EMBEDDED PC MONTHLY SECTION

CIRCUIT CELLAR INK®

THE COMPUTER APPLICATIONS JOURNAL

#95 JUNE 1998

GRAPHICS AND VIDEO

Steve & Jeff Turbocharge a Security System

Wearable Multimedia

Designing Low-Power Systems

HTML and Hand-Held Devices

$3.95 U.S.
$4.95 Canada

TASK MANAGER

A Mind is a Terrible Thing...

How often do you get the opportunity to influence and shape tomorrow's design engineers? If you're involved in education, you're already there. I'm not talking to you. I'm talking to the engineers who sit in a lab or in front of a computer all day and whose community involvement may extend to playing softball with the town's rec program.

One of our local universities (Eastern Connecticut State University) is exploring the possibility of building upon their already-established computer science program by adding one or more new majors and/or minors. One of the steps school officials are taking is to get feedback from the local community about what programs should be offered and how they should be structured. The Computer Science Advisory Board asked Circuit Cellar to select a representative to join the group, and I was elected. Other members include representatives from insurance companies, computer vendors, primary and secondary schools, community colleges, and nearby nationally known universities.

At our first meeting last night, I was amazed at how well such a diverse group could work together to discuss a common goal: what can we offer today's college students to best prepare them for the changing world of MIS and computer engineering. Being able to tap into my own educational background coupled with over a decade of work experience, I hope I'm able to offer some useful feedback and suggestions in the coming months.

As a magazine, I like to think we have some influence over tomorrow's engineers as well. Many dyed-in-the-wool engineers begin experimenting with computers and electronics long before entering college. We have many readers who fit that category and who benefit from articles written by engineers in the field. We also have our college program in which we supply professors with free copies of the magazine for all the students in their engineering classes.

So what's my point? I want your help. We've been doing some work on our Web site, and are going to start offering perks to subscribers. Among those will be short application notes. If you have a favorite tip or technique that you'd like to share, send it along and perhaps we can use it on the Web site. Our published articles are a technical source for these college students. Supplemental application notes and technical tips can only add to their total understanding.

At the same time, I want to encourage you to become involved in your local school system. Students need to be exposed to computers as early as possible if they are going to come out ahead in today's high-tech society. Teachers have enough to do without having to figure out what's wrong with a PC's configuration or why they can't see the server on the network. Our readers possess an incredible wealth of computer knowledge, and just a fraction of that applied to the schools could benefit dozens of young minds.

I look forward to continued work on the Advisory Board. And when my oldest daughter starts kindergarten in a few months, I plan to check with her school to see if I can do anything to help.

ken.davidson@circuitcellar.com

CIRCUIT CELLAR INK®

THE COMPUTER APPLICATIONS JOURNAL

EDITORIAL DIRECTOR/PUBLISHER
Steve Ciarcia

ASSOCIATE PUBLISHER
Sue (Hodge) Skolnick

EDITOR-IN-CHIEF
Ken Davidson

CIRCULATION MANAGER
Rose Mansella

MANAGING EDITOR
Janice Hughes

BUSINESS MANAGER
Jeannette Walters

TECHNICAL EDITOR
Elizabeth Laurençot

ART DIRECTOR
KC Zienka

WEST COAST EDITOR
Tom Cantrell

ENGINEERING STAFF
Jeff Bachiochi

CONTRIBUTING EDITORS
Ingo Cyliax
Fred Eady
Rick Lehrbaum

PRODUCTION STAFF
John Gorsky
James Soussounis

NEW PRODUCTS EDITOR
Harv Weiner

Cover photograph Ron Meadows – Meadows Marketing

PRINTED IN THE UNITED STATES

ADVERTISING

ADVERTISING SALES REPRESENTATIVE
Bobbi Yush
(860) 872-3064
Fax: (860) 871-0411
E-mail: bobbi.yush@circuitcellar.com

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

CONTACTING CIRCUIT CELLAR INK

SUBSCRIPTIONS:
INFORMATION: www.circuitcellar.com or subscribe@circuitcellar.com
TO SUBSCRIBE: (800) 269-6301 or via our editorial offices: (860) 875-2199

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 INK, 4 Park St., Vernon, CT 06066

AUTHOR CONTACT:
E-MAIL: Author addresses (when available) included at the end of each article.
ARTICLE FILES: ftp.circuitcellar.com

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

CIRCUIT CELLAR INK®, THE COMPUTER APPLICATIONS JOURNAL (ISSN 0896-8985) is published monthly by Circuit Cellar Incorporated, 4 Park Street, Suite 20, Vernon, CT 06066 (860) 875-2751. Periodical rates paid at Vernon, CT and additional offices. **One-year (12 issues) subscription rate USA and possessions $21.95, Canada/Mexico $31.95, all other countries $49.95. Two-year (24 issues) subscription rate USA and possessions $39, 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 INK Subscriptions, P.O. Box 698, Holmes, PA 19043-9613 or call (800) 269-6301.
Postmaster: Send address changes to Circuit Cellar INK, Circulation Dept., P.O. Box 698, Holmes, PA 19043-9613.

Circuit Cellar INK® makes no warranties and assumes no responsibility or liability of any kind for errors in these programs or schematics or for the consequences of any such errors. Furthermore, because of possible variation in the quality and condition of materials and workmanship of reader-assembled projects, *Circuit Cellar INK*® disclaims any responsiblity for the safe and proper function of reader-assembled projects based upon or from plans, descriptions, or information published in *Circuit Cellar INK*®.
Entire contents copyright © 1998 by Circuit Cellar Incorporated. All rights reserved. *Circuit Cellar INK* is a registered trademark of Circuit Cellar Inc. Reproduction of this publication in whole or in part without written consent from Circuit Cellar Inc. is prohibited.

INSIDE ISSUE 95

EMBEDDED PC

www.circuitcellar.com
★June's Password is Gotcha★

READER I/O

MORE THAN JUST THE BOTTOM LINE

I enjoyed Steve's April Priority Interrupt ("What You Get with a Handshake," *INK* 93). Over the years I've been a distributor, I've seen quite a few changes as well.

The most interesting trend is customer reliance on pre-sales engineering design support. The problems arise when the purchasing department is tasked only with component cost and cannot justify higher cost, regardless of the support offered. This is where the fear of catalog sales companies comes into play.

I look forward to several more years of change in our industry.

Chris Kracht
Hamilton Hallmark
ckracht@hh.avnet.com

MICROPROCESSORS IN TOASTERS? YES!

As an engineer in charge of designing electronic controls for appliances, I read Steve's May editorial ("Design98—A Marketer's PICnic," *INK* 94) with glee. Yes, