progress. I ran the outdoor lines last fall prior to the ground freezing. When the ground thawed, I connected the first set of sensors I constructed during the winter. The software seems to be operating well. The one area that requires additional work is the sensors; I want to make them less sensitive to noise. That may be this winter's project.

Dave Penrose is a management and engineering consultant in Bedford, New Hampshire. He has a Master's in Computer Science and holds an Extra Class Amateur radio license (AA5QL). His experience includes more than 30 years with a major aerospace corporation and more recently as vice president of systems and technology for a high-tech start-up company. You may reach him at penrose@mediaone.net.

SOURCES

80C451
Royal Philips Electronics
(212) 536-0500
Fax: (212) 536-0559
www.philips.com

SmartWatch
Dallas Semiconductor, Inc.
(972) 371-4000
Fax: (972) 371-3715
www.dalsemi.com

Geophone sensors
Geosource, Inc.
(703) 478-6893
Fax: (703) 777-9830
www.geosource.com

FEATURE ARTICLE

Paul Verhage

A Class Act

Making Your Own PCBs

School's in session as Paul shows us what it takes to design and etch your own printed circuit boards. If you're looking for a way to avoid the expense of using a custom board house, the "fab lab" approach may be just the solution for you.

The best way to build an electronics project is to solder the electronic components to a printed circuit board (PCB). After bing soldered, the completed circuit will be stable and the component connections less likely to break. Before soldering, you must either purchase or make your own PCB. A mass-produced PCB, or board, is inexpensive, but limited to the circuits that other people have designed and etched. If you want a unique circuit, you'll have to design and etch your own.

Board houses will help you with this endeavor, but they usually require a minimum order of $100. For a single PCB, $100 is not reasonable. It's more expensive if you make a design error.

To get around this cost, many people use a photocopier to make transferable toner resist masks. With this method, you copy the PCB pattern onto a transparency using either a laser printer or photocopier. A hot iron transfers the toner from the transparency to a clean copper clad board.

Afterward, you can etch the board with ferric chloride. The layer of toner prevents the copper from being etched away. My results with toner masks were not as satisfying as having a professionally manufactured PCB. Today though, I make my own high-quality PCBs at an affordable cost to facilitate my hobby of near space exploration.

With the lab I developed, I can go from concept to completed project during one weekend. And the best part is that a single-sided, 4″ × 6″, etched and drilled PCB costs me less than $4. The process is clean and simple. With a little care and practice, I can make double-sided boards, too.

The cost of building my PCB fabrication lab was about $300 excluding the PC and laser printer. I have no other costs except for purchasing inexpensive, sensitized copper clad boards, laser printer transparency sheets, and two inexpensive chemicals (developer and etchant). Having a PCB fab lab is like having a photographic darkroom. In this article, I'll tell you how to make and use a similar lab.

THE PARTS

You'll purchase most of the parts ready to use. Only the UV table and etcher will be built or modified from locally available parts. The items you'll need are listed in Table 1. Ares Lite is a Windows 95-based PCB CAD program. You'll use it to lay out the parts, traces, and ground planes (highly recommended) of your PCB.

Ares Lite prints UV-ready PCB masks on transparency sheets with your laser printer. The homemade UV table exposes sensitized copper clad boards to your PCB mask (see Photo 1). Constructing it will be an afternoon project. Because you don't want to shake a tray of etchant for 30 min., modify the cheapest foot massager you can find to do it.

To drill holes into the etched PCB, use a Dremel hand tool and drill press (unless you find a less expensive drill press). Carbide drill bits drill lead holes into the PCB. Don't use steel drill bits; the carbide bits cut a cleaner hole and stay sharp longer.

You need two inexpensive plastic photographic trays to develop and etch the exposed copper clad board. To keep the etching process clean, use freezer bags when etching the copper clad. Use GLAD-LOCK zipper bags.

Ares Lite PCB design software	$65
UV table (homemade)	$70
Dremel hand tool	$55
Dremel drill press	$50
No. 67 carbide drill bit	$4
Two photographic developing trays	$10
PCB etcher (a modified foot massager)	$25
Bug light and clamp-on reflector	$15
Two measuring cups and spoons	$6
One package, five sensitized 4″ × 6″ single-sided, copper clad (Kepro, DF1-465M)	$16
2 lbs of developer (Kepro, DFD-12G)	$14
4 lbs of etchant (Kepro, SP-1G)	$21
One package of one-quart freezer bags (Glad Bags)	$4
One box of 50 laser printer transparency sheets (Staples, Apollo CG7060)	$10

Table 1—*Here is everything you need. The consumable products, which determine your cost for making PCBs, are included.*

Only open the bag of sensitized copper clad in the subdued light of a yellow bug light. Because I don't want to leave a bug light in the socket in my bathroom (my current PCB fab lab), I use a portable light fixture. To measure the proper amount of water and chemicals, stop by the kitchen section of your local department store and get a teaspoon and tablespoon. Also get one 1/2 cup and 1/3 cup measuring cups or a beaker.

Now that I've introduced you to the required parts, let's start building the UV table. Later, you'll make a foot massager into the etcher. While building the UV table and etcher, purchase your chemicals and sensitized boards from Kepro. The most time-consuming part of making a lab is learning how to use Ares Lite. However, it's not a difficult program.

THE UV TABLE

Building the UV table is straightforward and will be accomplished using hand tools. According to Kepro, UV light sources that work with its boards include UV fluorescent tubes, photoflood bulbs (I assume these are the same as tanning bulbs), or the sun. I wasn't pleased with tanning bulbs and I want to keep the light source consistent, so I don't use sunlight. Instead, I use two fluorescent UV lamps. These lamps are white and do not have black coating like black lights. (You know you have the correct lamps if they burn your retinas when you stare at them!)

Next, pick up two 18″ fluorescent lamp kits (available at stores like Wal-Mart). These lamp kits are used as extension lighting and are not permanently wired into household wiring. I use the Under Cabinet Light Fixture from Lights of America. While at the store, pick up Stic Mount Cord Clamps, the kind used for tacking down extension cords or outdoor Christmas lights. Get two metal hinges (2.5″ wide), a bag of drywall screws (1 5/8″ long), a packet of four nail-in feet, a 3″ mending brace, white paint, and a paintbrush. Pick up the cheapest six-outlet power strip you can find. Finally, buy a yard of 1/4″ foam rubber and a plastic pull handle. These miscellaneous parts should cost less than $30.

At a glass store, purchase a 1/8″ thick sheet of glass, cut to 19.5″ by 9.75″. I assume all window glass is sufficiently transparent to UV.

To make the UV table, pick up a piece of 1″ × 4″ pine, 8′ long. Have the board cut into four pieces; two should be 20.5″ long and the other two should be 10″ long. And, purchase a sheet of 3/4″ plywood and have two pieces cut out of it that measure 20.5″ × 11″. You'll probably have to purchase a 4′ × 4′ sheet of plywood.

The last pieces of wood you need to buy are for mounting the glass into the table. At a hobby store, find basswood strips that measure 1/2″ × 1/8″. These come in 24″ long strips, so purchase three of them. Also buy a 3′ × 1′ sheet of 0.010″ thick polystyrene plastic.

BUILDING THE UV TABLE

Now it's time to build the UV table. Use wood glue and drywall screws to attach the 1″ × 4″ pine boards to one piece of the 3/4″ plywood to form a box (without a lid). The box measures 20.5″ × 11″ and is 4.25″ tall. The plywood is the bottom of the UV table and the pine boards are its sides. On the 3/4″ wide edges of the frame (at the top), glue the basswood strips. Position them on the outside edge of the frame to create a recess for holding the glass. This recess also keeps the sharp edges of the glass hidden from fingers that may brush against them.

Using an Exacto saw, cut a small notch into one side of the box that's large enough to pass the fluorescent lamp cords through. You may be tempted to drill a hole in the side of the frame, however, a hole large enough to pass the plugs of the lamp cords through would be about 3/4″ in diameter. A hole that large would leak UV and may be a hazard to eyes. Give the box and the remaining piece of plywood a thick coat of white paint. I assume the white paint will reflect UV the best. Finish by nailing feet onto the bottom of the box.

Now it's time to attach the lid. Align the lid on top of the box. While securely holding it in place, screw in the hinges on the long side of the box. Be careful and make sure the box will open after you screw on the hinges.

Open the box and mount the fluorescent lamp fixtures inside the table, centered on the bottom. There's a drilling guide on the lamp package for determining where to drill the holes. Use the Stic Mount Cord Clamps to mount the lamp cords securely to the inside bottom of the table so they can't pop up and cast shadows on your PCB.

Mount the power strip on the outside of the table. You want the power strip on the side with the notch for the power cords. Use a few of the drywall screws to hold the power strip to the side of the UV table. Be sure to mount the power strip so it's flush with one edge of the UV table's side.

To keep the power strip from sliding off its mounting screws, make a stop with the mending brace. Screw the brace to a corner of the table on the side that the power strip wants to slide off. Bolt the brace flush against the power strip.

With the mending brace flush against the power strip, the power strip can't slide off its mounting bolts. If you ever need to remove the power strip, you'll need to remove the mending brace first.

It's time to get back to the electrical work. Plug the lamp cords into the power strip. Using the power strip, you'll have a single switch to turn on/off both UV lamps. At this point, turn on the lamps and the power strip. When you know they work properly, switch off only the power strip (after you glue in the glass, you won't be able to reach the lamps). Next, remove the lamp covers and fluorescent tubes. Replace the original fluorescent tubes with the UV lamps, but don't replace the lamp covers.

Mount the pull handle to the center of the lid using the screws that came with the handle. Be sure to screw the handle tightly to the lid. Before proceeding, thoroughly clean the glass, getting rid of any dirt or fingerprints. Place the glass into the recess at the top of the box and use a little silicone glue in the corners to hold it down.

While the silicone glue is setting, cut a piece of 1/4" foam rubber the same size as the lid. Glue the 1/4" foam rubber to the inside of the lid using silicone glue. Finish the inside of the lid by gluing the 0.010" thick piece of styrene plastic over the face of the thin foam rubber. The plastic sheet and foam rubber create even pressure on the copper clad board and resist mask.

After the glass is in place, use silicone glue and attach a steel ruler or square to the top of the glass. The ruler will align the copper clad with the transparency. A metal square can be used to align doubled-sided boards.

Your UV table is now complete. The total cost should be around $70.

MAKING THE ETCHER

I started etching boards by agitating trays by hand. This made me realize that I had better things to do with my time than to shake trays for 30 min. So, I purchased a foot massager on sale that could agitate boards for me. At first, I tried placing the etchant and board into a sealed plastic freezer bag and dropping it into the foot massager filled with warm water. This resulted in unevenly etched boards.

Currently, I use a thin plywood platform built on top of the foot massager to shake a photographic tray filled with warm water, a freezer bag containing etchant, and an exposed copper clad board (see Photo 2). Eventually I want to find a thin box that I can set inside the massager when filled with warm water. Inside the box would be etchant and the board(s) standing vertically on edge. The water surrounding the thin box would keep the etchant warm, speeding up the etching process.

Let's get back to my current design. The etcher's wooden platform (table) is made from 1/8" airplane plywood purchased at the hobby shop. I glued a basswood strip to the bottom of one side of the plywood to keep it level when it is placed on top of the foot massager. I recommend using small metal screws to attach the wooden table to the top rim of the massager, although I use a set of dowels.

Glue down 1/4" basswood strips to form a rectangular barrier that holds the photographic tray. This allows you to set the etching tray on the table without permanently attaching the tray to the table. This barrier is high enough to stop the tray from bouncing all over the place. The two nice features of this etcher are that it's cheap and etches PCBs unsupervised.

SUPPLIES FOR ETCHING A PCB

To develop and etch exposed copper clad boards, pick up two inexpensive plastic photographic trays (one for developing and the other for etching), a plastic Tupperware juice container (for storing the dry etchant powder), and a set of measuring cups and spoons. Use a bug light to see what you are doing while the sensitized boards are outside their lightproof bag. (A photographic supply store may have a portable light fixture that you can use instead of the bug light.)

Also, buy a cellulose sponge to finish your boards. Cut the sponge into 1" cubes. I don't recommend sponges containing germicide, because I don't know how the resist will chemically respond to the germicide.

Purchase your PCB supplies from Kepro. To save money with experimental designs, you can use a 1/16"-thick, single-sided, 1-ounce, copper clad, flame-retardant, polyester board. They come in packets of five for $16.50. Each board has a layer of sensitized resist and is covered with a clear plastic sheet. This sheet is transparent to UV and protects the resist from being scratched while handling. Kepro also sells epoxy composite and glass epoxy-based boards. Polyester boards work well for my projects, but I would use a glass epoxy board for a PCB that I planned to sell.

Now that you have purchased all your equipment, constructed the lab, and learned to use Ares Lite, you're ready to start making your first printed circuit board. There is an 11-step process for fabricating a PCB. First, design the board in software, double-checking your design. Second, print an inverted-color copy of the artwork onto a transparency sheet with a laser printer. Then, trim and tape the transparency to the glass in the UV table.

Fourth, place the board copper side down on the transparency and expose it to UV for 60 s. For the next step, let the board's resist harden for 15 min. After that, develop the board for 60 s. Seventh, place the developed board into a freezer bag filled with etchant and seal the bag.

The next step is to agitate the board for about 30 min. Ninth, clean the board and let it dry. Next, drill holes in the board. And finally, the eleventh step is trimming the PCB.

Don't let these 11 steps scare you away from making a PCB; they're not difficult. Now, let's look at these steps in greater detail.

ARES LITE

I use Ares Lite as my PCB CAD program. This inexpensive Windows 95 program is easy to use. It has related software that does schematic captures and circuit emulations just like PSpice. But, I have been able to design my board without schematic capture, autorouting, or testing.

Ares Lite comes with a large number of component pinouts. You can develop any that are missing from the database. A background grid exists in the workspace allowing you to snap down components at 50-mils intervals. The program lays down copper traces with widths from 5 to 1000 mils. After

laying down the copper traces, you can create ground planes to fill in the open spaces on the board.

In most circuit board designs, ground planes are recommended. I use ground planes to reduce the amount of copper that needs to be etched from the board. Ares Lite allows you to design double-sided PCBs (the top copper is displayed in red and the bottom copper is blue). The software allows you to designate 14 additional layers, four mechanical layers, and a component side.



Photo 1—*Check out the UV table with mask. You can see the UV lamps beneath the glass.*

CREATING THE RESIST MASK

Start designing your PCB by loading the components from the database. At anytime during board development, you can load additional components. Change the orientation of loaded components by rotating or flipping them. After properly oriented, drag the component to the workspace and place it anywhere. At anytime while laying out the board, you can duplicate or drag components to a new location. You can even rotate components in the workspace without deleting them.

Ares Lite lets you lay down your board's copper traces whenever you wish. To lay down traces, first select the trace tool, then the side of the board you want the trace to be on (top or bottom copper), and then the width of the trace. On the workspace, make a series of clicks to define the path of the trace. And then to end the trace, right click the mouse.

While laying out components or the traces, you can change pad sizes if necessary. After components and traces are in place, create ground planes by filling open spaces with copper. The software allows you to create polygonal-shaped ground planes. Before finishing, add hardware mounting holes to the board.

You should finish by doing something that most techies don't do, document your work! There is a text feature in Ares Lite that enables you to draw rectangles, polygons, circles, and write text. This is especially handy when you go back to that board you were designing last week.

When finished designing your product, print the resist mask and any notes regarding component placement. To print the resist mask, select Output, then select Print. Select only the copper layer you want to print (in most cases, that's the bottom layer). Be sure to select the option to invert the colors. Usually I print a paper copy review. As long as you don't scratch the final transparency copy, you can use it to expose unlimited copper clads.

Next, print a copy of the component layer and your notes on paper. Don't select a copper layer when print-

ing notes. Also remember to deselect the box for reversing colors.

Carefully cut the transparency so it fits inside the UV table. Make sure the toner side is up by looking from a glancing angle. If the toner side is up, the face of the transparency will be dull over the black areas. If it's shiny, you have it upside down.

Use masking tape to hold the transparency on the glass. Use as little masking tape as possible, because it will act as a mask increasing the amount of copper that gets etched. Don't use transparent tape because it is too difficult to remove from the glass and mask. I usually trim and tape the resist mask up against the steel ruler. When taping down the transparency mask, avoid touching the glass, because you don't want to leave fingerprints where you may be exposing the board. Occasionally give the glass a light brushing to remove dust.



Photo 2—*The developed PCB is in a freezer bag filled with etchant. The bag is in a tray of warm water. The foot massager will be turned on next.*

EXPOSING THE BOARD

The lights must be off and the yellow bug light turned on. When you take out a board from the copper clad package, tape the bag closed to protect the other boards from light. Turn the board copper side down and place it over the mask. Do not play with the plastic covering on the copper side of the board because you risk leaving wrinkles in the plastic protector that can show up on the copper traces.

Place the board so that no traces on the mask are left uncovered. You don't want to shoot only half of the board. You can tape the copper clad down, but I haven't found this necessary. After positioning the board over the mask, close the lid of the UV table. The lid protects your eyes from the UV lamp, so it's important to close the lid every time you expose a board. Don't shake the copper clad around when closing the lid. I like to place a little pressure on the lid to make sure the copper clad is flat against the mask and glass.

Next, turn on the UV lamps by using the power strip switch. After the lamps start, wait 60 s for the proper exposure and then shut off the lamps. You've now exposed your soon-to-be-completed PCB.

After exposing the board to UV, let the resist harden for 15 min. inside the UV table (lamps off). If you need to shoot more boards, put the exposed board into a lightproof bag and shoot the next board on the UV table.

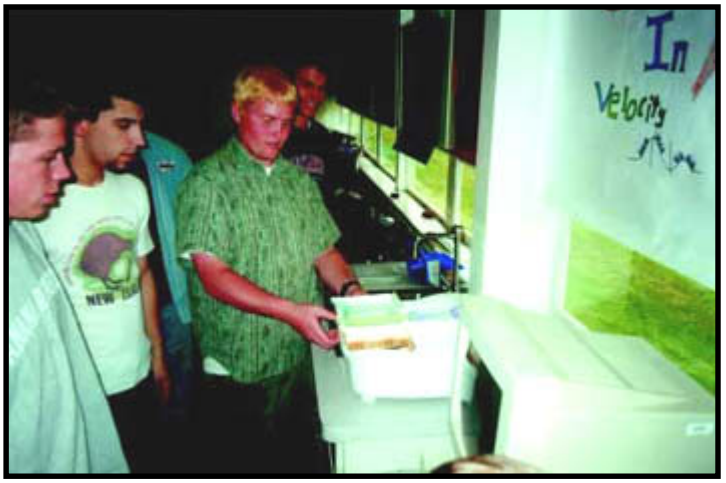


Photo 3—*At Nampa High School, Idaho, my electronics students design a PCB for a 555 IC LED flasher. Here, Adam Rainwater starts the etcher with fellow students (from left) Josh Roach, Bennett Yankey, and Steve Hyer.*

DEVELOPING EXPOSED BOARDS

Two of the goals while making a PCB are to keep the cost of consumable products low and keep the process clean. The method I use will let you shoot single boards without mess or waste.

While the exposed resist hardens, get the developer and etchant ready. The Kepro developer is just washing soda. It comes as a dry powder in a 2-lbs plastic bottle. When mixed, it will develop 120″^{2} of board (720 4″ × 6″ single-sided boards). To mix only enough developer for one board, pour 1 tsp (5 ml) of dry developer into a photographic tray. Do not add water yet.

To mix just the amount of etchant needed for a 4″ × 6″ single-sided board, pour 1 tsp plus 1 tbsp (20 ml total) of etchant into a Glad freezer bag. I use sodium persulfate (SP) as my etchant. I haven't had to use the accelerator that comes with the SP etchant. Don't use the developer or etchant to complete a second board, use fresh chemicals for each board.

After the resist hardens, pour 2/3 cups (150 ml) of hot water into the developer tray and dissolve the washing soda. Still with the lights off and the yellow bug light on, remove the exposed board from the UV table. Look at the exposed surface of the board and you'll see the faint traces of the latent image on the resist. After developing, the image will have much more contrast.

Peel off the plastic protective cover from the copper side and drop the board into the photographic tray filled with developer (keep the board face up). Spend 1 min. agitating the tray. The resist layer will turn purple underneath the bug light (go K-State!). After 1 min. of shaking, lift the board on its edge while it's still over the tray of developer. Gently wipe the copper surface with a small sponge dipped in developer. The copper to be etched

will come clean, leaving bright copper that contrasts with the purple dyed resist. Notice the resist has a slick feeling to it whereas the copper doesn't. Once, the resist completely dissolved off the board. I imagine this happened because I left the board in the developer too long.

It will be obvious when the copper traces are clear of resist, but you may lose a board learning this. The critical part of making a PCB is the development process; every other step has a little slop built into it. After developing the board, rinse it in warm water to stop the developing. Do not throw out the tray of developer yet, because you'll use it to remove the remaining resist after etching the board.

If you find any defects in the resist, let the board dry before repairing it. Use a hair dryer or Chem Wipes to speed up the drying, but don't use tissue paper because they will leave lint on the damp board. The boards usually come out of the developer fine. However, if you need to repair a notched trace, use a permanent fine-tip marker. Lightly tap the pen onto the trace and leave a spot of permanent ink. You should do this several times, but let the ink dry between markings.

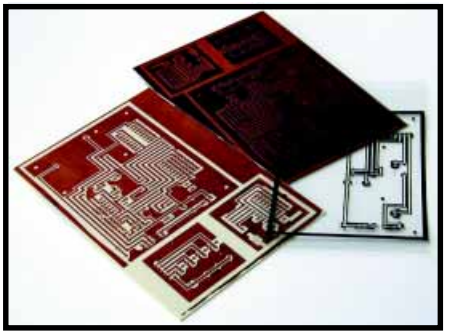


Photo 4—*From left to right, here's an etched PCB, PCB with developed resist before etching, and a mask on a transparency.*

ETCHING THE BOARD

Etch the board soon after developing it. In fact, if you're happy with the traces, you can start etching as soon as you rinse off the developer. Being a cautious fellow, I still leave the lights off while beginning this process. The etchant is a clear liquid that turns an attractive blue when etching copper. This etchant is not nearly as foul nor messy as ferric chloride. Kepro sells the etchant in a powder, which will store longer than when in solution. Two and a half pounds (the smallest amount Kepro sells) of dry etchant powder makes 1 gal of etchant. One gallon is enough for dozens of boards. I transfer the dry etchant into a Tupperware container and store it in a location safe from my house rabbits. You should do the same.

Pour 1/3 to 1/2 cup (100 ml) of hot water into the freezer bag containing the etchant. Knead the powder in the bag to dissolve it. After dissolving the etchant, place the developed board into the freezer bag and seal it. Double-check the seal. Then, set the freezer bag aside and pour hot water into the etching tray.

Place the freezer bag in the tray with the copper side of the board facing down. Wrap most of the excess bag around the board, leaving the excess bag on top. After placing the tray on top of the etcher, turn on the foot massager (see Photo 3). I like to set the

foot massager on a towel so it can't move around and to collect water drops. It takes less than 30 min. to etch a 4″ × 6″ board with SP. About halfway through the etching, change the water in the tray, replacing it with hot water.

You can etch two boards simultaneously. Place each board in its own bag filled with etchant and drop one on top of the other in the tray. Midway through etching, when you change the water, switch the position of the bags. I haven't tried a third board, but it would probably work with a deeper tray. After you start etching, you can turn on the room lights.

After etching the board, remove it from the freezer bag and thoroughly rinse it. Place the etched board back into the developer tray (the developer will be cold) and let it sit for a few minutes. You'll start to see the purple resist curdle and peel off the board. One minute later, you can use a sponge to finish wiping off the resist. You now have a clean and etched PCB (see Photo 4). Rinse the board and prop it up on its side to dry.

AFTER ETCHING

Drill your PCB with a carbide drill bit chucked into a Dremel mounted to a small Dremel drill press. You may want to tape a piece of paper over the base of the Dremel drill stand so that the PCB surface stays clean. I typically use a no. 67 drill bit, but it's too small for 0.025″ male headers. I experimented with miniature steel drill bits purchased from the local hardware store and discovered that they leave a pile of melted dust. I also discovered that carbide drill bits in surplus magazines do well. The bits I purchase from Kepro are only $4 a piece, so I usually use those.

The problem with carbide drill bits is that they are brittle. Although they remain sharp for a long time, they cannot tolerate sideways force; a jiggle while drilling is enough to snap them. With care, you will be able to drill hundreds (possibly thousands) of holes with carbides, but always keep extra bits on hand.

The next step is to cut and trim your PCB. Seldom will your design be exactly 4″ × 6″, so you'll have to cut off the excess board. Use a razor cutter mounted to a handle to trim the PCB. Using a straight edge, score the PCB by cutting it several times with the razor. After scoring the board, carefully snap the board along the score. Use sandpaper to smooth and round the edge. I lay sandpaper on a flat surface and run the board edge across it.

Some people tin their boards, but it's unnecessary. The Kepro boards are clean and I've easily soldered components to them. If you're going to store the board before soldering components, you may want to tin the board first, because the copper will oxidize over time.

Tinning compounds are available in a dry powder. To use them, mix the tinning powder with water and soak the board. Some of the tin in the solution will replace some of the copper on the board. After the tinning process, remove the board from the solution and let it dry.

THE FINAL PRODUCT

After you finish drilling and trimming (and possibly tinning) your PCB, it's time to solder the components. Now, use your printout of the component layer and notes. Chances are you'll never remember the placement or values of components, so your notes should be complete. I cannot stress enough the importance of complete documentation.

It also pays to get a good solder station and tools. Your boards will be excellent quality and deserve clean and trim solder joints. Although soldering is outside the scope of this article, I'd like to make a few recommendations. Always wipe the solder tip on a damp sponge and immediately tin it. It's a good idea to get yourself a solder sucker to clean up excess solder or to remove solder from pads. I use a set of sharp cutters to clean up my solder joints and exposed wires. Be careful, I have managed to cut a copper trace off a board with these cutters.

And, get a lead bender and use it regularly. It will bend resistor leads to the correct length the first time, saving strain on the leads. I use masking tape to hold components in place when soldering on the other side of the board. Solder two diagonal pins of an IC sockets first. Then, press on the socket as you reheat the two soldered pins. This allows you to seat the socket completely before soldering the remaining pins.

I hope you'll make your own PCB fab lab. It's a great hobby that saves money while allowing you to create a quality PCB. It's difficult to overemphasize the quality of these PCBs. It's almost automatic to end up with a board as good as those you purchase from a board house. The only thing lacking is plated-through holes. I'll bet Kepro has those for sale.

Author's Note: I thank Steve Kelly of Kansas State University who inspired this project. Steve introduced me to Kepro and how to design and make top-notch PCBs. Also, I'm interested in making a training video based on this article. Please contact me if you're in the Boise area and can help make the video.

Paul Verhage is a science and electronics teacher at Nampa High School in Nampa, Idaho. His interests include astronomy, meteorology, digital electronics, and outer space. To facilitate his hobbies, Paul designs his own PCBs. Recent PCB designs include a Lego robot controller, a digital display for a hat, and a controller for nearcraft (a high altitude, model satellite). He can be contacted at pverhage@sd131.k12.id.us

SOURCES

PCB boards, chemicals, and supplies
Kepro Circuit Systems, Inc.
(314) 426-2600
(800) 325-3878
www.kepro.com

Ares Lite
Labcenter of Electronics
+44 (0) 1756 753440
Fax: +44 (0) 1756 752857
www.labcenter.co.uk

Handtool and drill press
Dremel
(800) 437-3635
(262) 554-1390
www.dremel.com

FEATURE ARTICLE

George Novacek

Designing for Reliability, Maintainability, and Safety

Part 2: Digging Deeper

Having covered the consequences of not making your design safe and reliable, George is ready to get up to his neck in the details of the hot tub controller application. Relax, turn up the jets, and get ready to toast the success of your next design.

In Part 1 of this series, I talked you through designing a simple hot tub controller. You calculated its predicted reliability and discovered that it satisfies the MTBF design criteria. Reliability was improved by moving the controller to where the ambient temperature excursions combine with components' heat sinking, resulting in lower junction temperature than originally estimated.

Now that you have a controller that performs the desired function, it's time to satisfy the safety requirements. This is not as easy as it seems. I've stated many times that achieving the product's desired functionality is a fraction of the design effort. More effort is expended to make the design safe. So, let's discuss the details.

BEING PREPARED

Failure mode and effects analysis (FMEA) is a bottom-up review of a system. In this analysis, you examine components for their failure modes, notice how the failures propagate through the system, and study their effects on the system's behavior. This leads to design review and possibly changes to eliminate weaknesses.

By adding the criticality column in the FMEA work sheet, the analysis becomes FMECA (failure modes, effects, and criticality analysis). In most systems, it is not necessary to examine every component. You can rearrange the design into functional blocks and, when needed, consider individual component failures within functional blocks that may be critical. Take a look at Figure 1. This is the circuit of Figure 3 of Part 1 broken into four functional blocks, A, B, C, and D.

The work sheet shown in Table 1 is a standard format that engineers often tailor to fit their specific requirements. This matrix is simplified, limited only to issues you need to consider. The first column identifies the failure. For a more complicated system, you would have a separate database of the failures with reference pointers to the work sheet. The letter identifies the functional block, the number, and the individual failure of the block.

The next three columns are self-explanatory. The method of detection includes built-in test capability and status reporting. Your simple, hypothetical controller has some, but as I'll explain, every fault must be detected, therefore the design needs to be modified accordingly.

There are only two criticality levels, high and low. High criticality failure causes the heater to stay on to heat the water above 102°F; a noncritical failure causes loss of heating, and consequently, the use of the system is lost.

The probability column will assign a probability number to the fault taken from the reliability prediction in Table 2. To accomplish that, simply identify the components in the functional block, add their respective λ_p, and multiply by 10^6.

Observation is the only detection method of malfunction. This isn't acceptable for critical failure, when the water temperature exceeds the maximum limit and must be provided by the built-in test (BIT) function.

What do the FMECA results show you? They indicate that satisfying the 10^{-5} system availability will not be a

problem. The reliability prediction has already shown that. But, the FMECA brought several important facts concerning the design to the surface.

One fact is that failure A2 needs to be watched carefully (don't change the design until you finish full analysis). Failure of the power supply, just a cold joint of the grounding pin of U1, will likely damage the controller and could cause critical water overheating.

A3 means the power supply puts out less VDC than expected. It could be a half wave rectified AC. You have no idea how the controller will react to this. You could perform more analyses, going from block to component level, analyzing failure modes and effects of every component, and then try to improve the reliability of the components potentially responsible for critical failures. However, as the probability number shows, you are almost three orders of magnitude away from satisfying the critical performance (10^{-9} is required for water overheating). Therefore, a more drastic measure, other than beefing up components' specs, is needed.

B1 and B2 show that there is a two orders of magnitude deficit in satisfying the critical requirement. The microcontroller isn't the problem. Software is a potential culprit. Assume the software has been properly verified and validated and its reliability is not an issue. But, even 100% correct software can go on a tangent because of external effects. Therefore, the software probability of failure is pegged at $<10^{-10}$, which is normal.

Defects in the temperature sensor, block C, must be detected by the microcontroller running a plausibility check on the values. Two checks can be performed here: the value must be within a plausible range and the rate of change must not be greater than expected from the system. Your system will be fail passive, meaning that if the microcontroller detects invalid data, heating will shut down. The mechanical design must make sure thermistor R3 is exposed to the water temperature at all times.

I won't dwell on nonelectrical issues. Other than the mechanical influence, there's no defined failure mode where a thermistor value would remain electrically correct but fail to modify its resistance according to its temperature.

Block D is monitored for the solenoid valve (SV) current through R6. This allows detection and protection from short and open circuits. However, Q1 is a critical component. If it fails by shorting SV to ground, a critical fault will result. A similar situation exists for transzorb D5, SV, and SV's wiring (more about this later). D5 is not stressed unless there is a transient, and therefore, its effect can be adjusted by a duty cycle.

I'll give you one last tip. It's advantageous to have an indicator to announce the controller failure. Moving on, for the last step of the design evaluation, you'll perform a fault tree analysis (FTA).

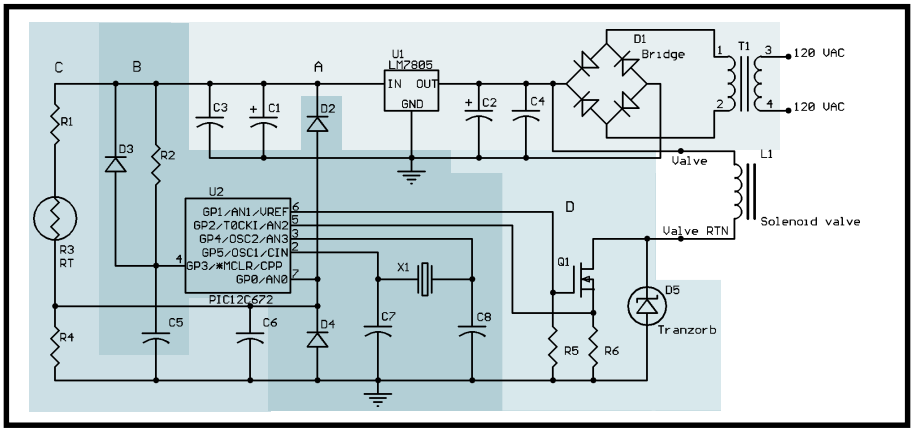


Figure 1—*To perform FMECA, the diagram is divided into functional blocks. It is a bottom-up review of the design. Consider functional failures and examine how they propagate to the system level. Generally, functional blocks give sufficient detail, but check out individual components only if there is a critical failure.*

FAULT TREE ANALYSIS

In many respects, the FTA and FMECA could be used interchangeably, because they are different representations of the same data. The difference is that the FMECA is a bottom-up and the FTA is a top-down graphical analysis. The FTA starts with the top event you're interested in, then builds the fault tree using Boolean logic and symbols. By adding known failure probabilities, the same used when creating the FMECA, you arrive at the probability of the event of interest. As with the FMECA, the analysis can be performed on the functional block as well as at the component level. Using Boolean logic, probabilities fed into an OR gate will be mathematically added, while the ones fed into an AND gate will be multiplied:

$$P_{OR} = P_1 + P_2 \ldots + P_N$$
$$P_{AND} = P_1 \times P_2 \ldots \times P_N$$

The top event you are interested in is the uncontrolled heating of the water. Because there is only an OR gate in the FTA, any one event in the circle can cause the top event. Having calculated the failure rate for the uncontrolled heating as $\lambda = 3.524 \times 10^{-6}$, you can calculate the probability of this failure occurring:

$$P_F = 1 - e^{-\lambda \times t}$$

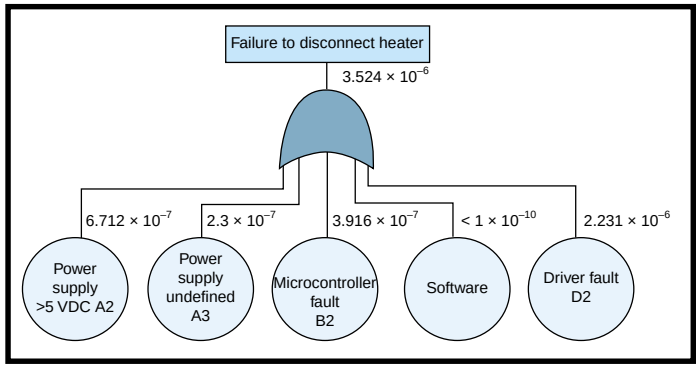


Figure 2—*The fault tree analysis supplements FMECA. This is a top-down view of the system. You identify critical failures and consider which causes will contribute to them.*

System: hot tub controller				Document number			Revision
Function: water temperature control				Environment: ground fixed			Date
Operation phase: all				Prepared			Checked
Failure no.	Failure mode	Possible cause	Failure effects	Method of detection	Criticality	Probability λ/h	Remarks
A1	Output = 0 V	Can be caused by a failure of any component within functional block A or an external short	Loss of water heating	Observation	Low	7.844×10^{-7}	
A2	Output > 5 V	Failure of T1 or U1	Potential damage to U2, unpredictable effects. Maybe loss of or continuous heating	Observation	High	6.712×10^{-7}	
A3	Output out of tolerance	C1, C2, C3, C4, D1, U1	High ripple or out-of-spec operating voltage; unpredictable.	Observation	High	2.3×10^{-7}	Can be eliminated by monitoring the power supply health and forcing reset if outside limits.
B1	Output continuously 0	U2, C5, C7, C8, R2, D3, software	Loss of water heating	Observation	Low	3.9×10^{-7}	
B2	Output continuously 1	U2, C5, C7, C8, R2, D3, software	Continuous heating	Observation	High	3.9×10^{-7}	This means the Microcontroller block is not working. Its output could be stuck in either state.
C1	Temperature sensing not working	R1, R3, R4; Any device open or short circuit	Loss of water heating	Input signal plausibility check by microcontroller observation	Low	2.296×10^{-7}	Resistor network is designed such that a short or open of any device takes the signal out of plausible range.
C2	Temperature sensing not working	Thermal link between water and R3 lost	Continuous heating	Observation	High	Undefined	Mechanical design issue
D1	No SV drive	Q1, R5, R6	Loss of water heating	Microcontroller monitors Q1current; observation	Low	2.304×0^{-6}	
D1	Continuous SV drive	Q1, D5	Continuous heating	Microcontroller monitors Q1current; observation	High	2.231×10^{-6}	Can be detected but not remedied by the system

Table 1—*The analysis data is organized in the FMECA work sheet, which makes it easy to review assumptions and conclusions.*

For t = 10 years, that's 87,600 hours of operation.

$$P_F = 1 - e^{-3.5424 \times 10^{-6} \times 10 \times 365 \times 24} = 0.266$$

Or, for P_F = 0.5 (50% chance of uncontrolled heating), it takes 22 years of operation. But that's not good enough for a system that can potentially cause injury. Using the equation above, calculate $\lambda = 1 \times 10^{-9}$, which is for the specification requirement. This would give even odds for the uncommanded heating after 79,000 years.

WHAT'S NEXT?

For the uncommanded heating, you are nearly three orders of magnitude removed from the specification goal of $\lambda = 10^{-9}$. It's unrealistic to come close to this goal by improving the components' reliability. But, what if you could feed the top event in Figure 2 into an AND gate? ANDing it with another signal of merely 2.8×10^{-4} probability of failure would do the trick (see Figure 3).

This is how high safety and reliability is achieved in systems by redundancy. You have to sacrifice the overall MTBF as you add components, but critical functions will perform better. The simplest approach, it might seem, would be to add a mechanical thermostat in series with Q1 to open the circuit at 102°F. However, every fault that could cause a critical failure must be either prevented from happening or detected. Adding a function that may or may not be available does not solve the problem.

The thermostat in the SV path doesn't solve the problem. Its failure can't be detected, meaning it has a dormant failure. As long as the electronic controller works properly, the thermostat could be defective yet you would never know. Conversely, the thermostat could be controlling the hot tub while the electronic controller is dead.

The most common solution is to double the processing channels and revert to a safe state, in this case the

Component	Description	$\lambda_p/10^6$ h	MTTF
R1	Resistor	1.0794×10^{-2}	9.2647×10^{7}
R2	Resistor	1.0794×10^{-2}	9.2647×10^{7}
R3	Thermistor	3.8760×10^{-3}	2.5800×10^{8}
R4	Resistor	1.0794×10^{-2}	9.2647×10^{7}
R5	Resistor	1.0794×10^{-2}	9.2647×10^{7}
R6	Resistor	6.3492×10^{-2}	1.5750×10^{7}
R7	Resistor	1.0794×10^{-2}	9.2647×10^{7}
R8	Resistor	1.0794×10^{-2}	9.2647×10^{7}
R9	Resistor	1.0794×10^{-2}	9.2647×10^{7}
R10	Resistor	1.0794×10^{-2}	9.2647×10^{7}
R11	Resistor	1.0794×10^{-2}	9.2647×10^{7}
R12	Resistor	1.0794×10^{-2}	9.2647×10^{7}
R13	Resistor	1.0794×10^{-2}	9.2647×10^{7}
R14	Resistor	1.0794×10^{-2}	9.2647×10^{7}
R15	Resistor	1.0794×10^{-2}	9.2647×10^{7}
R16	Resistor	1.0794×10^{-2}	9.2647×10^{7}
R17	Resistor	1.0794×10^{-2}	9.2647×10^{7}
C1	Electrolytic capacitor	3.0720×10^{-2}	3.2552×10^{7}
C2	Electrolytic capacitor	3.0720×10^{-2}	3.2552×10^{7}
C3	Solid capacitor	1.9829×10^{-2}	5.0432×10^{7}
C4	Solid capacitor	1.9829×10^{-2}	5.0432×10^{7}
C5	Solid capacitor	1.9829×10^{-2}	5.0432×10^{7}
C6	Solid capacitor	1.9829×10^{-2}	5.0432×10^{7}
Q1	MOS-FET	4.4352×10^{-1}	2.2547×10^{7}
Q2	MOS-FET	4.4352×10^{-1}	2.2547×10^{7}
U1	Regulator	1.9000×10^{-1}	5.2632×10^{7}
U2	Micro	9.4800×10^{-2}	1.0549×10^{7}
U3	Comparator	5.3200×10^{-2}	1.8797×10^{7}
U4	Reset IC	9.4000×10^{-3}	1.0638×10^{8}
D1	Bridge rectifier	9.2192×10^{-3}	1.0847×10^{8}
D2	Signal diode	1.3001×10^{-7}	7.6914×10^{12}
D3	Transzorb	8.2368×10^{-6}	1.2141×10^{11}
D4	Signal diode	1.3001×10^{-7}	7.6914×10^{12}
D5	Transzorb	8.2368×10^{-6}	1.2141×10^{11}
D6	Signal diode	1.3001×10^{-3}	7.6914×10^{8}
D7	Signal diode	1.3001×10^{-3}	7.6914×10^{8}
D8	Transzorb	8.2368×10^{-6}	1.2141×10^{11}
T1	Transformer	2.7720×10^{-1}	3.6075×10^{6}
X1	Crystal	1.3860×10^{-1}	7.2150×10^{6}
F1	Fuse	2.0000×10^{-2}	5.0000×10^{7}
Controller total		1.8611×10^{0}	537,320 h

Table 2—*The final failure rate calculation proves the reliability expectations will be met.*

heater disconnect, if the two channels disagree. Because you have no way of knowing which channel is correct, you can't continue operating. But if a fail operative system is needed, at least three processing channels with a majority vote will do the job.

When designing a redundant system, it is often advantageous (sometimes required) to design the channels differently to avoid common mode failures in channels. Similarly, you must avoid having a single point of failure, for example, feeding all channels from the same power supply where >5-V output could cause damage to the channels or uncommanded heating. Figure 4 is the simplified diagram of the hypothetical controller, now improved so that it meets the safety requirements.

Several circuit modifications were made to satisfy the specification. Modifications included adding transzorb D3 (5 V) and fuse F1 to the power supply. If the power supply output exceeds 5 V, the transzorb will conduct and the excessive current will blow the fuse.

The simple RC reset network was replaced with a Motorola MC34064 low-voltage sensor/reset IC. It will hold the PIC controller in reset any time the supply voltage drops below the TTL level.

To recover one microcontroller I/O pin, an external clock oscillator is used. GP2 and GP4 were switched to make the internal counter available for the monitor. And, a second SV driver, Q2, was added for a totem pole driver

topology. A hardware monitor that uses a single quad comparator, such as LM139, was added too.

How does the circuit work? The PIC controller reads the thermistor output and by driving Q1, turns on and off the SV to maintain set temperature. It also performs a sanity check on the thermistor input. A short or open fault of any component within the thermistor bridge would cause the output voltage to move out of the plausible range. Similarly, an abrupt change in temperature, inconsistent with the rating of the heater and water mass, would indicate a fault condition.

Parallel with the microcontroller, the sensor voltage is fed into comparators A, B, and C of U3, forming the front end of the monitor circuit. Thermistor R3 with R1 and R4 represent a single point failure. But, because that failure is detectable by both the processor and monitor, a single sensor will satisfy the safety needs. Resistors R4 and R17 isolate a fault in either the processor or monitor to stop it from propagating to the other channel.

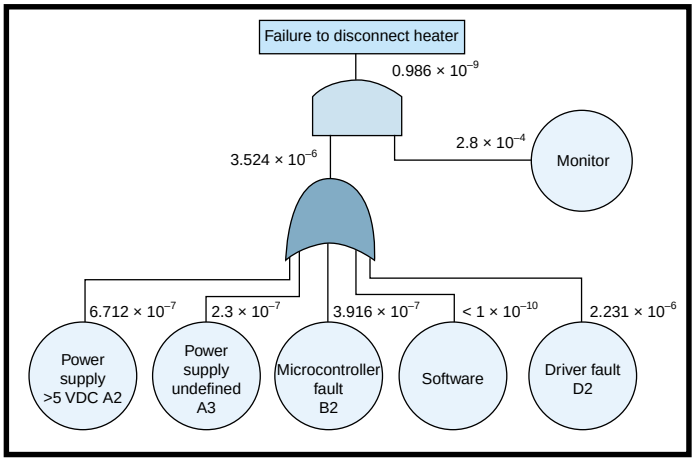


Figure 3—*FTA shows that by adding a monitor to the heater controller, the top-level event now requires two failures to happen simultaneously. The probability of such an occurrence has decreased significantly.*

All four comparators' outputs are ORed; LM139 has open collector outputs and is ideal for this purpose. When the temperature exceeds the maximum limit of 102°F, comparator A turns off Q2, thus removing power from the SV in case the microcontroller fails. Similarly, voltage comparators B and C form a window for plausibility testing of the temperature sensor. If it goes outside the predetermined limits, Q2 will be turned off regardless of the microcontroller action.

Now comes the difficult part. As I said, there must be no dormant failure in the system. All faults must be detected (it assumes only one fault happens at a time and that you're starting with a fully functional unit).

How do you make sure the comparators work properly and that Q2 can disconnect the SV? While heating, the microcontroller injects short pulses through diode D6 into the comparators. The voltage levels need to be adjusted accordingly through a resistor

divider network. This injects a fault into the monitor. At the same time, the microcontroller looks at the SV drive current as seen across R6. It must drop to zero for the duration of the test pulse. The microcontroller does the same, driving Q1 directly to verify it can turn off the SV. Because the mechanical parts of solenoid valves have 30- to 60-ms reaction time, this test pulse has no effect on the heater. If the microcontroller discovers the system response is not as expected, it will shut down the system.

Now that you know the monitor works, how do you know the microcontroller works, too? Comparator D does the job for you. Through D7, capacitor C5 is being continuously recharged every time the fault pulse is injected into the monitor, similar to a watchdog timer. It discharges through resistor R14, and if it's not recharged in time because of a fault in the microcontroller circuit, the comparator disables Q2.

But how do you prove the circuit is working? Every few seconds during the heating cycle, the microcontroller allows C5 to discharge. At this point, it must detect a drop in SV current across R6. But, what if the microcontroller is stuck high, keeping C5 charged? Then the test pulse into devices A, B, and C will stay high and Q2 will be off.

Close examination of the circuit shows that there still are several potential dormant failures. For example, transzorb D3 protecting the voltage regulator and D5 across the SV driver. To monitor D3, you may include a power-up diagnostic procedure to inject fault into the system. Careful circuit analysis may reveal that the transzorb is insufficient for the overvoltage protection and that a crowbar circuit would be more appropriate. Either way, you may consider detecting the power supply failure by a different method.

Because the analog comparators can handle 30 V_{CC}, they can be designed to detect the power supply as well as the microcontroller failure. The fuse is a different story—there is no nondestructive way to test it. You'll have to settle for the crowbar (or a transzorb) to handle the overcurrent indefinitely, or to blow a PCB track, or cause some other acceptable damage.

The potential D5 failure can be corrected by using transzorbs D5 and D8, as shown in Figure 5. A short circuit failure of either one will have the same effect as Q1 or Q2 failure and

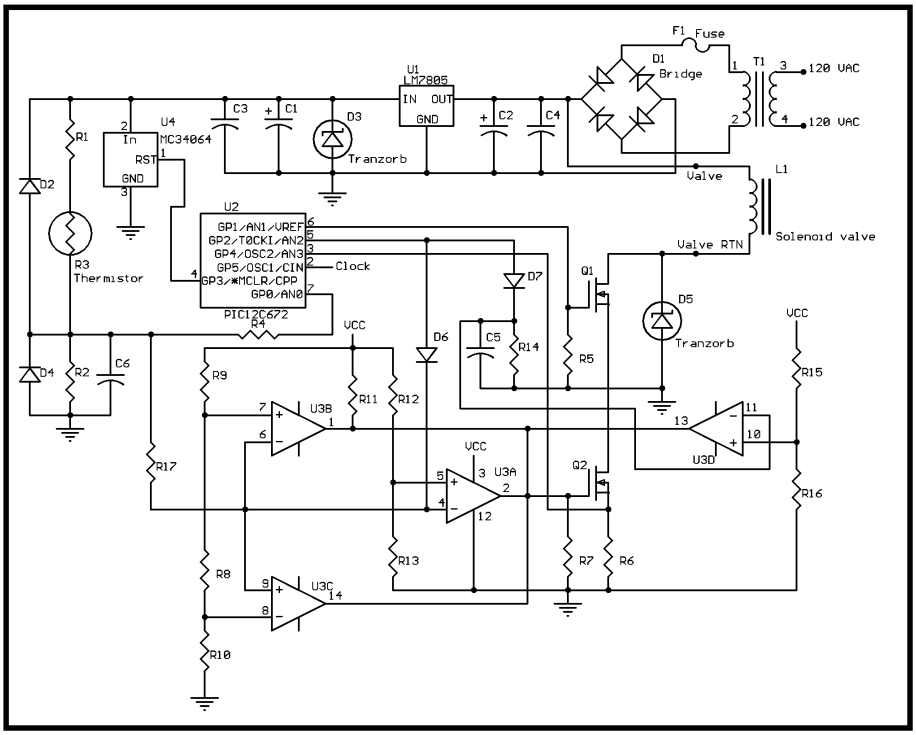


Figure 4—*A fail-safe water heater controller requires additional monitoring of circuits. This is my first attempt. It still does not satisfy the requirements.*

System: hot tub controller				Document number			Revision
Function: water temperature control				Environment: ground fixed			Date
Operation phase: al				Prepared			Checked
Failure no.	Failure mode	Possible cause	Failure effects	Method of detection	Criticality	Probability λ/h	Remarks
A1	Output = 0 V	Can be caused by a failure of any component within functional block A or an external short	Loss of water heating	Observation	Low	4.26517×10^{-7}	
A2	Output > 5 V	Failure of T1 or U1 and D3	Potential damage to U2, unpredictable effects. Maybe loss of or continuous heating	Observation and BIT	High	2.9621×10^{-7}	The failure is detected by the monitor and the heater disconnected. A double failure is needed for this condition, but dormancy exists.
A3	Output out of tolerance	C1, C2, C3, C4, D1, U1	High ripple or out-of-spec operating voltage; unpredictable.	Observation and BIT	Low	4.1592×10^{-7}	The power supply health is monitored. Reset is forced if the voltage is outside limits.
B1	Output continuously 0	U2, U4, X1, software	Loss of water heating	Observation	Low	2.3340×10^{-7}	
B2	Output continuously 1	U2, U4, X1, software	Continuous heating	Observation and BIT	High	2.4280×10^{-7}	The microcontroller lock is monitored by hardware and its erratic operation results in heater disconnect.
C	Temperature sensing not working	R1, R3, R4; Any device open or circuit short mechanical disconnect N/A	Loss of water heating control	Input signal plausibility check by microcontroller observation	High	6.6879×10^{-7}	Resistor network is monitored by BIT. Mechanical disconnect of the thermistor is prevented by design.
D1	No SV drive	Q1, R5	Loss of water heating	Observation BIT	Low	4.5431×10^{-7}	
D2	SV continuously on	Q1 or both transzorbs D5 and D8 failed short	Continuous heating	Observation BIT	High	4.5431×10^{-7}	Failure of either transzorb detected by BIT
E	Continuous SV drive or no drive	U3, Q2, R6, R7, R9–R17, C5, D6, D7	Continuous heating or loss of water heating	BIT observation	High	6.9058×10^{-7}	Monitored by microcontroller.

Table 3—*The final FMECA work sheet shows the design is safe. Faults are detected and the system shuts down.*

will be detected as such. An open circuit failure remains inconsequential until the corresponding MOSFET is damaged by a transient, at which time the condition will be detected. There also could be a far-fetched failure of the microcontroller whereby it is stuck in a loop driving the SV continuously while periodically recharging C5.

As you see, even a simple design can quickly snowball into a major project when safety becomes an issue. In this case, you may be able to show that a microcontroller failure with these symptoms is highly improbable, or you can take steps to detect such a condition. A timing window comparator is one way and a voltage comparator to track the two gate drive signals is another way of detection.

Although there is always room for safety improvement, you confront the law of diminishing returns quickly. Therefore, it's necessary to exercise good judgment and make sure you don't go overboard, increasing not only the product cost, but also complexity and occurrence of nuisance alarms. In more complex systems, you need to use tools such as testability analysis to achieve necessary fault coverage without going overboard. In simple, commercial systems such as this one, a lot can be accomplished by simply having an audible alarm to sound when system control is lost.

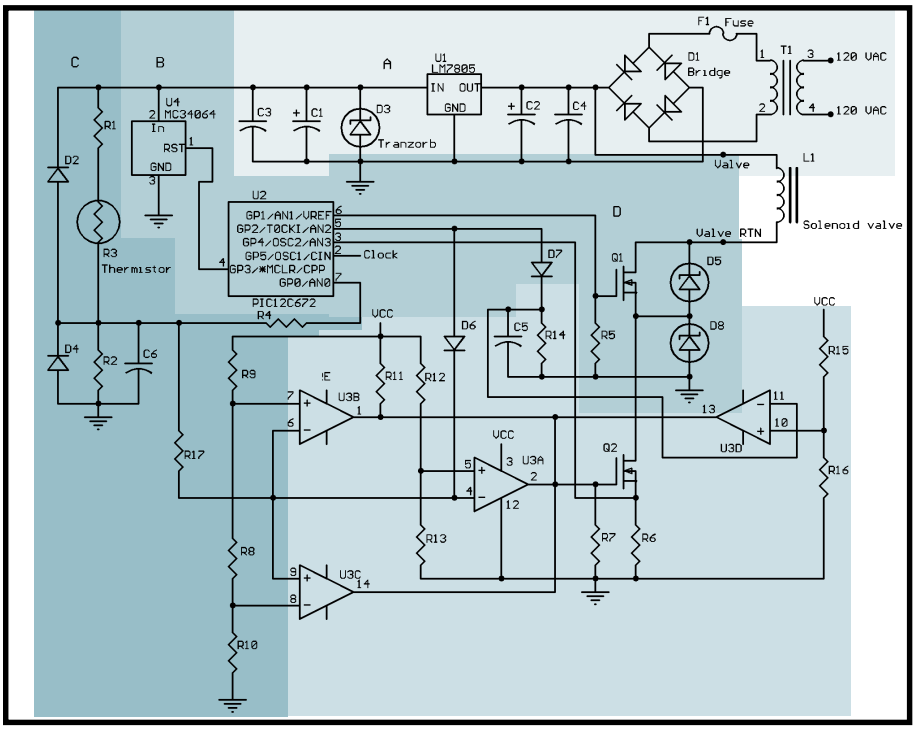


Figure 5—*This is the final design and it satisfies the specification. Failure monitoring added significant complexity to the original design.*

It's time for a word about watchdog timers, often touted as the guarantor of microcontrollers' faultless performance. They are useful, but have limitations and by themselves do not guarantee product safety. The watchdogs integral within the microcontroller are no more reliable than the micro. Although they may be useful to restart the program if it skips the rail because of a software bug or external transient, if there is a bona fide fault on the substrate, watchdogs are most likely toast.

External watchdog timers such as Maxim's are not affected by the microcontroller's failure. But, in order to rely on them alone for safety, you would have to prove that the software is structured in such a way that every conceivable fault of the microcontroller as well as any software bug will prevent the watchdog from being toggled and, consequently, will lead to reset. This is next to impossible.

As you now understand, performance monitoring can add complexity to an otherwise simple design. Usually, designing a functional product represents no more than 30% of the engineering effort. Making sure it fails (it always fails) in a safe, predictable manner takes the rest of the effort. Ensuring that BIT covers all faults of complicated systems requires a testability analysis, which is outside the scope of this article. BIT coverage in devices such as this one can be analyzed as a part of the FMECA by careful review.

WHAT ABOUT SOFTWARE?

The circuit would have been easier to implement and with deeper test coverage by using two microcontrollers, each checking the other. The problem is software. Years ago, software was viewed as the proverbial pot of gold that would cut the cost of hardware to next to nothing. This expectation has not materialized, partly because of the lack of discipline and corner cutting prevalent among commercial software developers.

Recently, I watched some unfortunate person being psychoanalyzed on a TV show. The psychiatrist would say a word and the guy stretched on a couch replied the first thing that came to his mind. This made me realize that every time I hear "software," the word "paranoia" pops into my head. Today, developing software and certifying it for a safety-critical application is expensive. The current software standard DO-178B separates code development into five categories, A, B, C, D, and E, category A being the most demanding.

Systems in which software functions can be checked by hardware supervisors often can be certified to levels D, C, or B. Even sloppy, buggy software may satisfy safety requirements if monitored by hardware, albeit at a loss of versatility, which is the selling point for software usage (see Figure 3). Where there is a critical application performed and also monitored exclusively by software, level A is the only acceptable alternative.

To write, document, and certify to level A, the code for this hypothetical controller would require several thousand engineering hours. A simple, single line of code mod is not unusual to take several months to document and recertify. In addition, level A requires separation between design and test, that is, testing must not be performed by the people who designed the software. For more information, read "Joys of Writing Software" series (*Circuit Cellar* 120–123).

There are several alternatives when designing a 100% software-driven, redundant, safety-critical system. The simplest would be a like processor, like software design. Identical hardware channels running identical software are used, comparing each other. This is not a preferred method because you must show that no common mode failure is possible; there is no condition, be it wrong data, external interference, or fault, that can bring both channels down simultaneously. You would waste more time trying to prove this than if you pursued an alternative.

A more common method is a like processor, different software design. There are two similar hardware platforms, but the software for each is designed by a different engineer. Sometimes there are additional differences, such as the control channel performing calculations in 16 bits, and the monitor does it in 8 bits and uses the free time for communications. Often, to satisfy level A separation requirements, team A writes the controller and tests the monitor software and team B writes the monitor and tests the controller.

For the most critical applications where paranoia is the rule of the day, the different hardware, different soft-

ware approach is taken. It is assumed that a fault may exist in the microcode, and therefore, different processors are used. This may sound drastic, but when faced with a multimillion-dollar satellite's computer hanging up during the first orbit, going through the extra development effort is justified.

SV1	SV2	Pressure 1	Pressure 2	Pressure 3
off	off	1	0	0
on	off	1	1	0
on	on	1	1	1

Table 4—*The solenoid valve isn't considered part of your design responsibility. It is usually sufficiently reliable for shutting off the fuel supply. If you need to include it in the system, you may have to use two and perform diagnostics as shown in this table.*

For triple and more redundant systems, these approaches are equally applicable. The advantage of triple and higher redundancy is that devices can keep operating under failure conditions, as long as two out of three agree.

THE RESULTS

Now that you have modified the design after considering the reliability, FMECA, and FTA findings, let's look at the results. Let's discuss the functional block FMECA (see Figure 5). The first step is to look at the effect of the additional components on reliability prediction. Table 2 shows the updated design and includes improvements such as decrease of the junction temperature and application of duty cycle.

With the failure rate values calculated, you can proceed to perform FMECA (see Table 3).

The important result is that all high criticality failures are monitored (see Figure 6). Again, the fault probability numbers for nodes are calculated by adding λ_p from the reliability prediction for every component within the functional block that could cause the given failure and multiplying it by 10^{-6} to obtain failure probability per 1 h. Where two failures are needed for the top event, the inputs are logicially ANDed (multiplied).

I should mention power supply failure mode A2, as well. For the output voltage to exceed 5 V and cause continuous heater operation, multiple faults would be required. Normally, FMECA and FTA are prepared on the basis of single faults. Logic AND gates exist for fault propagation, and the probability of multiple failures would be in the order of 10^{-12}. Because the power supply block can contain several dormant failures (i.e., the fuse and transzorb/crowbar circuit), you must treat the probabilities as logic OR. Fortunately, the monitor outside the power supply block will detect the excessive 5-V rail and switch off the SV via Q2.

A quick look at the FTA in Figure 6 shows that you exceeded the safety requirement by three orders of magnitude. But, there remains one other potential problem, the external valve. Its connection to the driver can short to the ground and cause continued energization of the valve. Or, the valve can be stuck in the open position.

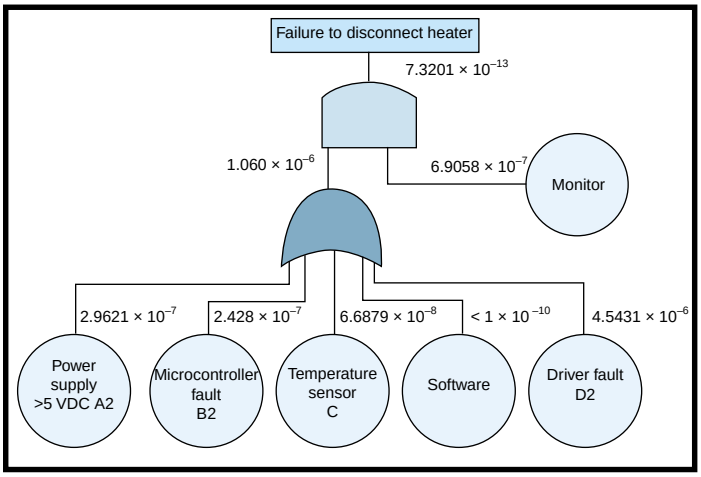


Figure 6—*The final fault tree analysis, which includes the monitoring circuit, proves that no single failure within the controller can cause a catastrophic event.*

The short to the ground problem can be addressed by careful wiring or, in a critical application, by using a high side driver or a dual high-low side interface. The mechanical failure of the solenoid valve is solved in many systems by using a high-quality valve with a filter on the input line to prevent dirt particles from entering. In critical applications, two valves are used. But this approach is expensive.

Both of the solenoid valves must have a totem pole driver. To monitor the valves' operation, you also need three pressure switches, one upstream of the valves (PS1), one downstream (PS3), and the third between them (PS2). The power-up BIT routine (P-BIT) energizes the valves as shown in the truth table (see Table 4) and reads the pressure to verify their operation. PS1 is there only to make sure the test routine is not performed without gas pressure, which would result in fault.

MAINTAINABILITY

The reliability prediction indicates that after you ship 10,000 units, you'll be ready to service at least two problems per day. You want to keep customers happy with a quick repair turn around time (TAT). You also want to keep the cost of service calls low.

Based on the complexity and cost of the controller, repair may be by replacement. The system is comprised of three subassemblies—the controller, temperature sensor probe, and solenoid valve. None of these is field repairable, so they are called line replaceable units (LRUs).

The maintainability analysis generates data showing the time needed to identify the faulty LRU, the time to replace it, and the time to retest the system and bring it up to speed again. This is called mean time to repair (MTTR), and identifies the tools, procedures, spare parts, and so forth needed for field repair. The analysis provides useful information for business planners and design engineers. For example, you may discover that a simple design change may eliminate uncommon tools otherwise necessary for the technician to carry. Or you may discover that the 5 min. required to replace the controller may have to be preceded by a 2-h system disassembly and followed by the same duration assembly.

Again, the most important aspect of the design is testability. Not only is it important in determining system safety, an effective BITE (built-in test equipment, the circuitry performing BIT) identifies the faulty LRU and displays it on the controller cabinet or transmits the data by a communications link. This reduces the MTTR. But, a 100% accurate BIT is nearly impossible to achieve. Usually 95% accuracy of fault isolation is acceptable; mean time between unscheduled removals (MTBUR) signifies the fault isolation accuracy. A controller with 10,000-h MTBF and 95% isolation accuracy will have 9,500-h MTBUR.

SUMMARY BENEFITS

In this two-part series, I approached a simple controller design from the perspective of reliability and safety. You learned how useful the reliability prediction, FMECA, and FTA become to an electronics designer. They help you create safe, robust designs, as well as provide insight into products' futures in terms of warranty, repairs, maintenance, and cost of ownership.

While covering this series, some questions were generated by introducing important subjects such as software development procedures and testability. These subjects need separate articles for a full discussion. For now, I want to reiterate that formal testability analysis is not only instrumental for BIT activity, but should be kept in mind while designing, even when there is no BITE present.

This applies equally to hardware and software. This requirement adds complexity to a simple design, but the alternative would be to prove the performance by analysis. Granted, there are functions that can't be tested, but the fewer the better. Proofs by analysis can be tedious, time-consuming, and quickly reach a dead end if conflicting engineering opinions come into play.

George Novacek has 30 years of experience in circuit design and embedded controllers. He currently is a general manager of Messier-Dowty Electronics, a division of Messier-Dowty International, the world's largest manufacturer of landing-gear systems. You may reach him at gnovacek @nexicom.net.

SOFTWARE

Reliability calculations are available on the *Circuit Cellar* web site.

REFERENCES

S.E.R. subcommittee, "Automotive Electronics Reliability Handbook," Society of Automotive Engineers, Warrendale, PA, February 1987.

P. Tobias and D. Trindale, *Applied Reliability*, Van Nostrand Reinhold, NY, NY, 1986. ISBN 0-442-28310-5.

U.S. Department of Defense, *Electronic Reliability Design Handbook*, MIL-HDBK-338, General Policy Series, no. 542.

U.S. Department of Defense, *Maintainability Program for Systems and Equipment*, MIL-HDBK-47OB, Washington D.C.: Government Printing Office, June 1995.

Ingo Cyliax

A Cup of Java

Part 2: Standard Java in Real-Time Embedded Systems

If last month's article just got you percolating, satisfy your Java craving with the second part of this series as Ingo delves into some of the domestic issues (e.g., threading and garbage collection) of working with Java in embedded systems.

Last month I introduced you to Java in the embedded world. In Part 1 of this series, I discussed three aspects of the Java strategy, the language, Java Virtual Machine (JVM), and run-time environment. Also, there are some advantages to using Java in the embedded space (e.g., code density). Java byte code is compact and uses a small amount of memory.

Another advantage is that the language is easy and safe. It won't let you make most of the common errors that C and C++ programmers do. It's being taught in most schools now as the primary programming language.

The lack of traditional pointers and garbage collection makes Java safe. In Java, pointers can only point to objects that have been allocated by the system. It's not possible to assign values to pointers that are not pointers to objects. Also, Java automatically tracks the memory you have allocated for objects and releases it after it's no longer in use. This prevents memory leaks, which is a common problem with C and C++ programs.

MULTI-THREADING

Java supports multi-threading. You can create and manage threads in much the same way as in other languages. To create a thread, make a thread object and implement a `run()` method. This is the method that will be called when the thread is started. Each thread in Java is an instance of the thread or inherited Thread class (see Listing 1).

Upon startup, the method `main` is called, just like in the examples from last month. The main thread then creates three new instances of your `thread1` class, which extends from the Thread class. This means that it inherits the `start` method. This method does all of the startup and then calls the run method, which is the method you want to run when your thread is created. In this example, you simply create three threads that run the same method.

There are points to consider. In `thread1`, there are the count (has static scope) and number fields. Static things in Java exist before an object has been instantiated. So, the `main` method and the count field in your class exist before any objects have been created. A fresh instance of the number is created every time a `thread1` object is created, just like you do to start the threads.

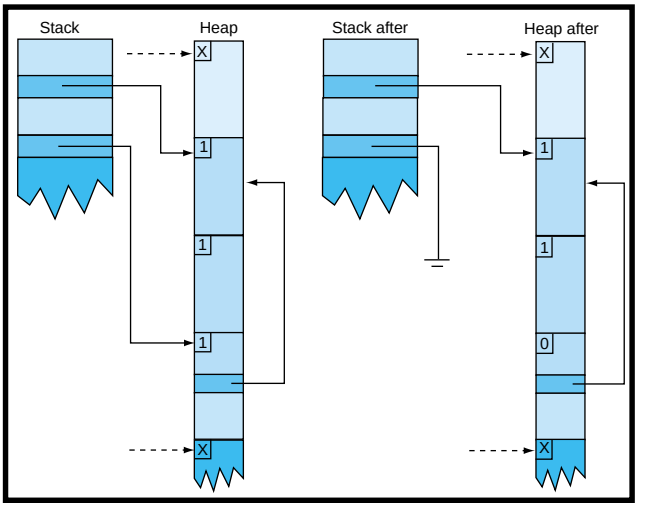


Figure 1—*In reference counting garbage collection, each object has a counter associated with it. When the counter reaches zero, the object is free and can be reclaimed.*

Listing 1—*Here is an example of some Java threads. They are created and run in parallel to the main thread. After all of the threads have finished running, the program exits.*

```
public class thread1 extends Thread {

   static int count = 1;
   int number = count++;

   public void run () {
       System.out.println("Thread "+number+" Started");
     try {
        sleep(100);
      } catch (InterruptedException e) {
         System.err.println("Interrupted");
  }
    System.out.println("Thread "+number+" Done");
}
public static void main(String[] args) {
   int n;
   n = 1;
    new thread1().start();
    new thread1().start();
    new thread1().start();
 }
}
```

Each time the object is constructed, the count initializes the number field and the count field is incremented.

So, when a thread is running in the `run` method, it sees its instance of the number field, and can then identify itself with a unique number. The thread running in the `main` method will not be able to access a default number field. It can, however, access each thread's copy of the number by referencing it (i.e., `thread1.number` or `thread2.number`).

Your program always starts with three threads, which print their field numbers and then sleep for 100 ms before terminating. The main thread simply terminates and the run-time system waits until all of the threads have finished before terminating. To compile and run the program, use the commands stated in Listing 2.

Java threads do everything you normally expect of threads. Threads can have different priorities, which are read with the `getPriority()` method and modified with the `setPriority()` method. Also, there are ways to wait and notify waiting threads (`wait()` and `notify()`). In fact, each object has a thread wait queue and threads are added to an object's queue using the object's `wait()` method.

Because object data can be shared among threads, there is a way to guarantee exclusive access to object data (see Listing 3).

Here, you can create three threads and an array object `map()`. The threads have to swap the two elements in the array. However, this is intended to be done atomically. The two elements add to five and each swap has to be completed before the next thread can swap the elements. The main thread prints an error message when the total is not preserved (see Listing 4).

Listing 2—*Sharing data among threads can be a problem. Here, I'm trying to swap two items in an array with multiple threads. But, because the operations are not atomic, the data will get corrupted.*

```
hugo 19% javac thread1.java
hugo 19% java thread1
Thread 3 Started
Thread 2 Started
Thread 1 Started
Thread 3 Done
Thread 2 Done
Thread 1 Done
```

Basically, each thread can step on each other's swapping process and leave either all 2s or all 3s in both memory locations. Also, the main thread only gets a snapshot of the object and can produce varied results.

Of course, this is bad multi-threaded programming in any language. And, other thread systems have various ways of providing data locking.

Because Java was designed with multi-threading support from the onset, it supports the synchronize primitive. There are two ways to use this. You can mark methods that belong to an object as synchronized. Then, only one thread is allowed to run a synchronized method. Another way is to surround the critical section in the code with a `synchronize()` block. Check out the example in Listing 5.

This is essentially the same program as in Listing 3, except that the map object is protected with a critical section when the elements are being swapped and checked for consistency. Of course, this program never has a data crash like Listing 3.

The synchronize primitive in the Java language is implemented as a monitor in the JVM. There are two instructions, `monitorenter` and `monitorexit`, that take an object reference and expect to lock the object atomically. If a thread tries to gain access to an object by using the `monitorenter` instruction that has already been locked, it has to wait until the thread that holds the lock executes a `monitorexit`. Otherwise, the count is simply incremented and decremented each time the same thread enters and leaves the object. The JVM specification explains the semantics of the requests for locked memory references.

Having thread support would seemingly make Java a natural for real-time applications. This is generally true, but most JVM implementations are geared to run on network clients and servers and are expected to implement GUIs or access databases. Although making GUIs and database access responsive is considered real-

time, it's not in the same class of hard and soft real time that you typically consider for control applications. Thus, Java run-time implementations tend to be optimized for the GUI- and server-based accelerations, and not deterministic real-time applications. This is evident when you look at how multi-threading is implemented.

IMPLEMENTING

There are several ways that multi-threading can be implemented. One way is to map Java threads to underlying thread constructs of the OS. This has a nice advantage in a system that mixes Java and C/C++ programming. All threads are managed in the same way. If this is done in an RTOS, you make some threads real time and assign fixed priorities to them. But, in general, you won't be able to share Java and C and C++ data resources in this way because there is no effective way to map them.

Java threading can be implemented in the JVM also. In this way, the OS running has no idea that there are threads in the JVM. A JVM instance then looks like a single process (or thread) to the OS and other threads and processes in the system, and the threads in the JVM are just simulated. One nice feature of this approach is that you can keep the JVM in sort of a sandbox and allocate a fixed amount of run-time resources to it. This way, the Java threads running within the JVM can't hog the system because all the threads in the JVM have to make do with the time resources available.

Worse, some Java implementations can also implement a dynamic priority scheme. The priority of a thread will age the longer it runs. This allows threads that deal with interactive events like keyboard input to have higher priorities than background threads that run when there isn't interactive activity happening.

So, what are the issues? Some of these schemes (i.e., dynamic priorities and software-based threading) won't work for real-time applications. They can't guarantee timeliness, such as deterministic response time to events, or even give guarantees to the amount of CPU time required.

Listing 3—*To protect the data, you can synchronize access to an object using the synchronize block. You can also mark a method belonging to an object or class with the synchronize keyword. This protects access to an object using a monitor. Only one thread is allowed to execute at a time.*

```
public class thread2 extends Thread {

   static int count = 1;
   int number = count++;

   static int map[];

   public void run () {
     int t;
        System.out.println("Thread "+number+" Started");
     while( true ){
        t = map[0];
        map[0] = map[1];
        map[1] = t;
     }
   }

    public static void main(String[] args) {
     map = new int[2];
     map[0] = 3;
     map[1] = 2;
      new thread2().start();
      new thread2().start();
      new thread2().start();
     while( true ){
        if(map[0]+map[1] != 5){
            System.out.println("map[0] "+map[0]+
             " map[1] "+map[1]);
        }
     }
   }
}
```

Implementing a real-time scheduler for Java requires, at the minimum, OS and hardware (timer) support. There are also refinements to the language that need to be considered, like how critical sections are implemented and defining different classes of threads. These are all features available in many RTOSs, and would be required to make Java run in real-time environments.

GARBAGE COLLECTION

Another big thorn for real-time performance is garbage collection (GC). Let's take a look at what GC is, and then how it's typically implemented and the real-time and embedded systems issues involved.

As I mentioned, Java uses a heap-based storage system where the data for objects is stored. Unlike C or C++, you don't explicitly call a `free()` call to de-allocate memory from the heap. Java uses GC to do this automatically.

Objects in Java are stored on the heap as blocks of memory and referenced using an object reference. An object reference is much like a pointer, except you can only assign them and de-reference the fields and methods inside an object.

After being allocated, an object is live in memory as long as there is reference to it. If there is no reference, then you can't access it. When an object is not live, it continues to sit on

Listing 4—*Here you see the program compiling and running.*

```
hugo 13% javac thread2.java
hugo 13% java thread2 | head
Thread 3 StartedThread 1 StartedThread 2 Started
map[0] 3 map[1] 3
map[0] 3 map[1] 3
...
```

the heap until it's removed by the GC. Usually, the GC is a background thread that wakes up when free memory is low and memory needs to be freed for new objects.

So, how does the GC figure out what objects on the heap are dead? After all, a dead object is not referenced anymore.

One way is to use reference counting memory (RCM). With RCM, a counter associated with each memory block corresponding to an object. Whenever the object referenced is assigned to an object reference, the counter is incremented. Whenever an object reference is lost, the counter is decremented. When the counter reaches zero, the memory block is not used anymore. All the GC has to do is sweep through the heap and look at the reference counter to see if this memory is free and can be recycled the next time memory needs to be allocated (see Figure 1).

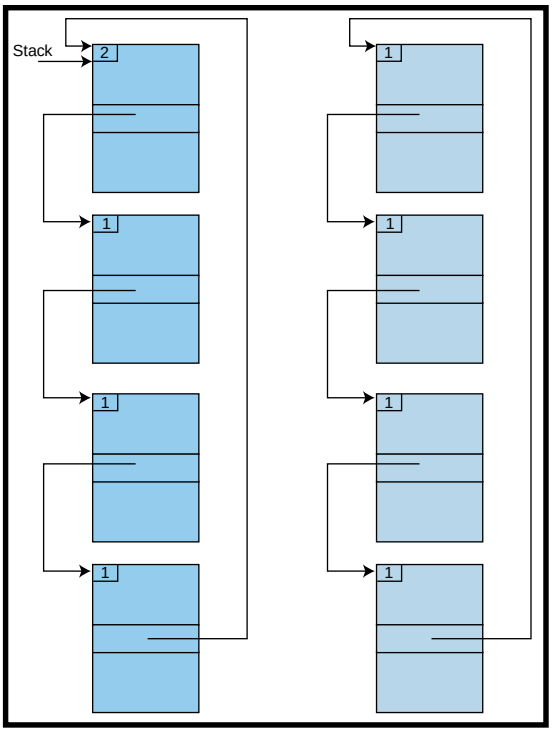


Figure 2—*Circular references are a problem in reference counting memory garbage collectors. For example, when a list is deallocated, the head of the list decrements to 1, but each element in the list is still referred to by its predecessor. Even though the list is freed, the RCM will not detect it and waste the memory.*

SMOOTH SAILING?

It sounds easy, but there are a couple of drawbacks with RCM. One problem is that it is possible to have loops of objects that reference each other, but are not referenced by the program. Consider a circular list of objects. After the reference to the list is released, only the counter of the object that the list reference pointed to is decremented. However, because this object is still referenced by the end of the list, the whole list of objects will still have nonzero reference counters and the GC will think it is still being referenced (see Figure 2).

Another problem is that RCM-based GC can't de-fragment memory easily. It can merge adjacent memory blocks by combining them when they're free, but if the free memory blocks are not adjacent and too small to be reused, they are lost.

In practice, RCM is not efficient when implemented in software and, as far as I know, RCM is not used in any JVM implementations.

In the stop-and-copy algorithm, the GC sweeps through the program's stack and finds all of the object references on the heap. It then copies the object to a new heap and adjusts the reference on the stack to match the new location of the object. It also has to follow any references that are in the field data of any objects it copies to make sure it moves all of the objects. If the GC encounters another reference to an object that has already moved from the old to the new heap, it simply updates the reference to the new location. Then, when the GC has copied all of the objects, the program can continue to run using the object data on the new heap. You can check out this process in Figure 3.

This algorithm will automatically de-fragment the heap when it copies the old to the new heap. It is expensive to copy the data in the object from the old to the new heap. Also, the GC needs to be able to tell the reference pointers from other data words on the stack and in the object's field data. In addition, you need enough memory to contain almost two heap's worth of data during the copying. In systems that support virtual memory, the last two issues don't have as large an effect because copying can be accomplished by simply changing the mapping of virtual memory pages, and the data is not physically copied.

Another algorithm used in GC is mark-and-sweep. In this case, the GC will go through the stack and field data, marking each object it finds. When reclaiming memory, the GC can go through the heap and find the objects that are not marked as being used. Of course, in this algorithm, just like RCM, de-fragmentation is limited to adjacent free memory objects. Note that it does not need to rewrite object references on the stack and field data because the objects do not move.

This covers the three big algorithms. Usually, software JVMs use either the stop-and-copy or mark-and-sweep algorithm. There are some high-performance GCs that adapt themselves and switch between the

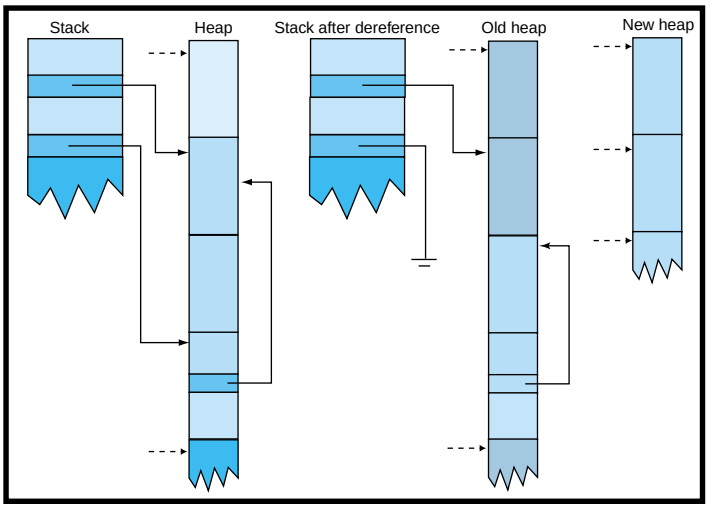


Figure 3—*In stop-and-copy, you copy all of the used memory segments that are still referenced to a new heap. This frees the unused objects on the heap and compacts and de-fragments the heap.*

two, depending on the memory shortage (i.e., it uses the mark-and-sweep algorithm for speed and then switches to stop-and-copy when it gets low on free memory).

For real-time systems, this sort of memory management system is a nightmare. For instance, when the GC is running and modifying object references with stop-and-copy, the whole Java system has to stop. This, of course, adds unpredictable latency to Java programs because you can't predict when this will happen. It is amplified in embedded systems that have less memory than their workstation or server counterparts and are likely to require GC more often.

There are a couple of techniques that can be used to make real-time performance more predictable. You can make your GC incremental. In this case, the GC will always run in the background and perform only small changes to the heap. This can be engineered so that each incremental step will take a bounded amount of time, perhaps only dealing with one object reference at a time.

Yet another technique is to divide Java threads in real time and other threads. These would have separate heaps. The heap that is not in real time can then be collected at any time, while making sure that the real-time threads are not blocked. Real-time threads would have to bound their memory usage, which is only allocated a finite amount of memory, and not use GC to free memory.

Keep in mind, this requires discipline from the programmer or tools that can analyze Java programs to make sure that the real-time threads do not have memory leaks.

There are several products that address these Java threading and scheduling issues in real-time systems. NewMonics and WindRiver both provide software-based real-time JVM solutions. Also, aJile Systems has a Java chip that diminishes some of the threading problems in hardware.

Listing 5—*Notice how you can surround the critical section in the code with a* `synchronize()` *block.*

```
public class thread3 extends Thread {

   static int count = 1;
   int number = count++;

   static int map[];

   public void run () {
     int t;
        System.out.println("Thread "+number+" Started");
      while( true ){
        synchronized(map){
          t = map[0];
          map[0] = map[1];
          map[1] = t;
        }
     }
   }

public static void main(String[] args) {
   map = new int[2];
   map[0] = 3;
   map[1] = 2;
    new thread3().start();
    new thread3().start();
    new thread3().start();
   while( true ){
      synchronized(map){
        if(map[0]+map[1] != 5){
             System.out.println("map[0] "+map[0]+
            " map[1] " +map[1]);

        }
      }
     }
   }
}
```

Ingo Cyliax is a computer and electrical engineer (BSCEE) and the founder of EZComm Consulting, which specializes in embedded systems and FPGA design services as well as troubleshooting. Ingo has been writing about various topics ranging from real-time operating systems to nuts-and-bolts hardware issues for several years. He can be reached at cyliax@ezcomm.com.

RESOURCE

Real-time standards, www.rtj.org.

REFERENCES

B. Eckels, *Thinking in Java*, Prentice Hall, Upper Saddle River, NJ, 2000.

B. Eckels, *Thinking in Java*, Prentice Hall, Upper Saddle River, NJ, 2000.

D. Flanagan, *Java in a Nutshell*, O'Reilly Publishing, Sebastopol, CA, 1999.

P. Hyde, *Java Thread Programming*, Sams Technical Publishing, Indianapolis, IN, 1999.

SOURCES

Real-time JVM
NewMonics Inc.
(630) 577-1590
Fax: (630) 579-9136
www.newmonics.com

JWorks
WindRiver Systems, Inc.
(510) 748-4100
Fax: (510) 749-2010
www.wrs.com

Java chip
aJile Systems Inc.
(408) 557-8279
Fax: (408) 557-8279
www.ajile.com

Fred Eady

Rabbit Season

Part 5: A Board You Can Afford

In true rabbit fashion, this project has resulted in quite a proliferation of articles. But Fred's got one last hop for the Rabbit dev board project. The jump to Internet connectivity was as simple as a twitch of the nose.

You know, having that Rabbit and Alice around has really made things interesting lately. And, another old friend, Grace Slick of Jefferson Starship, took some time last week to stop by the Florida room to say hello to Alice, the inspiration for her song, "White Rabbit," whom she hasn't seen in some 30 years. It's hard to work with all the music in the air.

Anyway, when thinking about this article, I came to a conclusion. Now that I've got a couple of *Circuit Cellar Online* articles behind me, it's obvious that I'm part of a powerful technical information outlet. I'm sure that if the *Circuit Cellar* columnists you follow every month in the print version were given unlimited article space, you'd probably be enjoying a magazine as thick as a phone book. Fortunately, the web site allows me and other authors to provide twice as much information each month.

After receiving some beta PPP networking code from Rabbit Semiconductor's network guru Charlie Krauter, it looks like the PPP hardware and all of the construction details I promised last month will appear online.

LET'S GO TO THE HOP

This is the last installment in the Rabbit series and, just as I moved the Seiko PPP IC and ISOModem to the web, I am moving the Rabbit 2000 Core Module to print. Everything you need to make the Rabbit 2000 IC the star of your design is included on the Rabbit 2000 Core Module, and is designed to be plugged in. This eliminates having to design and lay out the microcontroller engine part of your project. Because it's my job to make your job easier, Photo 1 is a shot of the Rabbit 2000 Core Module holding court on its development board.

Photo 2 shows that, in addition to the Rabbit 2000 Core Module, I've assembled a little plug-in module of my own. It consists of two parts, a MAXIM 235 and a 1-µF tantalum capacitor. The MAX235 is an RS-232-to-TTL interface IC used to convert modem signal levels to TTL coming into the Rabbit 2000 Core Module and TTL-to-RS-232 signal levels going out. It's a standard implementation, as is seen in Figure 1. Because the name of the game is dialup and it doesn't have

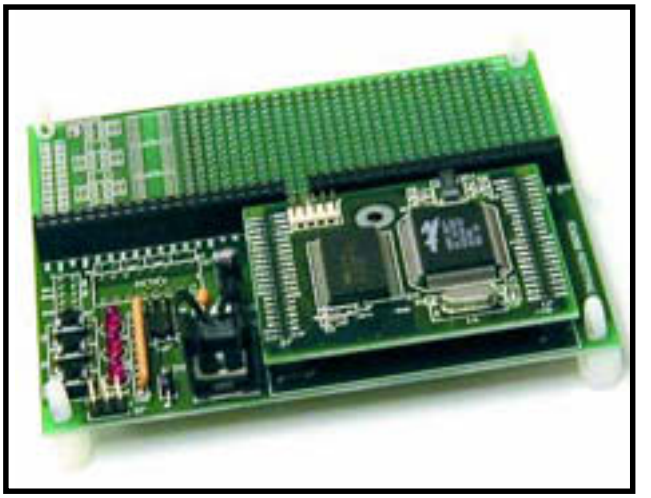


Photo 1—*I like this setup a little better than the Jackrabbit. The Core Module allows more flexibility as to what you can directly connect to it.*

to be fast, I pulled an old U.S. Robotics 28.8 external modem off the shelf for this project.

ONE HOP TO THE INTERNET

To check out *Circuit Cellar Online* if you aren't using a cable modem or DSL, dial a phone number to connect to your ISP and open the window to the 'Net. If you use a dial-up modem for Internet access, most likely the protocol you use to establish contact with the ISP is PPP, or point-to-point protocol. I'll use the same PPP rules that your desktop computer uses to get a phone to the Rabbit's ears and a call through to the Internet.

The mission this time is to get the Rabbit on the phone, and then on the Internet. After that, you can instruct it to send e-mail or serve HTML pages. The problem is, there are no predefined Dynamic C PPP libraries or routines, but this setback was quickly eliminated with a single e-mail to Rabbit Semiconductor. It seems that Charlie was working on the PPP code as I was working on this article.

PPP is taken for granted. But, PPP is an important step in the Internet access process. Like everything else in life, you don't appreciate it until you can't get it. With that, I'm going to delve into what makes PPP tick and some of the code necessary to hop around in the Internet.

PPP

A dial-up session to your ISP is defined as a point-to-point link (i.e., from your modem to a modem at the ISP location). When the physical modem-to-modem session is established, you can't just pass packets to the interface in any way you choose. There has to be some standard of communications, an agreement between the sending and receiving parties. This normally involves the determination of which protocol will be used and the maximum size of the packets that will flow between the peers within the constraints of the selected protocol.

The point-to-point protocol was designed to provide a standard method for transporting multi-protocol datagrams over simple point-to-point links. A simple point-to-point link is usually a full duplex connection that delivers packets in the order that they are transmitted. A datagram is the unit of transmission in the network layer (e.g., IP) and may be encapsulated in one or more packets before being passed down to the data link layer.

PPP was designed for easy use and implementation. To make the concept of PPP clearer, you can break it down into three main components—encapsulation, a link control protocol (LCP), and a collection of network control protocols (NCPs).

A packet is the basic unit of encapsulation. Packets are passed between the network layer and the data link layer. Because the data link layer is responsible for creating frames, a single packet or multiple packets are usually the cargo of frames. PPP uses a frame format and protocol similar to the HDLC (high-level data link control) frame, which begins and ends with a flag or sync character 0x7E. The combination of encapsulation coupled with framing allows different protocols to be transported over a PPP link.

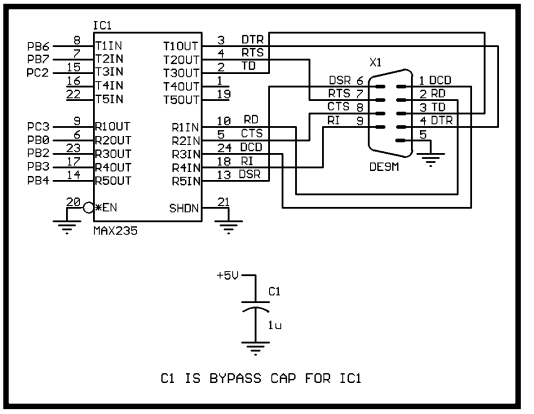


Figure 1—*There's nothing fancy or unknown here. This circuit has been used for years.*

RABBIT IN A FIELD

Figure 2 shows that a standard PPP frame consists of six fields. Let's examine them left to right. I've already mentioned the flag field, a single byte that sits at the beginning and end of a frame. If you're wondering what would happen if your data contained a 0x7E, not to worry. PPP has the capability to escape such characters. In fact, PPP escapes characters less than 0x20. This escape process involves exclusive ORing the control code with 0x20. For instance, to send 0x7E as data, 0x7E would be exclusive ORed with 0x20, yielding 0x5E. Then the escape character 0x7D is inserted before the 0x5E, resulting in a two-byte sequence of 0x7D and 0x5E, representing the data value of 0x7E. At this point, the receiving machine simply does another exclusive or with 0x20 against the 0x5E and discards the 0x7D escape character to return the original value of 0x7E.

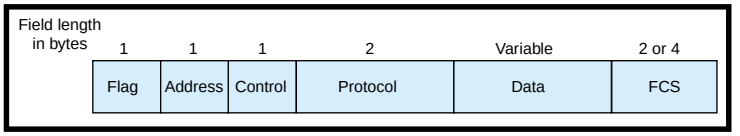


Figure 2—*You just visited a similar "field" in the world of Ethernet.*

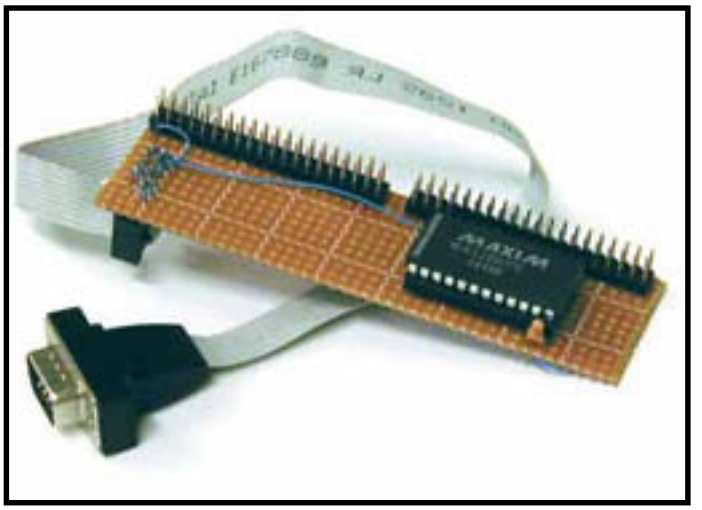


Photo 2—*This is really quick and nasty, but the functionality is clean.*

PPP has no method for assigning individual station addresses. So, the Address field is a standard broadcast address of 0xFF. PPP is connectionless (i.e., it doesn't require a formal session with sequenced frames). The 0x03 in the Control field specifies this as it represents the transmission of data using an unsequenced frame. If you were to dig out the bit scheme, you'd find that the 0x03 is the unnumbered information (UI) command with the poll/final (P/F) bit set to zero.

Knowing what's in the information field of a PPP frame is important. The two bytes of the protocol field identify the protocol that's encapsulated in the information field. Table 1 states some of what you can put into the protocol field and what the values represent with the protocol. All values are in hexadecimal.

0001 to 001f	Reserved (transparency inefficient)
0021	Internet protocol
0023	OSI Network layer
0025	Xerox NS IDP
0027	DECnet phase IV
0029	Appletalk
002b	Novell IPX
002d	Van Jacobson compressed TCP/IP
002f	Van Jacobson uncompressed TCP/IP
0031	Bridging PDU
0033	Stream protocol (ST-II)
0035	Banyan vines
0037	Reserved (until 1993)
00ff	Reserved (compression inefficient)
0201	802.1d Hello packets
0231	Luxcom
0233	Sigma network systems
8021	Internet control protocol
8023	OSI Network layer control protocol
8025	Xerox NS IDP control protocol
8027	DECnet Phase IV control protocol
8029	Appletalk control protocol
802b	Novell IPX control protocol
802d	Reserved
802f	Reserved
8031	Bridging NCP
8033	Stream control protocol
8035	Banyan vines control protocol
C021	Link control protocol
C023	Password authentication protocol
C025	Link quality report
C223	Challenge handshake authentication protocol

Table 1—*During the debugging process, I got to know the C021 code well.*

The Data, or information field, is interesting because it can be zero bytes in length. The default maximum length of this field is 1500 bytes (this is negotiable). The Data field can also be padded. An easy way to find the end of the Data field is to find the flag at the end of the frame and count back past the two FCS bytes in the Frame Check Sequence field.

Usually, where you find fields that comprise a frame, you find a protocol that goes with them. LCP is the PPP method of establishing, configuring, maintaining, and terminating the point-to-point connection. LCP packets are akin to marines because they are the first to fight. Each peer participating in a PPP link sends LCP packets to configure and test the data link before establishing communications over the PPP link. You can be authenticated only after the link is deemed satisfactory. So, authentication comprises supplying the log-on user ID and password to the remote peer.

After the physical link is made, PPP then sends NCP packets to choose and configure one or more network-layer protocols. After each desired network-layer protocol has been configured, datagrams from these protocols can be exchanged between the peers on the link. The link configuration will exist until LCP or NCP packets close it down or an operator or predefined timer stops the link.

PHASES

To better understand LCP, you can divide its functionality into four phases. As I stated earlier, the first operation is physical link establishment. In Figure 3, link dead is the starting and ending point of the phase relationships. The modem is commanded to dial and subsequently contacts the answering modem. Carrier detect from the modem on your end signals the LCP that the physical link is active and ready to be used.

The UP signal from the physical layer puts the LCP process in the link establishment phase. Before any datagrams can be exchanged over the newly opened link, LCP opens the connection and negotiates configuration parameters. This phase is complete when a configuration-acknowledgment frame (Configure-Ack) has been both sent and received. All of the configuration options are assumed as defaults until altered by the negotiation process. These configuration options are not protocol-dependent. The NCP handles any protocol-dependent configuration issues in the network-layer protocol phase.

If necessary, you are authenticated before any network-layer protocol datagrams are exchanged between the peers on the point-to-point link. PPP supports two authentication protocols. Password authentication protocol (PAP), a simple method of establishing user identity, is the most popular. Following the establishment of the PPP link, the user ID and password are sent to the remote peer until the authentication is positively acknowledged or the link is broken. With PAP, everything is sent in the clear with no encryption. This leaves the remote peer in charge of determining how to affect some sort of log-on security. The second authentication protocol is challenge handshake authentication protocol (CHAP). This method sends a

Code	Identifier	Length	Data
1 byte	1 byte	2 bytes	Variable

Table 2—*This may look like any other field I'm discussing with you today, but remember that encapsulation is the key to making it all work.*

1	Configure-Request
2	Configure-Ack
3	Configure-Nak
4	Configure-Reject
5	Terminate-Request
6	Terminate-Ack
7	Code-Reject
8	Protocol-Reject
9	Echo-Request
10	Echo-Reply
11	Discard-Request
12	RESERVED

Table 3—*These are bounced back and forth between peers many times during a session.*

challenge message to the remote peer, which responds with a calculated value. If the sender agrees with the returned value, the authentication is positively acknowledged. Because this calculated value is always unique, CHAP obviously offers a higher level of log-on security than PAP. That's all good, but by default, authentication is not necessary.

After the link is opened and the user or peer is authenticated, network-layer protocol configuration negotiation occurs. Each desired network-layer protocol (IP, IPX, etc.) is configured separately by the appropriate NCP. As long as the PPP link is up to this point, each individual protocol can be brought up and taken down at any time by its controlling NCP. If LCP closes the link, the network-layer protocols are alerted so that they can take appropriate action.

LCP can terminate the link at any time. Normally, this is planned and executed by you or the administrator. Other physical events, such as the expiration of an idle-period timer can also terminate the link.

Each LCP phase employs a particular frame. Link-establishment frames consisting of link configuration packets (Configure-Request, Configure-Ack, Configure-Nak and Configure-Reject) are used to establish and configure a link. Link-termination frames containing link termination packets (Terminate-Request and Terminate-Ack) are used to terminate a link, and link-maintenance frames filled with link maintenance packets (Code-Reject, Protocol-Reject, Echo-Request, Echo-Reply, and Discard-Request) are used to manage and debug a link.

Table 2 is the format of an LCP packet. The Code field denotes the type of LCP packet. For instance, 0x01 in the Code field shows that the LCP packet is a Configure-Request packet. A list of the Code field entries is shown in Table 3.

The Identifier field is used to match requests and replies to the peers. If this field is invalid, the packet is ignored without any action springing from its contents.

The Length field includes the Code, Identifier, Length, and Data field byte counts. Everything outside the limits defined by the Length field entry is treated as padding and thrown away.

The Data field contents are directly dependent on the Length field byte count. Also, the format of the Data field is controlled by the type of packet defined in the Code field. Typically, the Data field format follows the Type/Length/Data layout. A list of LCP option type field entries is shown in Table 4.

As all of these various types of frames are flying among peers, commands to configure and send the aforementioned packets and frames are issued depending on the conditions at a particular point in the execution of a phase. These packet-assemble and packet-send commands are generated by events, actions, and state transitions. This collection of events, actions, and state transitions (as they relate to PPP) are known as the option negotiation automaton.

Without going into lengthy discussion, the option negotiation automaton is a definition and model of how the states and state transitions must be made to follow the rules in implementing PPP. PPP code is written using the option negotiation automa-

Listing 1—*This is a dump snippet. The idea is to convey the fields and how they all play together with the help of encapsulation.*

```
AT
OK
ATZ
OK
Modem  Initialized

ATDT6334710
CONNECT  9600/ARQ/V32/LAPM/V42BI

**  Florida  Online  MAX-ROC9  **


login:edtp
password:
     Entering PPP Mode.
     IP address is 208.14.41.96
     MTU is 1524.

Sent: FF 03 C0 21  01 08  00 18  01  04  05 DC  02  06  00
 00  00  00  05  06
Got  LCP  Request
       FF 03 C0 21  01 01  00 3C  01  04  05 F4  02  06  00
 0A  00  00  07  02

Sent: FF 03 C0 21  04 01  00 0E  01  04  05 F4
Sent  LCP  Reject
Received  LCP  Config-ACK
Got  LCP  Request
Sent: FF 03 C0 21  02 02  00 12  01  04  05 F4  02  06  00
 0A  00  00  07  02
LCP  Config  Acked

PPP  established

IP  address  is  208.14.41.96


Sent: FF  03 C0 21 05  0A  00  04  00 FE  06 C3 51 D0  0E 29
 60  00  00  00
Terminate  Ack

NO  CARRIER
ATH
OK
ATZ
OK
```

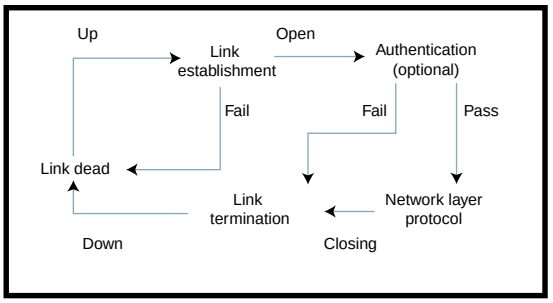


Figure 3—*I'm glad it doesn't use "dead" for low and "alive" for high in the logical sense.*

were indeed 24 bytes from the Code field in the data area to the byte (three bytes in from the end flag).

The next byte is the LCP Configuration Options Type field. The 0x01 shows that this is an MRU setting. The length of this MRU segment is 0x04 octets and the value of the MRU (maximum receive unit) is 0x05DC, or 1500 decimal. Again, following the Type/Length/Data layout and beginning with the next byte, 0x02, the async control character map is being negotiated and you're asking for no escaped characters with the four octets of zeroes. The 0x05 and 0x06 bytes that follow are the beginning of sending what is called the magic number. The magic number is an arbitrary number derived from any means that is as random as possible. The idea is to

1	Maximum-Receive-Unit
2	Async-Control-Character-Map
3	Authentication-Protocol
4	Quality-Protocol
5	Magic-Number
6	Reserved
7	Protocol-Field-Compression
8	Address-and-Control-Field-Compression

Table 4—*The magic number is akin to a boomerang in that it is thrown out to see if it returns.*

ton as the flowchart. Now that you're dangerous, let's look at a piece of PPP protocol and break down the fields to determine what went right and what went wrong in Listing 1.

BREAKING IT DOWN

Listing 2 is a code snippet that runs to produce the AT and ATZ modem commands. In terms of phase, you are at dead here. The modem initializes and the ISP is dialed. When the carrier detect is sensed, the establish phase is entered. In the meantime, the ISP asks for a login and password. This is not a PPP function. The Rabbit 2000 Core Module is assigned an IP address and given the MTU size.

The next line is where the real PPP fun begins. Let's break down the trace. The 0x7E is really there on the datascope, but the debug code in the Rabbit library doesn't display it. So far, everything is just as it should be. The flag byte is followed by 0xFF in the Address field and 0x03 represents un-sequenced frame transmission in the Control field.

The next two octets (there's Mozart again) define the protocol encapsulated in the packet. 0xC021 is the assigned number for LCP. The following fields are Data fields formatted according to the protocol called out in the protocol field (0xC021). Another look at Table 2 shows that, for an LCP packet, you should expect a single-byte Code field. This Code field is populated with 0x01. This is a Configure-Request packet. The next byte is the Identifier, which, in this case, is 0x08 followed by the two-octet Length field containing 0x0018. This trace doesn't show all of the bytes between flags, but I verified on the datascope that there

Listing 2—*I'm really showing you this so you can hook up your Rabbit 2000 Core Module to a modem.*

```
/*******************
external_modem.lib

A set of routines for controlling an external modem through a
 full RS232 link.
Default hardware is a core module using serial port C
Pin assignments for control lines are:
       DTR - PB6 (out)
       RTS - PB7 (out)
       CTS - PB0 (in)
       DCD - PB2 (in)
       RI  - PB3 (in)
       DSR - PB4 (in)
       TD - PC2 (out)
       RD - PC3 (in)

***************************/

/*** BeginHeader */
//#ifndef __DCSPPP_LIB
//#define __DCSPPP_LIB
#ifndef __EXTMODEM_LIB
#define __EXTMODEM_LIB

unsigned long extmodem_baudrate;
//current baud rate for communication with modem

//setup flow control for port C
#define SERC_RTS_PORT PBDR
#define SERC_RTS_SHADOW PBDRShadow
#define SERC_RTS_BIT 7
#define SERC_CTS_PORT PBDR
#define SERC_CTS_BIT 0

/*** EndHeader */

/*** BeginHeader ModemOpen */
int ModemOpen(unsigned long baud);
/*** EndHeader */

/* START FUNCTION DESCRIPTION
 ****************************************
```

(continued)

Listing 2–*continued*

```
ModemOpen
 <EXTERNAL_MODEM.LIB>

SYNTAX:                      int  ModemOpen(unsigned  long
 baud);

DESCRIPTION:  Starts  up  communication  with  an  external  modem

PARAMETER1:          The  baud  rate  for  communicating  with  the
 modem

RETURN  VALUE: 1  -  External  modem  detected
               0  -  not  connected  to  external  modem
END  DESCRIPTION
**********************************************************/

int  ModemOpen(unsigned  long  baud)
{
        extmodem_baudrate  =  baud;
        //check  DSR  line
        if(ModemReady())
        {
                serCopen(baud);
                serCflowcontrolOn();
                return  1;
        }
        else
        {
                return  0;
        }
}

/***  BeginHeader  ModemInit  */
int  ModemInit();
/***  EndHeader  */

/*  START  FUNCTION  DESCRIPTION
 ********************************************
ModemInit
 <EXTERNAL_MODEM.LIB>

SYNTAX:                      int  ModemInit();

DESCRIPTION:  resets  modem  with  AT,  ATZ  commands

RETURN  VALUE: 1  -  success
                                   0  -  modem  not  responding
END  DESCRIPTION
**********************************************************/

int  ModemInit()
{
        int  i;

        for(i  =  0;i  <  3;i++)
        {
                ModemSend("AT\r");
                if(ModemExpect("OK",  2000))
                {
                        ModemSend("ATZ\r");
                        if(ModemExpect("OK",  2000))
                        {
                                return  1;
                        }
                }
        }
        return  0;
}
```

send this and see what you get back. If your magic number is returned to you, then you are in a loop-back mode. On the other hand, if you get a totally different number than what you concocted and sent out, that probably means the link is ready for use. If the magic number is not negotiated, it consists of four octets of zeroes.

The next line after "Got LCP Request" is incoming from the peer at my ISP. Notice the differences in the MRU and async control character map entries. The Rabbit rejects the Configure-Request with a 0x04 (Configure-Reject) in the Code field. The ISP peer then says OK and sends a Configure-Ack with settings you can agree on. At this point, you should be OK with the negotiated values on your end, so you send a positive acknowledgement to the ISP peer. The PPP link is established.

OUT OF PAPER

The *Circuit Cellar* editorial staff has one of those long hooks they used to use to pull performers off stage. That means I'm out of paper, but I'm not done with dialup and PPP. I'll cover the next two phases, authenticate and network, next month. Until then, remember, it doesn't have to be complicated to be embedded.

Author's Note: I would like to thank Tamara Kaestner and Charlie Krauter for sharing their wealth of Rabbit knowledge.

Fred Eady has more than 20 years of experience as a systems engineer. He has worked with computers and communication systems large and small, simple and complex. His forte is embedded-systems design and communications. Fred may be reached at fred@edtp.com.

SOURCE

Rabbit 2000 Core Module
Rabbit Semiconductor
(530) 757-8400
Fax: (530) 757-8402
www.rabbitsemiconductor.com

Mike Baptiste

What's New on the Home (Automation) Front?

The Electronic House Expo

This month, Mike trades in his design tools for a pair of walking shoes and hits the show floor bringing us a report from the latest home automation show. From lighting systems to water sensors, the next generation of HA has arrived.

For most industries, tradeshows are an integral part of marketing products, and the home automation (HA) industry is no different.

Usually HA shows are part of a larger show dealing with security and alarm products. However, there is a set of shows, run by the folks at EH Publishing, that is completely dedicated to HA, the Electronic House Expo.

The EH Expo West began after the original Home Automation Expo was combined with the ISC security shows. The first EH Expo was not supposed to be until 2001, but given the high concentration of HA companies and installations on the west coast, the show was pulled in a year.

I manned the Creative Control Concepts booth and was able to check out all the new goodies and talk with vendors about the future of HA.

EMBEDDED IS HERE TO STAY

As the Microsoft juggernaut grew in the early '90s and PC prices plummeted, many people felt that embedded HA systems like the HCS-II were doomed. Just a few years later, your home could be controlled by a PC running software on Windows.

Although there are many PC-based HA packages out there, companies selling systems with embedded processors are doing well. Systems like the JDS Stargate, Homevision, HCS-II, and RCS CommStar are gaining ground.

Clearly there is no way of knowing what the future will hold, but more and more companies seem to be moving towards (or sticking with), embedded hardware for HA. Many customers still feel Microsoft Windows is too unstable to use for home control.

BATTLE FOR CONTROL

One area of HA that has been in flux for some time is how control data will be transmitted in the future. X10 devices still rule in the powerline carrier (PLC) arena because they are inexpensive and widely available. But, serious users want a more robust and reliable PLC transport. X10 is notorious for false triggers or modules that don't respond on command. You can spend hundreds of dollars on amplifiers and filters before X10 works reliably (a result of all the powerline noise in today's high-tech homes).

CeBus has been discussed for years, but is currently used only by a few companies for systems costing thousands of dollars. Now that Microsoft has adopted CeBus, a number of companies are stepping up to the plate to manufacture lamp modules, appliance modules, switches, and more.

In talking with other automation controller vendors, I found that they still view CeBus with a wary eye. The amount of overhead and programming required to handle it looks daunting. Also, the chipsets and interfaces are still expensive. Only time will tell.

Of course, there are other control methods on the horizon. Ethernet was a hot topic as vendors scrambled to get their controllers on the 'Net. Although some vendors think Ethernet should be used to communicate with I/O nodes and light switches, many vendors think the overhead and extra cost outweigh any benefits of using Ethernet to control everything.

Another technology is Bluetooth wireless. One of its standard modes is UART to wireless. Because many HA systems use simple serial protocols to communicate, Bluetooth should make it easy to add wireless capability to many of the RS-485 and RS-232 network modules. However, this depends on the price points for the Bluetooth chipsets and modules.

A-10

Advanced Control Technologies has developed what it believes is the answer to PLC noise problems. It calls the technology A-10. It is 100% backward compatible with current X10 devices, but when A-10 transmitters are combined with A-10 receivers, the technology shines.

The demo at this booth was impressive, with four PLC devices in an electrical box, a regular X10 module, a normal Leviton switch, a commercial Leviton switch with Intellisense and AGC, and an ACT A-10 switch. While an A-10 transmitter turned the units on and off, Advanced Control Technologies used a signal generator to inject noise onto the powerline.

As the noise increased, the normal X10 devices stopped responding or became intermittent and the A-10 switch continued to turn on and off. At one point there was 6 V of injected noise and the A-10 switch still responded (A-10 signals are usually 6 V, X10 devices normally put out 3 V). An oscilloscope display monitoring the powerline signals was so scrambled at those noise levels that the signal wasn't recognizable among the noise.

Real-world noise may be more random and troublesome than if it was injected by a frequency generator, but the demo still impressed me. If the cost of A-10 devices approaches that of quality X10 gear, this technology may be the sleeper hit in the PLC area.

The new A-10 switches are powerful, allowing for complete customization of what each pushbutton does. Features like group commands, Nite Lite mode, Two-way X10, and auto refresh are possible. The switches can also send out their X10 address when they are turned on manually so you don't need to ask the light if it was turned on.

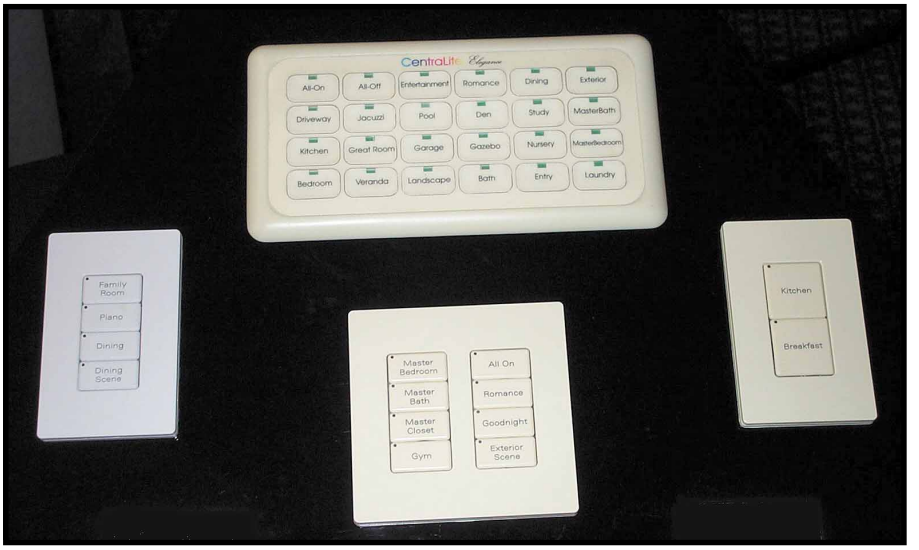


Photo 1—*The CentraLite Elegance lives up to its name. Both the main control panel and switches are attractive. Having the switches etched with what they control makes for a truly customized system.*

HARDWIRE

If you don't want PLC-controlled lighting, there are hardwired solutions available. CentraLite released a new lighting control system called Elegance (see Photo 1). CentraLite systems use centralized solid-state relay panels to control lights. Low-voltage switches with feedback LEDs are used to control each lighting circuit.

A central controller allows more powerful control features like dusk/dawn events and security flash. The controller has numerous serial ports that allow a central HA controller to be connected for remote control. An Ethernet port is also included to provide extensive control capabilities.

Elegance can control up to 192 circuits with standard dimming capacities of 10 and 15 A (1,200 and 1,800 W). The relays can dim incandescent, halogen, and most low-voltage loads. Up to 256 scenes can be created in the central controller.

The switches are custom engraved with whatever phrases you would like. Switches with one to four pushbuttons can be used in a single gang electrical box. The switches can be wired in a daisy- chain style to reduce the amount of wire that needs to run to the central controller.

The main control panel is mounted in a convenient location. It contains 24 pushbuttons with LEDs so you can get a quick view of the status of several lights in your home. There is also a wireless option allowing remote control of lighting via wireless key chains.

CentraLite also makes a smaller system called Ambiance. Both systems allow you to have extensive control of your lighting using a convenient wiring scheme. Note that, given such a scheme with lights wired directly to a central panel, Elegance is generally limited to new construction.

SENSORS

A common complaint among HA enthusiasts is the lack of attractive temperature sensors. Most are housed in small vented surface-mounted boxes. A company named Automated Environmental Systems (AES) came out with a line of sensors housed in standard Decora plates.

The TEMPPLATE line of sensors comes in various versions. In addition to a standard 10-mV/°F, there is a 19.6-mV/°F version with



Photo 2—*The TSPOT sensors on the left are about the size of a quarter, and the newer sensor prototypes are even smaller (on the right). The sensors can be painted so that they practically disappear. Their one-wire digital interface makes wiring these devices a breeze.*



Photo 3—*The WaterCop system utilizes wireless WaterHound transmitters to sense water leaks, which causes the WaterCop to turn off the water supply. The WaterCop can also be monitored and controlled by any HA systems with digital I/O.*

an 8-bit scale factor, and another that adds an extended range for monitoring temperatures below freezing.

There is a combination temperature and relative humidity (RH) sensor in a single gang Decora wall plate. This sensor has four outputs. Temperature is available in 10 mV/°F and 19.6 mV/°F and humidity is available in 10 mV/% RH and 19.6 mV/% RH.

This product is excellent for monitoring temperature and humidity in every room. Using these sensors, you can experiment with HVAC airflow settings to try to get uniformity throughout your home. There is also a version available that replaces the humidity sensor with a light sensor.

At first, 19.6 mV per unit seems strange, but it is useful. Any sensor that outputs in 19.6-mV steps that is fed to an 8-bit ADC with a 5-V range results in a 1-unit resolution. This allows you to monitor these sensors without having to convert the ADC reading to volts.

AES has a new device coming to the market called the TSPOT (see Photo 2). This device embeds a temperature sensor into a cylinder the size of a quarter that can be flush-mounted in any wall. When painted, the sensor is almost invisible.

In addition to a standard analog version, AES is coming out with a digital version of the TSPOT. This sensor conforms to the Dallas 1-wire standard, allowing you to easily wire all the sensors using a single run of CAT5 cable. The sensors behave just like DS1820 sensors, which makes them compatible with any system capable of reading DS1820s.



Photo 4—*The TR-36 graphic display thermostat may seem like overkill for a thermostat, but it allows a home automation system to display text messages. The large graphic display also makes for an easy-to-read thermostat with an attractive information layout.*

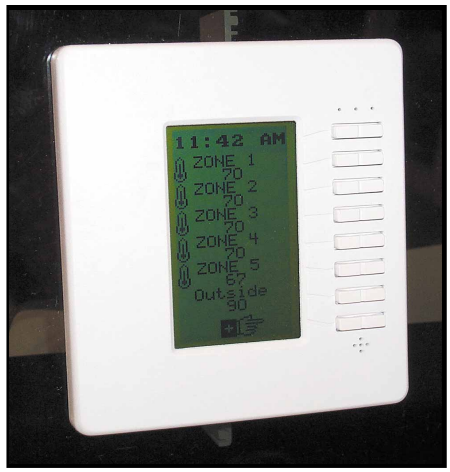


Photo 5—*This display allows you to develop attractive menus using graphic icons. The buttons on the side can be associated with the information you have on the LCD display. Each display fits into a standard single gang electrical box.*

POLICE YOUR WATER

As engineers, I'm sure you are familiar with Murphy's Law. Your philosophical side may wonder why, for example, the leak upstairs had to happen during your vacation instead of the weekend you spent at home. To keep the ceiling from caving in next time, maybe you can train the family pet to turn off the water in the event of a leak.

This option doesn't seem likely, so there's a better way. DynaQuip Controls created a device called WaterCop that connects to your main water supply line (see Photo 3). It contains a wireless receiver that controls a ball valve and you place tiny wireless water sensors, called WaterHounds, in the places where leaks are likely to occur (behind a refrigerator, etc.)

If a WaterHound senses water, it will send a signal to the WaterCop to turn off the main water supply. This ensures that the water will stop before significant damage is done. The WaterCop is available with 1/2", 3/4", and 1" ball valves.

The WaterCop is also available with hardwire I/O that allows you to monitor and control the WaterCop. This allows the HA system to act when a leak occurs (i.e., page or e-mail you via a macro). It also lets your HA system control the water supply to the home.

This allows you to shut off the water as part of a vacation program or if you have other types of water sensing in your system. Any dry contact relay

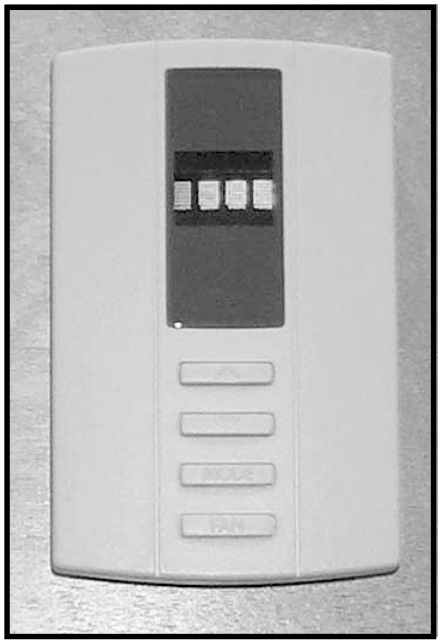


Photo 6—*This new thermostat combines temperature and humidity monitoring that can be accessed via an RS-485 network. The four character display displays the temperature and humidity readings and can also display short text messages.*

output will work. The WaterCop is powered by 120 VAC and the WaterHounds are powered with two AA batteries. The latter are addressable, which allows you to have multiple WaterCops in close proximity.

RCS

Residential Control Systems (RCS) has two new products aimed at improving the user interfaces in hardwire HA systems.

The first product is a thermostat with a graphical LCD display (see Photo 4). The TR-36 has the same features of RCS's other RS-485 thermostats (see "An RS-485 Gateway," *Circuit Cellar* 122). But, the TR-36 allows an HA system to display text messages on the thermostat. Local pushbuttons are included, providing a fully interactive HA user interface.

Because most thermostats are centrally located in homes, the TR-36 provides a great combination of a thermostat and HA interface. The TR-36 is RS-485-based, so it can be used with many HA systems including the RCS CommStar, JDS Stargate, and soon the HCS-II via the new Stat-Link interface.

In addition to the TR-36, RCS has a graphical LCD keypad that fits inside a single gang electrical box (see Photo 5). It has a graphical LCD display and seven pushbuttons along the side. You can create graphical icons to display along with text. The display is laid out so you can easily associate displayed graphics with keys on the right.

The unit also has three LEDs and a beeper that can be used to indicate various states or whatever the programmer desires. Again, the keypads are RS-485-based, which makes installation a snap. In the future I plan to add support to the HCS-II for these gems.

ELK

If you read "An RS-485 Gateway," *(Circuit Cellar* 122), you saw the RCS TR-15 communicating thermostat. ELK Products, in cooperation with RCS, has come out with a new thermostat based on the TR-15 design (see Photo 6). However, instead of a two-digit LCD display, it uses a four-character alphanumeric LED display. It also adds a humidity sensor to allow the thermostat to monitor both temperature and humidity.

Most communicating thermostats can be converted from temperature to humidity, but few allow both. This new thermostat can also display short messages by scrolling them across the four-character display. The thermostat is designed for use with ELK's Magic Module controller.

Of course, it is also RS-485-based, which means it can be used on other RS-485-based systems via a protocol converter. So, if you really crave combined temperature and humidity monitoring in an attractive case, just use a PIC to convert the ELK protocol to whatever protocol your RS-485 system uses!

CONVERGENCE

There has been a lot of talk about the convergence of the computer and the entertainment center. A few companies tried it, but the concept seems to have gone down in flames.

Still, DSC has come out with an easy-to-use convergence concept that seems to satisfy a number of customer needs. The Communiqué is a PBX system for use at home or in a small business (see Photo 7). It can handle eight CO lines and 24 extensions, more than enough for most homes. It also has built-in voicemail support with up to 24 mailboxes.

What sets the Communiqué apart is that it seamlessly adds support for home security and automation. Each phone has keys labeled Security and Automation. The Security button allows you to monitor and control your alarm system. The Automation button allows you to control X10 and (you might want to sit down for this), CeBus devices from any extension.

Each phone has 23 feature buttons that can be used to dial other extensions or execute sets of security or home automation commands. The buttons are also backlit with LEDs so they can flash to draw your attention, or indicate that an extension or line is in use. The phones are attractive and have backlit LCD displays, something that companies like Panasonic are just starting to add to their PBX sets.

By combining powerful security and home automation functionality in feature-rich PBX, the Communiqué may be a convergence idea that works.

OTHER EXCITEMENT

Unfortunately, I can only discuss a few products in the space I have. The EH Expo is held in Orlando every March and outside of Los Angeles every October. By the way, most HA companies that exhibit at the show get free tickets.



Photo 7—*The Communiqué PBX allows you to use your phone as a security and home automation controller. The inclusion of CeBus illustrates that it's ready for the future.*

Besides new products, there was plenty of industry-wide activity going on. The Home Automation Association has changed its name to the Home Automation & Networking Association to reflect the effect that home networking has had on the industry. There was also talk about trying to retire the phrase "home automation," and replacing it with "home control."

RS-485-based HA systems continue to thrive. The buzz around the show was that some vendors may band together to draft a standard control protocol for RS-485-based systems.

Well, that's about it. I hope to see you at the EH Expo in Orlando. You can catch us in booth 916.

Mike Baptiste earned a B.S. in Computer Systems Engineering from Rensselaer in 1992. After a seven-year "hiatus" working for a large telecommunications company, he returned to his roots working with embedded processors in home automation. He can be reached at baptiste@cc-concepts.com.

RESOURCE

Electronic House Expo
EH Publishing
"Ride the Wave to the Connected Home"
October 25–27, 2000
www.ehexpo.com

SOURCES

A-10
Advanced Control Technology Inc.
(800) 886-2281
Fax: (317) 337-0200
www.act-solutions.com

Elegance
CentraLite Systems, Inc.
(877) 466-5483
Fax: (334) 607-9117
www.centralite.com

TEMPPLATE and TSPOT
Automated Environmental Solutions
(714) 826-7908
Fax: (714) 826-2908
www.automated1.com

WaterCop
DynaQuip Controls Corporation
(800) 545-3636
Fax: (636) 629-5528
www.watercop.com

TR-36 and LCD display
Residential Control Systems, Inc.
(888) 727-4822
Fax: (916) 635-7668
www.resconsys.com

Temperature and humidity thermostat
ELK Products, Inc.
(800) 797-9355
Fax: (828) 397-4415
www.elkproducts.com

Communiqué
DSC Group of Companies
(888) 887-8730
Fax: (817) 281-9408
www.dscgrp.com

Home Automation & Networking Association
(202) 712-9050
Fax: (202) 216-9646
www.homeautomation.org

MICRO SERIES

David Tweed

Fundamentals of Second-Order Systems

Part 1: Where It All Began

Part 1 of 3

Understanding second-order systems isn't a mystery, but with so many options and applications, it's hard to get an understanding of the principles. So, Dave kicks off this series by explaining some of the basics.

The second-order system is common in nearly all branches of engineering—car suspensions, robotic servos, audio filters, to name a few—and so, a good understanding of it can be a powerful tool in your arsenal. But, the approach taken in many engineering schools discourages students and leaves them with an understanding that associates black magic with the topic.

In this series, I'll dispel that notion by demonstrating how common second-order systems are and by showing how analogies can be drawn among them. Yes, I'll get into mathematics, even calculus, but I promise that it will be painless. And when you notice how widely applicable the math is, you won't mind at all.

In Part 1, I'm going to talk about the basic second-order system in terms of a physical model (mass-spring-damper) and an electrical model (LCR circuit). You'll learn the equivalencies between these two models and derive the basic mathematics that describe them.

Part 2 will expand the discussion to the use of second-order systems as filters and then move on to cover servomechanisms. You'll learn how negative feedback affects the behavior of the system and extends the mathematical model. Other implementations based on op-amps and DSP (digital signal processing) will be discussed, too.

Part 3 will bring the discussion home with a detailed analysis of some real-world servomechanisms such as a mechanical positioner for a robot and an electronic phase-locked loop. I'll discuss how to predict their performance. How unavoidable nonlinearities affect the behavior of the system and ways to mitigate their effects will be covered.

WEIGHTS, SPRINGS, AND DAMPERS

In general terms, a second-order system is any system that has a measurement that can change. It stores energy related to that change (restoring force), stores energy related to the rate of change (inertia), and dissipates energy and comes to a stop (resistance). Nearly every textbook about this subject starts by discussing the physical system of mass, spring, and damper, so I'll start there.

Figure 1 shows the arrangement; a movable mass is attached to both a spring and dashpot (shock absorber), and the other ends are fixed. The vertical position of the mass is the changeable measurement. The spring provides the restoring force and stores potential energy in proportion to how far the mass is moved from its equilibrium position. The mass stores kinetic energy in proportion to how fast it is moving. The dashpot dissipates energy (turns it into random heat) at a rate proportional to the velocity of the free end. Assume that all of the friction in the system is embodied in the dashpot.

The mass's important parameter is, well, its mass in units of kilograms. (I'll stick to the metric system throughout this series because it re-

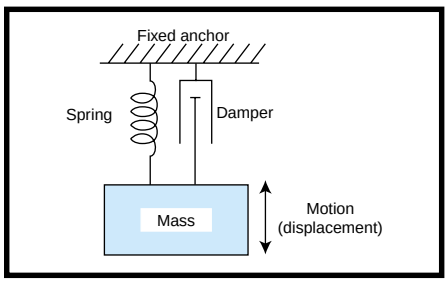


Figure 1—*The basic second-order mechanical system consists of a spring, mass, and damper, or dashpot.*

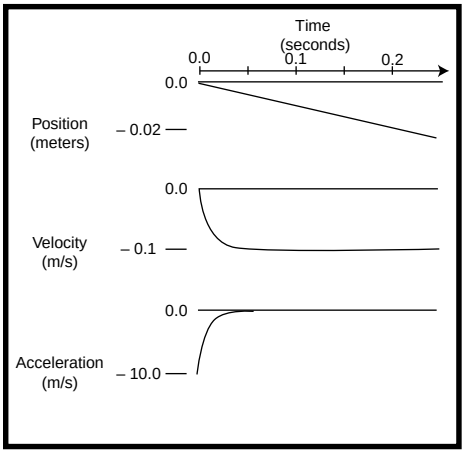


Figure 2—*A system of just a mass and a damper quickly reaches a terminal velocity, where the acceleration has decayed to zero.*

quires fewer conversion constants.) In the earth's gravity field, which has an acceleration of roughly –10 m/s^2, it takes a force of 10 newtons (N) to suspend a mass of 1 kg in midair.

The spring is characterized by its spring constant (K_s), which indicates how much restoring force it develops for a given amount of displacement. The metric units would be newtons per meter. The restoring force is in the opposite direction relative to the displacement; hence, if you move the mass downward, the spring's force is upward, and vice versa.

Assume the spring is –100 N/m. If you hang your 1-kg mass on it, the spring will need to develop a 10-N upward force to counteract the –10-N (downward) force of gravity. As a result, it will stretch by:

$$\text{displacement} = \frac{\text{force}}{K_S} = \frac{10\ \text{N}}{-100\ \text{N/m}} = -0.1\ \text{m} = -10\ \text{cm}$$

The dashpot also develops a negative force, but this force is proportional to the velocity of the free end rather than its position. The dashpot is characterized by the constant K_d in newtons per meter per second. The definition of "work" is a force operating over a distance. It has the same units as energy, which are joules (J), and 1 J is defined as a force of 1 N operating over a distance of 1 meter, or 1 newton-meter.

The work associated with moving the free end of the dashpot against its force is turned into randomized heat and is essentially lost to the system. This is in contrast to the spring, which stores that work as potential energy and returns it to the system when the movement reverses direction. Let's play with different values of K_d and determine what effect this has on the response of the system.

Imagine the dashpot filled with thick, syrupy oil. It requires a great force to move it even slowly; it has a high K_d, say, 10 N/m/s. If you hang the 1-kg mass from the dashpot, it quickly reaches a terminal velocity of –0.1 m/s. The work, or energy, needed to move the dashpot comes from the fact that the mass is descending in earth's gravitational field. Terminal velocity occurs when the two forces balance; the 10 N of gravitational force on the mass is balanced by 10 N of upward force from the dashpot when the velocity is –0.1 m/s.

A SIMPLE SIMULATION

Let's put together a mathematical simulation. I use MathCad because it makes it easy to see the math and graphics at the same time, but you can use a spreadsheet (see "Simulating in a Spreadsheet" sidebar) or your favorite programming language.

The general technique is simple: set up three variables that represent position, velocity, and acceleration, set their initial values, and write simple equations that update the three values after a short increment of time. After a few hundred (or a few thousand) timesteps, you can graph the evolution of the system by graphing the three values. In MathCad, use a vector to hold the three values so that you write a single equation to do the updates for each timestep.

Start by entering the system parameters (gravitational acceleration, mass (M), and damper parameter, respectively):

$$G = -10 \times \frac{\text{m}}{\text{s}^2}$$

$$M = 1 \times \text{kg}$$

$$K_d = -10 \times \frac{\text{N}}{\left(\frac{\text{m}}{\text{s}}\right)}$$

Also, set your timestep and number of steps and set up an index variable. The following equations are timestep, number of steps, and index variable, respectively:

$$dT = 0.001 \times \text{s}$$

$$N = 200$$

$$I = 1 \ldots n$$

The next step is to enter the initial conditions (position, velocity, acceleration, respectively):

$$p_0 = 0.1 \times \text{m}$$

$$v_0 = 0 \times \frac{\text{m}}{\text{s}}$$

$$A_0 = G$$

Next, enter the expressions that control updating. Position is easy, it gets incremented by the previous velocity value multiplied by the timestep. Similarly, velocity gets incremented by the previous acceleration value multiplied by the timestep. Acceleration is trickier, it's the result of the net sum of the forces (in this case, the constant acceleration of gravity and the damper force that is proportional to the velocity) divided by the mass. Use vector notation to ensure that all three values get updated at each timestep before moving on to the next, putting the position on top, velocity in the middle, and acceleration (a) on the bottom:

$$\begin{bmatrix} p_i \\ v_i \\ a_i \end{bmatrix} = \begin{bmatrix} p_{i-1} + dT \times v_{i-1} \\ v_{i-1} + dT \times a_{i-1} \\ G + \frac{Kd \times v_{i-1}}{M} \end{bmatrix}$$

Figure 2 shows the results, with the mass quickly reaching a constant terminal velocity when the force of the dashpot balances that of gravity. You can also think of this in terms of energy flow; terminal velocity occurs when the work being put into the

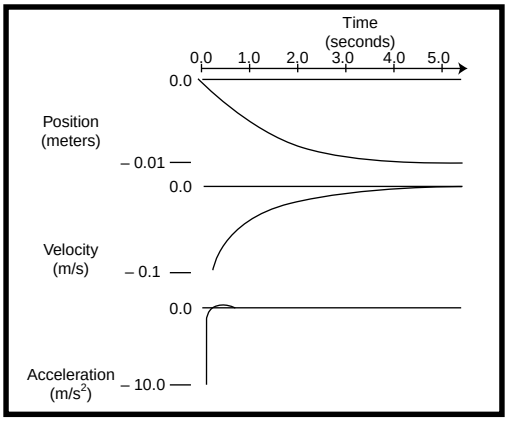


Figure 3—*An overdamped system responds to disturbances with a slowly decaying exponential movement.*

Physical system		Electrical system	
Item	**Units**	**Item**	**Units**
Position of mass	Meters	Capacitor charge	Coulombs
Velocity of mass	Meters/second	Current	Amperes (coulombs/second)
Physical force	Newtons	Electromotive force	Volts
Physical work	Newton-meters	Electrical work	Volt-coulombs
Mass	Kilograms	Coil	Henries
Spring	Newtons/meter	Capacitor	1/farads (volts/coulomb)
Dashpot	Newtons/meter/second	Resistor	Ohms (volts/coulomb/second)

Table 1—*There is a direct analogy for each parameter of the physical system in the electrical system.*

system by the gravity field equals the energy being dissipated by the dashpot as heat. Remember this point, because I'll come back to it later.

Now, let's go back to the system that includes the spring. It has an equilibrium point defined by where the weight of the mass is balanced by the upward pull of the stretched spring. If you lift up the mass from this point and then release it, it will gradually return to that equilibrium point. The acceleration of the mass is no longer constant, because as it moves down, the spring exerts an increasing upward force that partially cancels the force caused by gravity.

The other upward force on the mass comes from the resistance of the dashpot. The work being done on the mass by the spring (and gravity) must equal the work being done on the dashpot by the mass and getting dissipated. But, the dashpot's force is proportional to the velocity of the mass and the spring's force is proportional to the position of the mass. The net force on the mass is proportional to its acceleration.

So, the position, velocity (time derivative of the position), and acceleration (time derivative of the velocity) of the mass decay to zero in proportion to each other. One curve whose derivative varies in proportion to itself is the exponential (the general shape of the path followed by the mass).

To see the effect of adding the spring, add its parameter at the top of the simulation you did previously and modify the updated equation to calculate the spring parameter:

$$K_s = -100 \times \frac{N}{m}$$

$$\begin{bmatrix} p_i \\ v_i \\ a_i \end{bmatrix} = \begin{bmatrix} p_{i-1} + dT \times v_{i-1} \\ v_{i-1} + dT \times a_{i-1} \\ G + \dfrac{K_d \times v_i - 1 + K_s \times p_{i-1}}{M} \end{bmatrix}$$

You'll need more steps in order to see everything else, so change *N* to 5000 (see Figure 3).

RESONANCE

Now imagine a dashpot that's filled with only air with K_d of 2. If you change the parameter in the simulation and rerun it, you get the behavior

shown in Figure 4. Because the dashpot provides so little force, the mass has quite a bit of momentum built up when it reaches the equilibrium position, so it keeps going until the increasing force from the spring becomes enough to stop it.

However, now it's too low, so it starts back in the other direction. This process keeps repeating at a regular rate, called the natural, or resonant, frequency of the system. A system configured this way is said to be underdamped. The oscillations do die out eventually, because the dashpot still is dissipating energy from the system.

The resonant frequency (F_0) is primarily determined by the mass and spring (although, as I'll explain later, the dashpot does affect it). A higher K_s will raise the frequency and a larger mass will lower it. Ignoring K_d for the moment, the resonant frequency is:

$$F_0 = \frac{1}{2\pi}\sqrt{\left(\frac{|K_s|}{M}\right)}$$

With the values just mentioned, this equals 1.592 Hz. In fact, you can see about eight cycles in the 5-s elapsed time of the simulation.

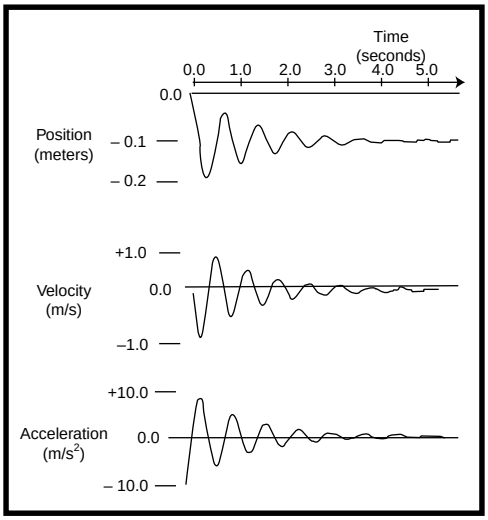


Figure 4—*An underdamped system oscillates in response to disturbances at a particular resonant frequency. This oscillation dies out exponentially.*

You can imagine a case somewhere between these two extremes of K_d. At some point, as the resistance of the dashpot is gradually decreasing, the system's response is just about to become periodic. In other words, there's a definite boundary between the overdamped and underdamped regimes. When the system is in this configuration, it is said to be critically damped. I'll leave this subject for now, but I'll return to it later.

COILS, CAPACITORS, AND RESISTORS

Now let's take a look at the canonical electrical second-order system, which consists of a coil, capacitor, and resistor (see Figure 5). The coil is equivalent to the mass, the capacitor is equivalent to the spring, and the resistor corresponds to the dashpot. In fact, for every parameter or quantity in the physical system, there's a corresponding one in the electrical system (see Table 1).

Start by defining the electrical equivalent of position of the mass to be charge (on the capacitor), which is measured in coulombs. This means that the electrical equivalent of velocity, or meters per second, is coulombs per second, or amperes (A). This seems natural enough.

Next, define the electrical equivalent of mechanical force, or newtons, to be electromotive force, or volts. Using both of these definitions, the mechanical concept of work = force × distance leads to the electrical concept of work = volts × coulombs. Following this to its natural conclusion, power is defined as work per unit of time. If you write power = volts × coulombs/seconds and note that cou-

Simulating in a Spreadsheet

Even if you don't have a mathematical modeling software package like MathCad, you can do this type of simulation with a spreadsheet program. I'm going to briefly outline the general technique, which should work with nearly any spreadsheet program.

The idea is to dedicate one column of the spreadsheet to each variable in the simulation. In the top row, label a column for each variable. Put the initial value of each variable in the next row, below its label. It's a good idea to have a column that represents time, with an initial value of zero. The other columns will represent acceleration, velocity, position, and so on.

After you have written the mathematical expressions that update the variables, enter them as formulae into the third row of the spreadsheet and use the values of the system variables from the row above. The update formula for time is simply the previous value plus a constant that represents the time interval for each step of the simulation.

Here's the trick: using whatever mechanism your spreadsheet uses, select the cells in the third row and replicate them downward, using as many rows as the number of timesteps you want to simulate. Make sure that the cell references for system variables remain relative, always using values from the row immediately above.

Now you should see a huge field of numbers that show the evolution of the system over time. If your spreadsheet supports the generation of graphs (nearly all of them do), you can graph the columns representing the system variables against the column representing time, and you should get graphs similar to the ones that MathCad generates.

If you want to change the initial conditions, modify the values in the second row of the spreadsheet. If you want to change the update formulas, do that in the third row, but then you'll need to replicate the cells below the one(s) you change again.

You can avoid this second replication if you know how to generate absolute as well as relative cell references in your formulae. Instead of putting constants into the formulae, replace them with absolute references to cells near the top of the spreadsheet. Then if you want to change the parameters of the simulation, you can just change the contents of those cells and the rest of the spreadsheet will update itself.

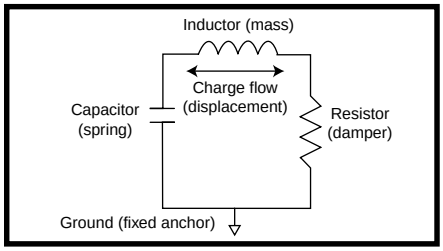


Figure 5—*The basic second-order electrical system consists of a capacitor, coil, and resistor.*

lombs/seconds = amperes, you get power = volts × amperes, which we're all familiar with.

The remaining equivalencies among the mechanical components and their electrical counterparts may look strange at first, but let's work through the details.

The coil, whose inductance is measured in henries (H), is equivalent to the mass, which is measured in kilograms. Using the relationship F = M × a, kilograms can be thought of as F/a, or newtons per meter per second squared. By definition, a coil is a device that develops a voltage across itself in proportion to the rate at which the current is changing, which means that henries can be thought of as volts per ampere per second, or volts per coulomb per second squared. You end up with the corresponding fundamental units you started with in corresponding positions in the two expressions.

The electrical capacitor is equivalent to the mechanical spring. You measured K_s in units of newtons per meter. In a capacitor, Q (charge) = C (capacitance) × V (voltage), so C = Q/V. In order to establish the equivalence between K_s and the capacitor, you need to use 1/C, or units of 1/farads. In effect, a stiff spring (large K_s) corresponds to a tiny capacitor (low capacitance).

Finally, the electrical resistor corresponds to the mechanical dashpot. Using the relationship V = I × R, ohms can be thought of as V/I, or volts per coulomb per second. Again, the units match up in the expressions for the two systems.

Everything I said about the mechanical system also applies to the electrical system when you substitute the correct quantities and units. The first simulation you did corresponds to a circuit with just a coil and resistor. The force of gravity is equivalent to putting a battery in series, therefore, you can see how the terminal velocity of the mechanical system corresponds to the VDC current that flows in the R–L circuit.

For the second simulation, you put the capacitor back in the circuit. It now limits the flow of charge in the same way that the spring limits movement. And in the third simulation, lowering K_d is equivalent to lowering R, and then the electrical circuit rings. When you change the units in the formula for the resonant frequency, it becomes:

$$F_0 = \frac{1}{2\pi}\sqrt{\left(\frac{1}{L \times C}\right)} = \frac{1}{2\pi \times \sqrt{L \times C}}$$

SHAKE, RATTLE, AND ROLL

The next question is, what happens if you apply a continuous stimulus to either the physical system or electrical system, instead of just nudging it and watching what happens? For a number of reasons, the sinewave is the simplest form of continuous motion, so let's study that. Fourier states that you can make any other periodic motion by adding sinewaves, which makes it a good place to start.

Start with the physical system. You can apply a stimulus by making the fixed anchor movable and then either applying a force to it or moving it a specific amount. In either case, moving the anchor requires that an entity outside of the system do work—an energy input to the system. The only energy output from the system still occurs via the damper (or resistor). Therefore, it seems obvious that the system response will move toward a condition in which the dissipation of the damper equals the input energy.

Clearly, both the spring and dashpot will attempt to transfer any motion of the anchor to the mass, whose inertia will resist it, causing the spring and dashpot to change length. If you do this slowly, the spring and dashpot don't change length by much; instead, the entire system, including the mass, simply moves up and down in sync with the anchor.

On the other hand, if you do it quickly, the inertia of the mass holds it nearly still and all of the motion is

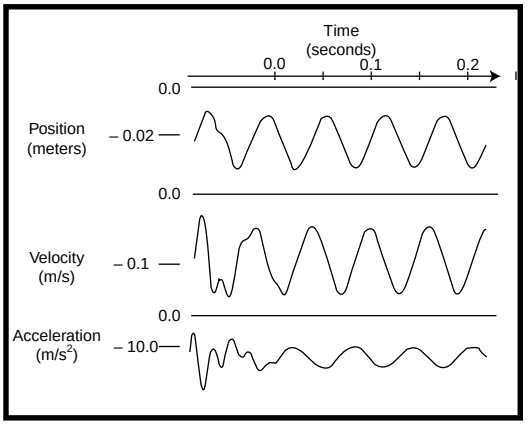


Figure 6—*The system response under continuous stimulation shows both a transient part and steady-state part.*

$$F = 0.5 \times Hz$$

$$Force(t) = \sin(2 \times \pi \times F \times t)\,N$$

$$p_0 = -\frac{G \times M}{K_s}$$

$$\begin{bmatrix} p_i \\ v_i \\ a_i \end{bmatrix} = \begin{bmatrix} p_{i-1} + dT \times v_{i-1} \\ v_{i-1} + dT \times a_{i-1} \\ G + \dfrac{K_d \times v_{i-1} + K_s \times p_{i-1} + Force(dT \times (i-}{M} \end{bmatrix}$$

taken by the spring and dashpot. Between these two extremes is where more interesting things happen.

Applying a force to the anchor is equivalent to a series voltage source. Actually moving the anchor is equivalent to using a current source in parallel with the inductor. The analogy is that if the mass (coil) is large, it moves (carries current) hardly at all, while all of the motion (current) is taken by the spring (capacitor) and damper (resistor).

If the frequency of the voltage or current source is low, the voltage (and charge) on the capacitor will track that of the source closely. But if the frequency is high, it will change little.

In an overdamped system with a large K_d, most of the force applied to the anchor will be coupled with the mass via the damper rather than the spring. The electrical analogy is that the high R, rather than the impedance of the capacitance, limits the movement of charge through the inductor and the rest of the circuit.

In the underdamped system, different things happen depending on whether the applied frequency is less than, equal to, or greater than the system's resonant frequency. Let's update the simulation to incorporate this external force.

An external force applied to the anchor point gets added to the other forces acting on the mass. Set the initial position equal to the equilibrium position to simplify the interpretation of the results (see Figure 6).

The first three equations are frequency, anchor force, and initial position, respectively:

You can see irregularities in the graphs near the beginning, before things settle down to a regular pattern. Looking at the acceleration graph, you can see that the transient part has a frequency component that is similar to the natural frequency of the system calculated earlier and an shape that resembles the graphs in Figure 4. This resonance is excited by the fact that the simulation applies the acceleration suddenly at time zero (a phenomenon known as jerk). After this transient response dies, the only frequency component present is the 0.5 Hz of the stimulation.

UNTIL NEXT TIME

You've observed the analysis of both the transient and steady state responses of some simple second-order systems. It's been an interesting mental exercise, but I'm sure you're wondering how all of this can be applied in the real world. Well, don't worry, that's what I'll get into in Part 2.

Dave Tweed is an independent consultant. He has been developing hardware and real-time software for microprocessors for many years. His system design experience includes computer design from supercomputers to workstations, digital telecommunications systems, and the application of embedded microcomputers and DSPs. You may reach him at dtweed@acm.org.

SOFTWARE

The MathCad files are available on the *Circuit Cellar* web site.

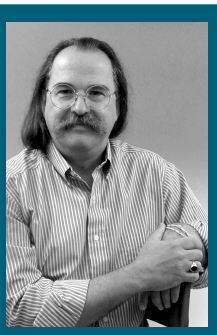
SOURCE

MathCad
Mathsoft, Inc.
(617) 577-1017
Fax: (617) 577-8829
www.mathsoft.com

Jeff Bachiochi

Where's Waldo?

Pinpointing Location by Interfacing with a GPS Receiver



Jeff's not really looking for a character with a red and white hat, but with today's GPS receivers, pinpointing locations is only getting easier. Carmen Sandiego, beware!

The term GPS (global positioning system) has lost its effect to raise eyebrows. There was a time when GPS was strictly a tool for guiding missiles toward their intended targets. The government, in its fundamental wisdom, allowed the military to add a secret fudge factor to GPS signals that produced errors in the accuracy of the system. If other countries used GPS data to aim their own missiles, they wouldn't be hitting their targets. Of course, this means they would hit somewhere else, potentially in a heavily populated area. This might be fine from a military standpoint, but probably wouldn't sit well with the people in that area.

Hand-held GPS devices have been in sporting goods stores for years. These early models couldn't actually tell you your location, but merely allowed you to get back to where you started. Many of today's hand-held devices also contain a map for true stand-alone position identification. Recently, GPS was introduced in conjunction with some of the higher priced automobiles (e.g., GM's OnStar). Although more than just a GPS, the feature to pinpoint a car's location for emergency service can be quite comforting, even if it comes with a monthly charge.

Those of us who are forced to look in the aftermarket for these accessories will find vehicle-mounted GPSs complete with downloadable area maps. You can download business, educational, governmental, recreational, and religious points of interest in addition to the road and highway systems. Most systems consist of hardware and software. The Street Atlas mapping software accepts input from the hardware GPS. Latitude and longitude are entered automatically and the correct map appears on the screen. The GPS can't indicate a true direction unless you're moving. Direction is calculated based on where you are and where you've been.

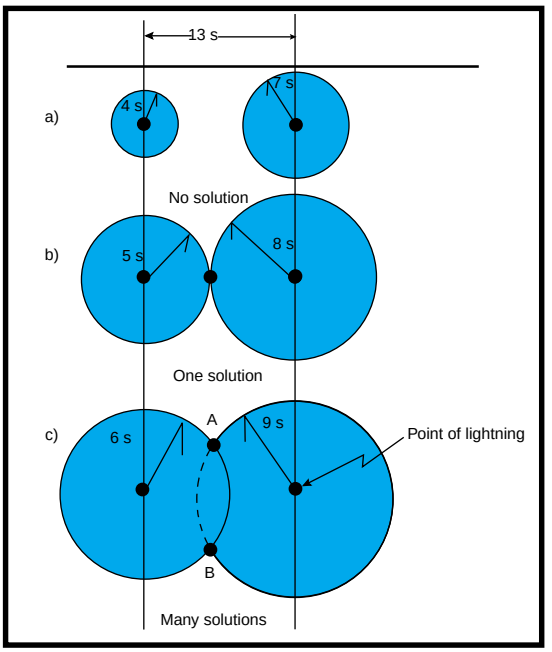


Figure 1—*The circles represent your possible position when you count the time between lightning flash and thunder crack.* **a**—*This has no intersecting solution because the times are less than the actual distance between the two lightning strikes.* **b**—*If you happen to be at just the right distance, circles just touch and produce a single-point solution.* **c**—*Most likely, circles will overlap giving two points on a flat earth. In 3-D reality, the circles are actually bubbles where their intersection is a circle of possible solutions.*

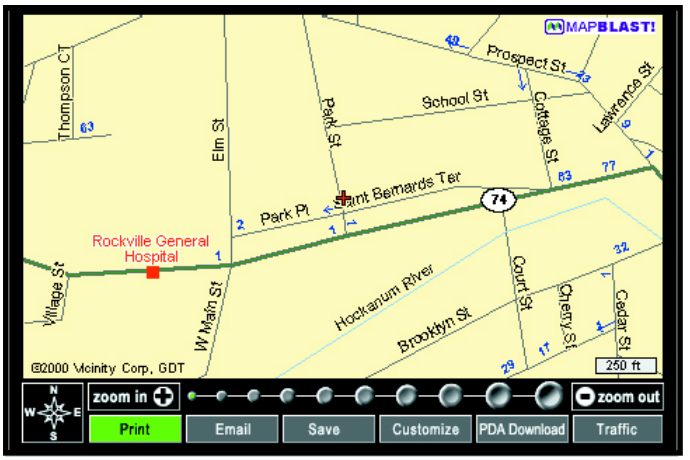


Photo 1—*This location map, based on longitude and latitude, is courtesy of MapBlast (www.mapblast.com).*

GPS RECEIVER

Stand-alone GPS receivers are available today. Just check the pages of *Circuit Cellar* for manufacturers. In fact, if you want to know the details about GPS data output and how to calculate distances on bearings from any two coordinate points on earth, see Jeff Stefan's article, "Navigating with GPS," (*Circuit Cellar* 123). This month, I plan to go into why it works and how you can use this new technology. First, I want to look at what it takes to pinpoint a location.

Each of the 24 GPS satellites orbits the earth in about 12 hours. The GPS satellites are arranged in six shells of four satellites per shell. Each group of satellites within a shell is spaced equidistant from one another in their orbit at a specific shell altitude. The orbits of each shell are offset at 60° covering a full 360°. This arrangement allows at least five satellites to be seen from any point on earth.

For the moment, let's think about a satellite as a thundercloud. If you count how many seconds it takes for you to hear the crack of thunder after you see the flash, you can figure out how far away you are from the cloud. If the cloud is stationary and you know exactly where it is, you could be anywhere around the cloud, a diameter equal to the distance the sound takes to reach you. If you know the location of two stationary clouds and count the times for each, you can draw circles around each cloud with radii equal to the times you counted. This would produce two intersecting circles. Figure 1a shows an impossible solution, where the times could not be real, and no intersection is possible. Figure 1b shows a solution where the circles intersect at only one point. This pinpoints your location and you must be on a line directly between the two clouds. Figure 1c is most likely the solution, but you still cannot be sure if you're at location A or B (in reality, there are many more points to this solution).

I drew circles to indicate the distance. Think of these circles as three-dimensional spheres, not just two-dimensional circles. This means that in Figure 1c you might be anywhere the two spheres intersect, which is a circle around the axis of a line between the two clouds. On a flat surface point, A and B are the only solutions, however, in the GPS world you may be at any altitude, so other solutions are possible. Using the known position of three clouds can narrow it to two points (eliminating one if you know elevation), however, it can take four to narrow it down to a single solution in free space.

This example of the GPS system is like a snapshot in time. With all the satellites orbiting above, you must first be able to figure out exactly where they are to be able to figure out where you are. Each satellite transmits two microwave carriers; one is modulated with a 1023-bit sequence that repeats every millisecond. A GPS receiver looks for these sequences. When identified, it uses the start of sequence as a time of arrival (TOA) value. The TOA is an indication of distance because you know how long it takes the microwaves to travel through the atmosphere.

To help identify the satellite's location, the carrier is also modulated by navigation and system data. From this data, the GPS receiver can calculate the satellite's earth-centered, earth-fixed x, y, and z (*ecef* xyz) coordinates at a particular time of day UTC (Universal Time Coordinated), as shown in Figure 2. If you only knew exactly when the satellite started its transmission, you could calculate the distance by subtracting the two times. The GPS receiver can't get synchronized in time with this single satellite. But, by "seeing" other satellites and calculating their exact *ecef* xyz positions, the GPS receiver can adjust all of the relative TOAs because they are synchronized. With four satellite *ecef* xyzs and ranges to each satellite, the GPS receiver can then calculate its *ecef* xyz and translate it into latitude and longitude.

APPLYING GPS

At this point, you can begin to appreciate the amount of computational power a GPS receiver is capable of. I have a TF10 GPS receiver that uses a GPS-1MAL active antenna from Laipac Technology. The default TTL serial output is all six National

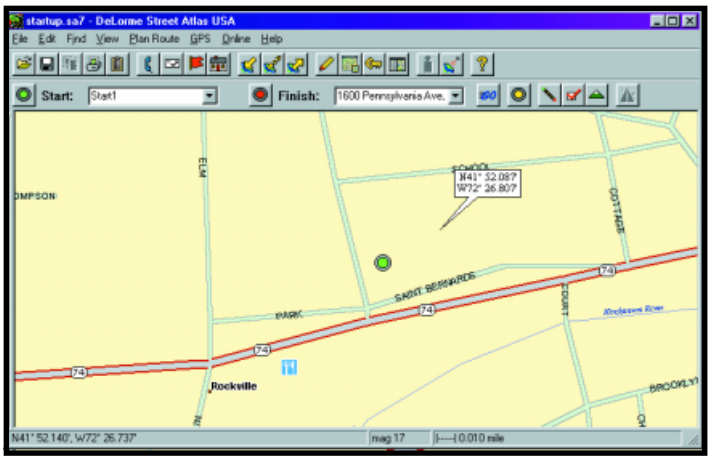


Photo 2—*This location map, based on longitude and latitude, is courtesy of Street Atlas (www.streetatlasusa.com).*

Marine Electronics Association (NMEA) messages. Every piece of data I'm interested in is contained in the RMC. The RMC data format is an ASCII string consisting of this information: message ID, UTC, status, latitude, N/S indicator, longitude, E/W indicator, speed, course, date, magnetic constant, and checksum<CR><LF>.

I found myself thinking about several situations where this information could be helpful. This project takes the GPS receiver's ability to pinpoint locations along with the cell phone's ability to communicate with a home computer in a wireless fashion to create a wireless vehicle tracker. This project could be used to locate a stolen vehicle or to simply track a number of delivery vehicles at any time.

Take a look at Figure 3. A PIC micro is used to absorb the data output from the GPS. To simplify the software, I used the PicBasic Pro Compiler from microEngineering Labs. So, I could use any port pin for serial transmission and reception. This has advantages (can use true or inverted data levels) and disadvantages (software serial means no interrupts). However, you don't need multiple functions happening at once, so this approach should work well.

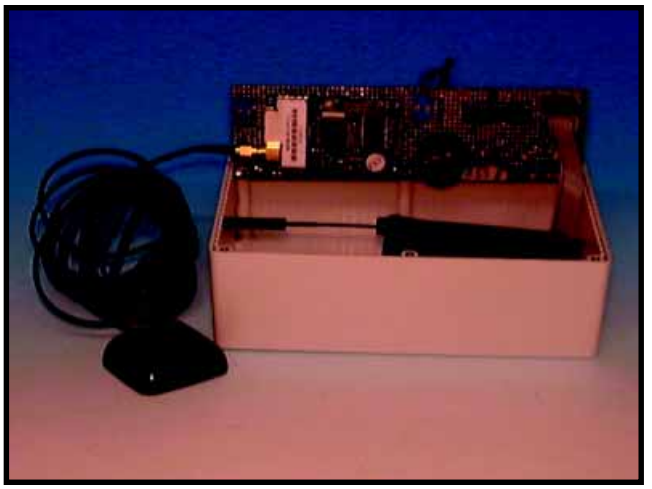


Photo 3—*The project fits into a small enclosure that can be hidden in your trunk. But, the magnetic GPS antenna works best where it has a full view of the sky.*

Follow the program flow shown in Figure 4. This project has five basic operations—wait for an alarm, monitor GPS output, dial the modem, send a message, and determine what to do next. The alarm is an input port bit with internal pull-ups enabled. A simple switch to ground (or a logic 0 signal) activates the alarm and causes the program to exit this loop.

The 4800-bps TTL GPS output comes straight into the PIC. From my office window, the GPS can pick up enough satellites to make a positive location fix in less than 1 min. The software monitors the serial datastream until a $GPRMC message ID match occurs. This subroutine returns indicating success or failure. If no serial data is received, then the GPS is reset and the routine is called again. If the data is bad or no fix has been determined yet, the routine is called again. If a fix has been estab-

lished, then the program continues.

If the modem is already connected (i.e., in continuous location reporting), the dialing routine is skipped. However, because this is the first time through here, the PIC uses the AT command set to dial a stored number. This is a local number and it covers a wide area. If the vehicle enters an area outside the local cell exchange, the number needs to be changed to include the area code. I didn't bother to code in an alternate number to use when you can't connect to the local number; this and some other advanced features are left to the experimenter. There are a number of responses the modem can return when a number is dialed. The connect response (actually C) allows the program to proceed. All other responses allow the program to hang up the modem to try dialing again.

After a connection is made, a message is sent to the modem (and the called computer). Here's an excerpt from that message:

```
FTB126–auto locator–Jeff Bachiochi–
    10/18/2000
UTC = 18:04:56 latitude = N41 52 04
    longitude = W072 26 52
MPH = 0 last heading = 150° 10/19/00
```

The time of day is based on GMT. You need to adjust the hours depending on your earth position. Knowing the longitude, you might even wish to adjust this automatically, although this may get complicated considering daylight-savings adjustments.

When the message has been sent, the program will make one of three branches. If the continuous input is strapped low, the modem connection is maintained indefinitely and the program loops back to obtain a new GPS fix. If the continuous input is high and the WAIT_HOUR input is strapped low, the modem hangs up and the program goes into a count-off-every-hour loop. After an hour, the program loops back to check the alarm input again. If both continuous and WAIT_HOUR inputs are high, the program immediately jumps back to check the alarm input again.

The difference between this mode and the continuous mode is that the modem hangs up here and the alarm is monitored. In continuous mode, when the alarm triggers, the modem remains connected and the alarm input is no longer monitored. Choose the appropriate mode carefully, because cell phone use adds up. Of course, if your car was stolen, you'd want to get a position fix ASAP. Talk time is cheap compared to a lost or damaged vehicle.

CAR 54, WHERE ARE YOU?

If you go to MapBlast's web site (www.mapblast.com/myblast/index.mb) and click on the advanced search, you'll be presented with a screen that allows you to enter latitude and longitude coordinates. These need to be in signed degrees with decimal minutes and seconds formats. My coordinates of 41 52 04 N/072 26 52 W translates to 41.8677/–72.4477.

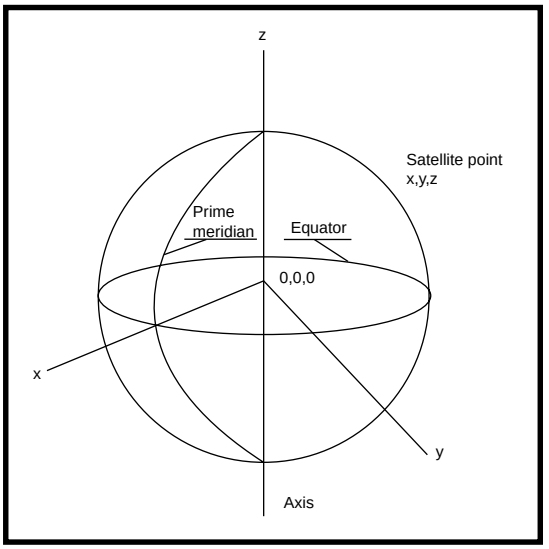


Figure 2—*All x, y, and z points are based on the center of the earth (0, 0, 0). The x axis starts at the intersection of the equator and the prime meridian and extends through the center of the earth. The y axis starts at the intersection of the equator and the 90° prime meridian and extends through the center of the earth. The z axis begins at the North Pole and extends through the center of the earth. A satellite's position can be defined using* ecef *x, y, and z coordinates at any point in time.*

Plugging in these values, you get a map (see Photo 1) that identifies the location as the corner of Park St. and Saint Bernard Terrace, which is *Circuit Cellar*'s home base. If you have a Street Atlas program, you can likewise plug in longitude and latitude to get a map similar to that in Photo 2.

WRAP IT UP, I'LL TAKE IT!

My first thought was to include a couple of regulators with logic inputs, U2 and U3, as in Figure 3. To reduce current draw, I would turn off the devices until they were needed. At an operating current of less than half an amp, I'm not going to worry about it.

All this fits into a box about the size of a brick (see Photo 3). Most new vehicles come with an alarm system. You can take an alarm output and connect it to trigger the AutoLocator. Or, mount a simple toggle switch somewhere under your dash and power the AutoLocator from this switch through the accessory circuit that's powered when the key is turned on. If anyone tries to steal your car without turning off the secret switch, the AutoLocator goes into action.

You will need to keep your computer in host mode to answer the AutoLocator's calls. Now, I bet you can think of a quite a few other great ways to use this project. There's another feature I want to add, but that's all I'll say about it for now. Care to venture a guess?

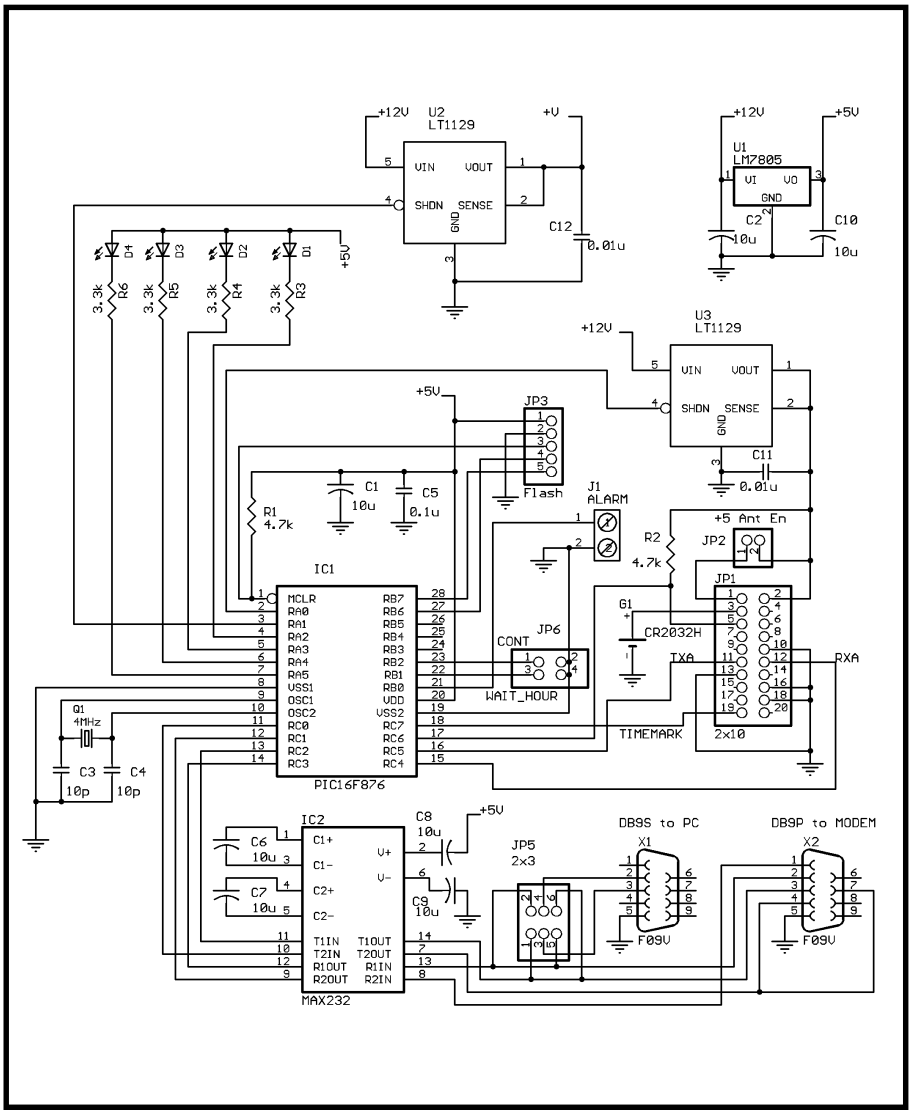


Figure 3—*JP1 is the 50-mm GPS socket header. GPS serial runs at 4800 bps and the receiver can be reset by the PIC's RC6 output bit going low. X2 is the modem connector, and this traffic runs at 9600 bps. Depending on the jumper selections on JP5, X1 can be connected to a PC to monitor the PIC-to-modem or modem-to-PIC traffic.*

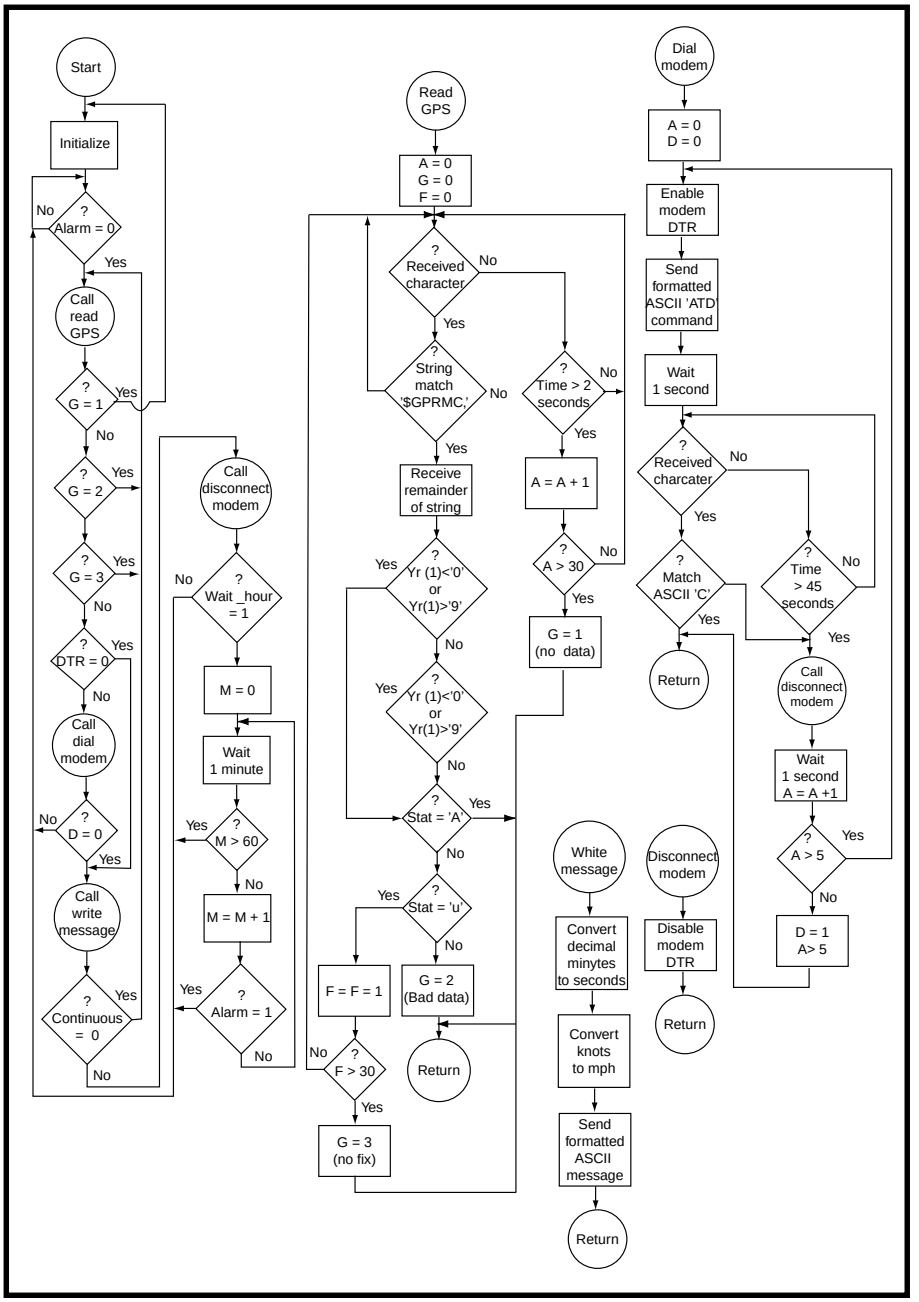


Figure 4—*This flowchart reveals the operations necessary for this project. The use of subroutines simplifies the program. LEDs indicate when the program is in various routines, aiding both visual status and debugging.*

Jeff Bachiochi (pronounced "BAH-key-AH-key") is an electrical engineer on Circuit Cellar*'s engineering staff. His background includes product design and manufacturing. He may be reached at jeff.bachiochi@circuitcellar.com.*

SOURCES

TF10 GPS receiver, GPS-P1MAL antenna
Laipac Technology Inc.
(905) 762-1228
Fax: (905) 770-6143
www.laipac.com

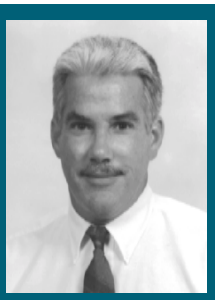
PIC16C87*x*
Microchip Technology Inc.
(480) 786-7200
Fax: (480) 899-9210
www.microchip.com

PicBasic Pro Compiler
microEngineering Labs, Inc.
(719) 520-5323
Fax: (719) 520-1867
www.melabs.com

SILICON UPDATE

Tom Cantrell

Sweet Sixteen



A low-power, flash-memory MCU from Texas Instruments? That was enough to catch Tom's attention. Despite a few skeletons in the closet, TI looks like it may be ready to jump into the MCU field.

There can be little doubt that Texas Instruments is an outfit deserving of a prominent spot in the IC Hall of Fame. In fact, TI's Jack Kilby just received a Nobel prize in Physics (well deserved, I say) for his pioneering work. But, don't forget TI also wrote the bible on TTL, long shouldered the American memory chip flag, and drove the DSP concept beyond its rocket science roots and into the heart of main street products.

Then there's the parade of pioneering consumer products—from calculators, to digital watches, to toys like the good old Speak & Spell.

Of course, every successful company has skeletons in its closet. The question for companies that have never had any problems is, just how gracefully will they handle the inevitable arrival of the first rattling visitor? Successful outfits know when it's time to put a product in the closet, how to keep it there until it's laid to rest for good, or in rare cases, bring it back to life. "Hi, I'm TI-99/4" is the name tag on TI's chief closet dweller.

Old-timers will recall the heyday (before IBM changed the rules) when anyone with a garage, soldering iron, and credit card could enter the PC biz. Quite a few companies gave it a whirl, as TI did with its TI-99/4, which was introduced with fanfare in June 1979 (see Photo 1).

Despite (or perhaps as a result of) TI's heritage with calculators and such, their computer foray was rather rocky. The '99/4 lived fast, died young, and left some 2.5 million good looking corpses. During my research for this story, I was surprised to discover that there's still an active community of '99/4 diehards out there (www.99er.net is a good site to get you headed in the right direction).

'99ers recall the day of reckoning in June 1983 when the '99/4 was axed as Black Friday. Not that TI, Atari, Commodore, Coleco, Exidy, et al had a chance when IBM weighed in, but they were still dark days for the folks in Dallas.

You may ask, "What the heck does some old war story mean for designers today?"

BABY AND BATH

Much like the PC biz, the MPU/MCU market was wide open at the time. Unlike today's domination by a few large players and architectures, there were dozens of microchip wannabes jockeying for position.

Few may recall TI's TMS 9900 chip (here's the connection) that served as the brains for the '99/4. Developed in 1975, it was actually quite advanced, a true 16-bit processor for its time (see Photo 2). Although it had an



Photo 1—*Even with a full quiver of sidecar expansion options, the TI 99/4 became one of the here-today-gone-tomorrow casualties of the pre-IBM personal computer free-for-all.*

interesting design, the TMS 9900 pretty much went down with the '99/4 ship.

Even though I'm sure there's more to it, I can't help but connect the dashing of its computer pretensions with the fact that TI never became a major factor in the mainstream MPU/MCU business. TI has sold a fair number of micros (including 4- and 8-bits), but it has yet to rise beyond the "other" column in market stats. TI can rely on a few huge OEM customers to keep the lights on, but hasn't had much in the way of MPUs and MCUs ("Er, could I interest you in a DSP?") to offer the average designer on the street.

All right, enough nostalgia. Fast forward to the present.

Flipping restlessly through the pile of press kits from the recent Embedded Systems Conference, I almost blew right past the TI stuff. I thought it would probably tout yet another DSP, or perhaps a new and improved op-amp (recalling that TI recently acquired Burr-Brown).

I did a double take and paused in mid-toss to take a closer look….

"Low-Power Flash MCUs from Texas Instruments Available for Only $0.99."

My, my, what have we here?

THEY'RE BACK

My PR radar missed it, but apparently TI has been mobilizing for a major MCU push with the MSP430 family, most recently with the low-cost, flash memory-based MSP430F11*x* variants described in the press release. There are higher-end parts with lots of memory and peripherals, but it's the small (20-pin), inexpensive flash memory 'F11*x* I find most intriguing (see Figure 1).

Now the 'F11*x* lineup consists of four chips—'F110, 'F112, 'F1101, and 'F1121. The amount of onboard memory is the difference between the '110 and '112; the former having 1-KB and 128-byte flash memory and 128-byte RAM and the latter sporting 4-KB and 256-byte flash memory and 256-byte RAM. The 'F11*x*1 versions add an analog comparator.

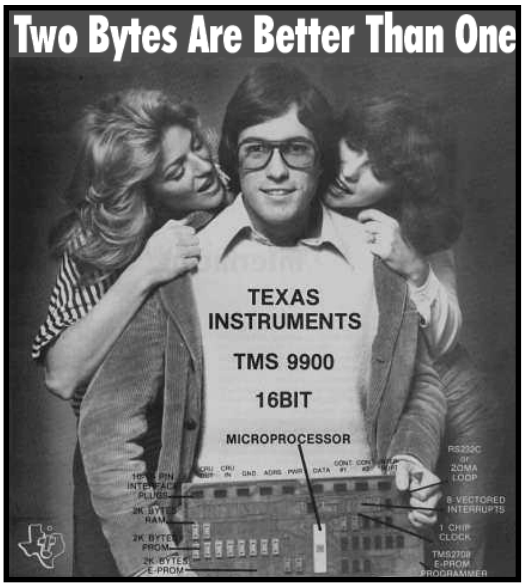


Photo 2—*In the early '80s, two bytes may have been better than one, but still not enough for TMS 9900 to make a dent in the MCU market. Nice collar, dude!*

Figure 1 looks rather typical, yet understates the capabilities and unique features of the part.

Consider the flash memory. Rather than having a single monolithic array, it's comprised of 512-byte main memory segments and 128-byte information memory segments. A segment is the minimum unit of erase granularity, but bytes within an erased segment can be written individually. The main and information memories would typically correspond with an application program and nonvolatile data, but there's nothing in the architecture that demands such a split (i.e., program and/or data can be stored in either memory).

The flash memory offers guaranteed 10k (minimum) cycle write endurance, which is the hardest to meet at the cold extreme of the standard –40° to 85°C temperature range. However, in more temperate environments, the datasheet indicates that 100k cycles is typical. Worst-case data retention is 100 years. That's a lot better than many chips' 10-year specification, which is questionably short for long-life embedded apps.

Rather luxurious for an entry-level chip, there's also on-chip flash memory programming voltage (i.e., V_{PP}) generation, with the proviso that V_{CC} must be kept between 2.7 to 3.6 V during program and erase, versus 1.8 to 3.6 V ('110*x*) or 2.2 to 3.6 V ('112*x*) for normal operation.

It's impressive that the flash memory can be programmed in three ways. The first is via the JTAG port that is also used for debugging. The second option is in-system programming under control of your application software. Because the entire flash memory becomes inaccessible during program and erase, the programming routine must be loaded into RAM for execution. Fortunately, the needed code is trivial because all the low-level details are handled in hardware. The setup is hardened against self-lobotomy with access keys, protected against improper operation or unexpected hardware events, and even has an emergency exit bit, should things get out of hand.

Best of all, there is a bootstrap ROM monitor built in (see Figure 2). Normally, the TEST pin is used to switch four port pins between JTAG or I/O operation. However, tickling the TEST pin just right (two rising edges) at *RST causes the bootstrap ROM to take control. Communicating with a host via two port pins configured as a 9600-bps UART, the bootstrap ROM includes functions to erase, read, and write the flash memory, as well as accessing on-chip RAM and peripherals.

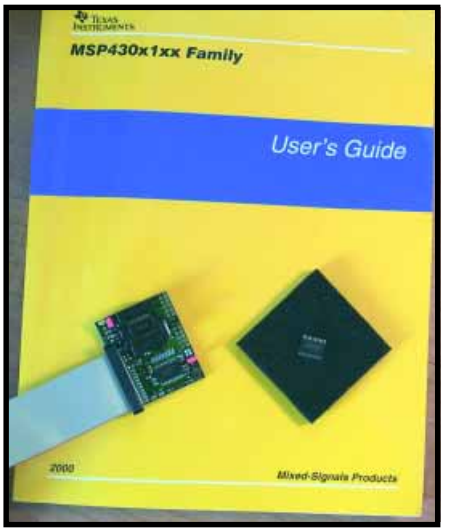


Photo 3—*People complained that you needed a TI minicomputer to develop programs for the '99/4. Not so with the 'F11x controller; just a PC and $49 EV kit will do the trick.*

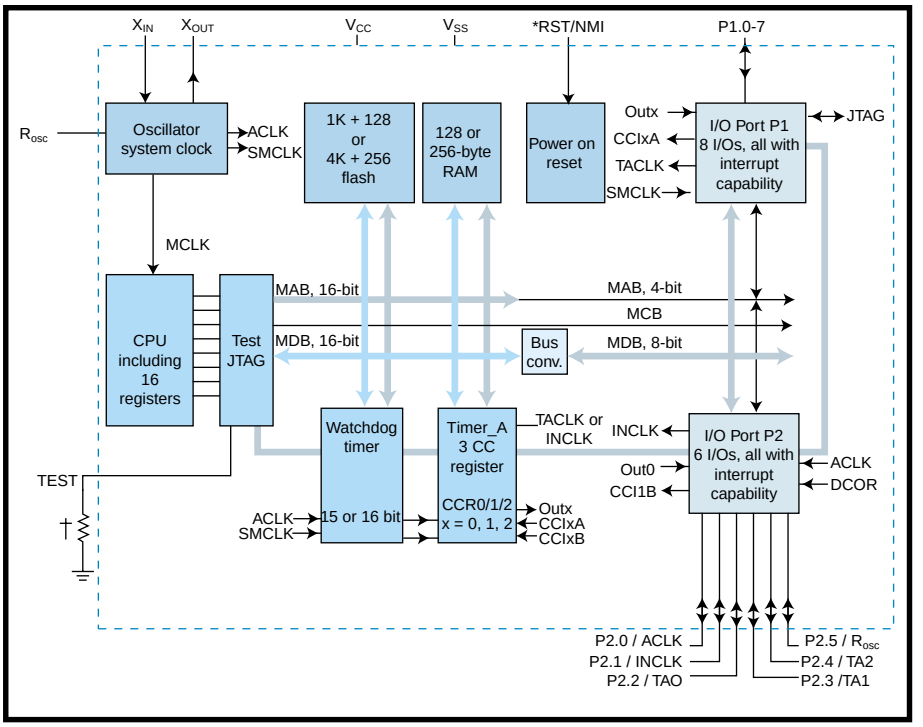


Figure 1—*The 'F11x is real simple and, at under a buck in volume, real sweet.*

ROCK AND CLOCK

Fancy clock generators are the rage, which isn't surprising considering their effect on power consumption. The 'F11*x* diagram just shows an oscillator system clock, but under the hood it's as fancy as anything out there (see Figure 3).

The chip takes advantage of selectable internal or external clock sources. The X_{IN} and X_{OUT} pins provide connections for a 32.768-kHz watch crystal or a high-speed (up to 8-MHz) crystal, resonator, or external clock. Other higher pin count members of the '430 family (e.g., the 'F13*x* and 'F14*x*) even include a connection for an optional second external crystal.

On-chip there's a digitally controlled RC oscillator (DCO) with eight software selectable rates between 100 kHz and 5 MHz, and also the option to set timing with an external resistor. As with all RC oscillators, there's a fair amount of temperature and voltage drift, on the order of ±10% of the nominal setting.

To improve, TI added a modulation feature. In addition to the basic one-of-eight setting, a 5-bit MOD field sets a duty cycle at which the DCO switches between the desired and next higher DCO setting. Assuming an accurate timebase is available (e.g., watch crystal), software can periodically recalibrate by fine-tuning the DCO. MOD is a 5-bit field, yielding about 3% control (32 selections). Because the basic eight steps for the DCO represent about 10% each, that means overall tuning is only a fraction of 1% over the long term (32 cycles), with the understanding that there will be clock-to-clock jitter (DCO step size).

Either the crystal or DCO oscillator (or both) can be individually enabled and disabled. After powerup, during which the DCO is the default clock source, software can switch to the external clock. In this instance, the DCO acts as a fail-safe clock source in case the external clock disappears.

In turn, the two source clocks, crystal and DCO, selectively drive three on-chip clocks, each via its own independent 1/2/4/8 programmable divider. MCLK is the master CPU clock, and peripheral modules can be independently programmed to use either SMCLK or ACLK. ACLK is sourced from the external crystal, and SMCLK can be sourced from either the crystal or the DCO.

A full quiver of five power modes derives from the various clock sourcing and routing options, ranging from Active, with all clocks operating, to Low-Power mode 4, in which everything stops and only the RAM, registers, and I/O port contents are retained.

It's no surprise that low voltage and a leisurely clock rate make for miserly power consumption, but I was surprised by how low these parts go (see Figure 4). Even fully active with both the DCO (1 MHz) and crystal oscillator (32.768 kHz) running, you're talking about 1 µW or so. Stepping through ever-lower power modes, that already miniscule demand is further cut by a factor of 10, and then by 100. Batteries will last a long time, which is nice for customers and the environment.

TIME TRIP

Rounding out the 'F11*x* peripherals are the parallel I/O ports and 16-bit Timer_A.

The general-purpose I/O port setup is clean. Every bit has a place in seven registers—input, output, direction, interrupt, interrupt edge select, interrupt enable, and selection—as either general-purpose or on-chip peripheral I/O. As the list implies, any pin can be used as an interrupt source. I was surprised to see no I/O port characteristics (e.g., totem-pole versus open-drain) or pull-up options (we're getting spoiled, aren't we?).

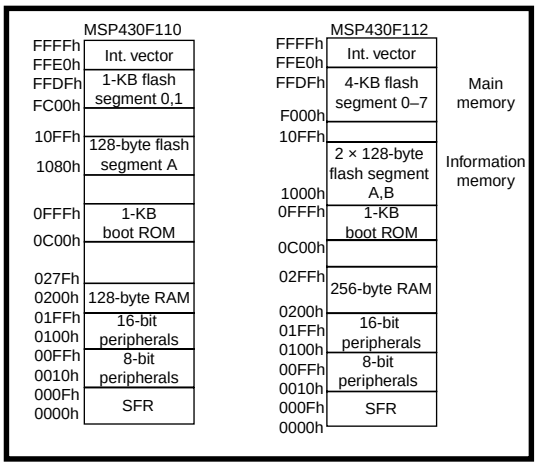


Figure 2—*The top 60 KB is set aside for flash memory and the bottom 4 KB accommodate special function registers (SFRs), 8- and 16-bit memory-mapped peripherals, RAM, and 1-KB boot ROM for easy flash memory programming.*

By contrast, no feature is left out of the Timer_A, which offers an array of options starting with four choices for the clock (two external pins and the aforementioned ACLK and SMCLK on-chip clocks). The chosen source then feeds a 1/2/4/8 programmable divider to generate the timebase for three separate capture and compare channels with various pin and trigger options.

There's also a 16-bit watchdog timer with eight selections for the timebase. Half of the selections are based on SMCLK, and half on ACLK. In a typical configuration (e.g., high-speed crystal or DCO on SMCLK or the watch crystal on ACLK), this allows a broad selection of timeouts, from microseconds to seconds. The watchdog comes up enabled at power-up, but can be used as an interval timer instead by setting a control bit that makes it generate a regular interrupt instead of a RESET.

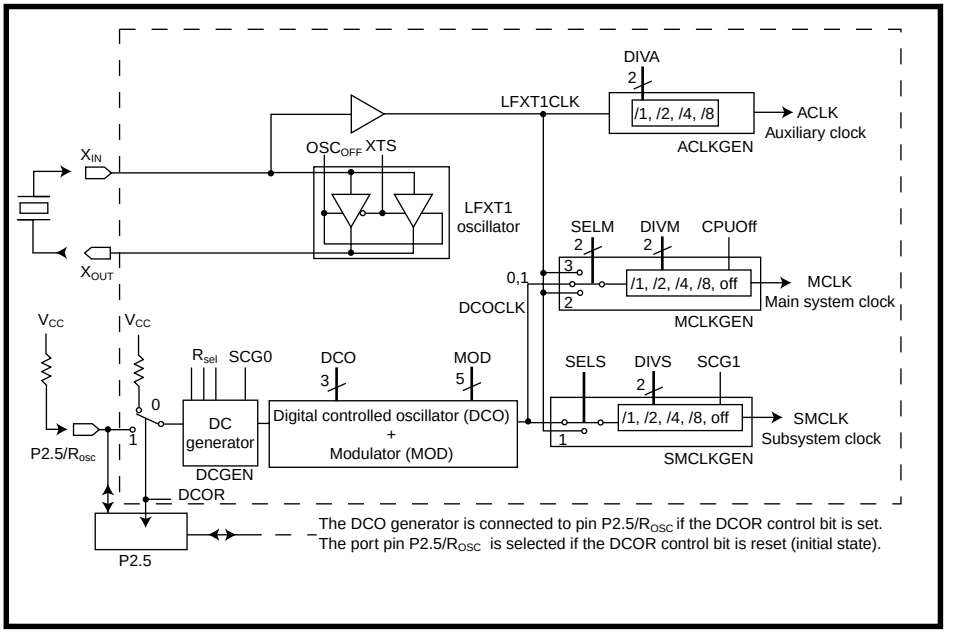


Figure 3—*The clock generator features on- and off-chip sources, extensive configuration options, and complete software control.*

RE-ANIMATOR

TI's "Two Bytes Are Better Than One" advertisement may not have aged well, but maybe the message is more timely than ever. I must say, I felt more than a bit of TMS 9900 déjà vu as I waded through the 'F11*x* docs, so much so that I actually scrounged around for an old datasheet.

Yes, it's all coming back. The '9900 featured a scheme based on only three registers—the PC, a status register, and a work space pointer. The work space pointer pointed to a 16-word window within the memory that was accessible as registers, requiring only short 4-bit fields in the opcode. The only problem was that, because these registers were really off-chip memories, access was slow (extremely slow if you funnel the 16-bit accesses through an 8-bit off-chip bus like the '99/4 designers did).

On the plus side, the architecture is clean. It's a CISC, but it's not plagued with the warts and restrictions like other chips made in that time. Notably, the architecture was pretty regular in terms of allowing an arbitrary mix of instructions, registers, and addressing modes. Arriving on the scene years before the 68K and other chips that touted the same advantages, the '9900 was well ahead of its time.

The MSP430 CPU architecture isn't a TMS 9900 in drag, but it does carry forward many of the same basic concepts. The most obvious change is the addition of a real register file. Otherwise, the philosophy of a simple instruction set with general-purpose registers and plenty of addressing modes remains unchanged.

In fact, with a mere 27 native instructions, the TI part is a CISC that is more reduced than most RISCs. For your programming convenience, a couple of dozen extra instructions are synthesized. But even the entire complement would have to be considered lean and mean by anyone's standards.

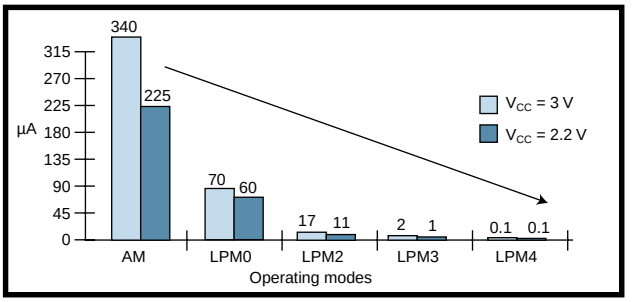


Figure 4—*For price and power, the 'F11x seems more like a 4-bit than a 16-bit chip. Even fully active at 1 MHz (power scales linearly with frequency), power consumption is miserly and drops further in various power-saving modes.*

Thanks to the addition of a real register file, coupled with its CISC roots, the '430 supports your favorite programming style. Those of you who prefer a RISC approach can utilize the register file in a load-and-store fashion and other people can take advantage of the easy addressing to do everything in memory.

GET ON BOARD

With their flash memory, low power, and competitive price, the TI parts are great. But, it's going to take more to make a dent in the contested MCU market. In particular is the need to seed and support thousands of designers instead of just a few focus accounts, which can be problematic for companies that are used to dealing with only a few big customers.

Whether or not TI has the commitment and staying power remains to be seen, but it seems to be making all the right moves. Case in point: TI recently signed Digi-Key as a distributor, a must for anyone hoping to be a contender against major competitors like Motorola and Microchip. TI also offers low-cost, surprisingly useful EV kits.

I couldn't resist fooling around with the $49 Flash Evaluation Tool (FET, see

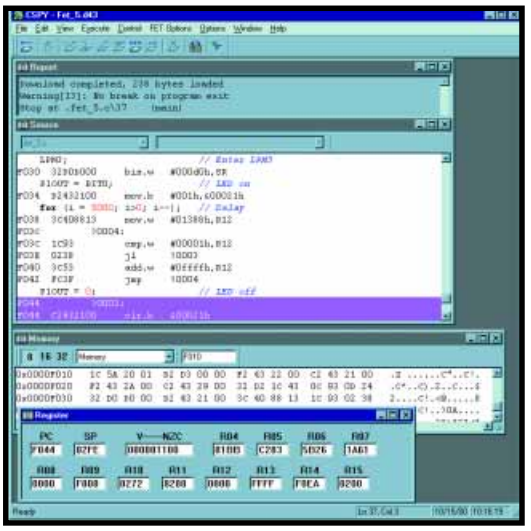


Photo 4—*The EV kit includes a surprisingly functional demo version of IAR's luxury liner Embedded Workbench tool chain, which also can be downloaded for free from TI's web site.*

Photo 3). This little puppy takes full advantage of the 'F11*x* flash memory, low power, and on-chip debug logic.

I admit that I got off to a bit of a rocky start and was reminded why I never became a brain surgeon, as I fumbled with getting the tiny surface-mount chip properly seated in the "I'm-amazed-it-works" spring-loaded ZIF socket.

The good news is, with their low-power advantage, EV power is supplied by the host PC parallel port, so there are no wall warts. The bad news is that some notebook PCs' parallel ports don't cut it. The other good news is that TI states this in its read-me-first docs. More bad news, no one reads that stuff. Switching to a desktop PC got me on the air.

The board comes with a demo version of the IAR Embedded Workbench tool chain that includes the whole shebang—C compiler, ASM, debugger, simulator, IDE, and so forth (see Photo 4). There are demo restrictions such as a 1-KB C code limit, but the useful stuff (e.g., ASM, debug, and simulator) is functional. Support by IAR, one of the leading MCU tools suppliers, lends further credence to the TI campaign.

BACK TO THE FUTURE

To say that a chip that traces back 25 years could be a hit today sounds funny. That's progress? Maybe it is. There are plenty of other old-time MCUs out there to keep TI company.

Sometimes chips that were a good idea yesterday are still a good (maybe even better) idea today. I think the MSP430 family has a lot going for it.

Maybe what the world needs is a new MSP430-based '99/4-*plus*. On second thought, for the chip's sake, better let that sleeping skeleton lie.

Tom Cantrell has been working on chip, board, and systems design and marketing for several years. You may reach him by e-mail at tom.cantrell@circuitcellar.com.

SOURCE

TI-99/4, TMS 9900, MSP430 CPU
Texas Instruments Inc.
(800) 477-8924, ext. 5801
Fax: (972) 995-4360
www.ti.com/sc/docs/products/micro/msp430/msp430.htm

CIRCUIT CELLAR Test Your EQ

Problem 1—The three main operations in a fuzzy interference program are Rule evaluation, Fuzzification, and Defuzzification. Match each term with its definition below and place them in the correct order.

Value based on weighted average
Value based on matrix
Value based on membership function

Problem 3—We know electromagnetic fields are made up of E-fields and H-fields. The electric field (E) is generated between conductors of different potentials. It is proportional to the voltage difference divided by the distance between the conductors and is measured in V/m (volts/meter). The magnetic field (H) is generated around a conductor carrying current. It is proportional to the current divided by the distance to the conductor.

An electromagnetic wave is generated when a varying voltage generates a varying current. This wave travels in free space at the speed of light. Near the source, the strength of the E and H fields will depend on the source make-up. At some distance from the source the make-up of the electromagnetic field changes. Any distance less than this point is called near field. Any distances further than this point are called far field. This distance is based on frequency.

What is Maxwell's formula for this point? What type of field does a dv/dt create? What type of a field does di/dt create?

Problem 2—Serial data transmission speed is limited by cable length. Balanced transmission lines (RS-422) can improve the maximum transmission rates over unbalanced lines (RS-423). For instance, for identical cable (24 AWG twisted pair), an unbalanced interface at a cable length of 100 feet is maxed out at 34 kbps, while the balanced interface could run at 4 Mbps. To extend the cable to 1000 feet, the maximum data rates would need to be reduced to ~1200 bps and 400 kbps, respectively. What would be the maximum data rates be using a balanced interface for 10k feet?

Problem 4—Capacitance (C) varies with the plate area (A), plate separation (D), and material between the plates (K).

$$C = \frac{(8.85 \times 10^{-12}) \times K \times A}{D}$$

The dielectric constant (8.85×10^{-12}) is the permittivity of vacuum. When K = 1, then the dielectric material is air or a vacuum. Other materials (those with different K values) will obviously affect the equation. What must be the area of the plates for a 1-µF capacitor with plate spacing of 1 mm? How is the area affected by changing the dielectric material to Tantalum Oxide?

What's *your* EQ?—The answers and 4 additional questions and answers are posted at www.circuitcellar.com.

You may contact the quizmasters at eq@circuitcellar.com.

8 more EQ questions each month in Circuit Cellar Online see pg. 2

ADVERTISER'S INDEX

The Advertisers' Index with links to their web sites is located at www.circuitcellar.com under the current issue.

Page		Page		Page		Page	
94	Abacom Technologies	87	Designtech Engineering Co.	85	MetaLink Corp.	94	Sigg Industries, Inc.
90	Ability Systems Corp.	89	Diversified Engineering	87	Micro Digital Inc	85	Signum Systems
93	ActiveWire, Inc.	16	DreamTech Computers	92	MicroEngineering Labs, Inc.	94	Simple Network Magic Corp.
8	ADAC	10	Driven to Design Contest	32,82	Micromint, Inc.	91	Sirius microSystems
34,81	Advanced Transdata Corp.	56	ECD (Electronic Controls Design)	77	Midwest Micro-Tek	91	SMTH Circuits
47	All Electronics	61	Earth Computer Technologies	88	MJS Consulting	89	SmartHome.com
68	Amazon Electronics	86	EE Tools (Electronic Engineering Tools)	88	Mosaic Industries	26	Solutions Cubed
33	Amulet Technologies	90	Electronic Systems Consulting	37,57	MVS	85	Square 1 Electronics
86	Andromeda Research	58	EMAC Inc.	15	NetBurner	94	TAL Technologies
94	AP Circuits	56	Engineering Express	94	NetMedia Inc.	58,74	Technologic Systems
72,73	Atmel Corp.	86	FDI (Future Designs, Inc.)	85	Nohau Corp.	93	Technological Arts
76	Avocet Systems	90	General Device Instruments	90	Ontrak Control Systems	C3	TechTools
64	Axiom Manufacturing	23	General Software	C2	Parallax	87	Tern, Inc.
93	Bagotronix	86	Hagstrom Electronics	92	PCB Express, Inc.	74	Trilogy Design
86	Basic Micro	91	HVW Technologies	87	Peter H. Anderson	69	Vetra Systems Corp.
68	Beige Bag Software	88	IMAGEcraft	85	Phytec	77	Virginia Tech Transport Inst.
90	CCS (Custom Computer Services)	87	IndustroLogic, Inc.	88	Phyton	86	Virtual Tools, Inc
92	Cermetek Microelectronics, Inc.	91,92	Intec Automation, Inc.	93	Picofab, Inc.	93	Weeder Technologies
23	ChipCenter (eChips)	41	Interactive Image Technologies, Ltd.	91	Prairie Digital, Inc.	1	Wilke Technology GmbH
88	Circuit Specialists	91	Intronics, Inc.	88	Pulsar, Inc	91	Xilor Inc.
37,89	Connecticut microComputer, Inc.	90	Island Logix	17	R4 Systems	89	Z-World
93	Conitec	65,87	JK microsystems	66	Rabbit Semiconductor	92	Zanthic Technologies Inc.
91	Copeland Electronics, Inc.	75	JR Kerr Automation Engineering	65	Remote Processing	39	ZiLOG
94	CSMicro Systems	89	Laipac Tech	86	R.E.Smith		
93	Cyberpak Co.	7	Lemos International	89	RLC Enterprises, Inc.		
25	Cygnal Intergrated Products	9	Link Instruments	92	RMV Electronics, Inc.		
89	Data Design Corp.	90	Lynxmotion, Inc.	59	Saelig Co.		
C4	Dataman Programmers, Inc.	2	Marlin P. Jones & Assoc. Inc.	5	Scott Edwards Electronics, Inc.		
68	Decade Engineering	92	MCC (Micro Computer Control)	90	Senix Corp.		

Quirks and SHARCs

Where I Stand: A Digital Altimeter

Designing with the AVR: An Overveiw

A Remote-Controlled AV Switch with S-Video

Above Ground Plane: Bring the Noise!

From the Bench: Where (Exactly) Are You?

MicroSeries: Fundamentals of Second-Order Systems: Part 2:

Silicon Update: '51 Soldiers On

EPC **Real-Time PC:** A Cup of Java: Part 3–Hardware Issues

EPC **Applied PCs:** One Thin Dime

PREVIEW

Signal Processing

127

PRIORITY INTERRUPT

Electronic Evolution



I wish that everything could stay the same, but we all know it can't. When I look at my picture on this page I have to admit that my hair is a whole lot grayer now than when that picture was taken seven years ago. The other immutable fact is that a copy of *Circuit Cellar* costs about 2½ times more to produce and ship today than it did back then. The increased costs aren't just paper and postage. We have a larger staff these days, pay a lot more to our authors, and have to deal with a lot more of the competitive realities of being a leader in what we do.

Normally, when a company experiences increased costs the remedy is to either reduce services or raise prices. There was a period of time this fall when we recognized the inevitability of the situation and caved in to business reality. The domestic price of a *Circuit Cellar* subscription, originally set at $21.95 in 1993, was scheduled to go up to $24.95 with this issue. Foreign rates, already a wallet-killing $50 per year, would have gone up as well. Shortly before it was to be cast in print, however, I decided to poll the readers about a change in magazine delivery that might solve everyone's problem.

A couple of months ago we posted a poll on the *Circuit Cellar* web site about offering the magazine in PDF format. Let me clarify that further by saying that this PDF magazine would be 100% equivalent to the print magazine. It would be in color, contain all the advertising, and be printable. Basically, all I was asking was if you would subscribe to a PDF version of *Circuit Cellar*.

About 500 people answered the poll in the short time it was posted. The results demonstrated an intense desire among the web-connected, Internet-savvy *Circuit Cellar* audience. 70% of the respondents answered that they would be interested. 71% of foreign non-subscribers would go for it right away and finally, a phenomenal 90% of our present foreign subscribers were interested, too. About the only concerns expressed were that the print magazine continued to be available and that the PDF subscription should cost less than print.

Well, I wish all my publishing decisions were this easy. Starting this month, *Circuit Cellar* magazine is available in PDF format. Of course, easily said isn't necessarily easily or inexpensively done. For everyone who thinks everything posted on the Internet should be free, forget it. You get what you pay for. We are publishing our new Electronic Edition to reduce the cost burden on the print magazine and maintain both our circulation and quality. A magazine has considerable overhead besides paper and postage. These expenses remain regardless of whatever delivery means we use. To reduce the cost-burden being solely on the print magazine, I've determined to spread these costs across our entire circulation base. It keeps the print subscription at $21.95, offers a reduced price ($15 per year) to Electronic Edition subscribers, and gives foreign readers an option that is less that $50 per year.

The infrastructure for doing an Electronic Edition properly is also costly. If our projections are anywhere near correct, we will be dealing with tens of thousands of downloads very quickly. Systems quickly crash when dozens of readers around the world attempt to download a 6-MB file at the same time. This concern is even more acute if they are all DSL or cable modem users. Rather than limiting ourselves to a modest connection like a T1 line (1.5 Mbps), for the actual downloads *Circuit Cellar* is contracting for a high-bandwidth OC-3 connection through two different Tier-1 backbone service providers. At 155 Mbps, this OC-3 is equivalent to one hundred T1 lines! We trust that this will be enough bandwidth for even peak times.

Finally, I want to assure everyone that the print magazine is not going away and we aren't just moving online. The Electronic Edition is the most effective way for us to regulate costs and expand *Circuit Cellar* into the ever-increasing global marketplace. This is merely part and parcel to the continuing evolution of *Circuit Cellar*. I invite you all to check it out at www.circuitcellar.com/electronic-edition.



steve.ciarcia@circuitcellar.com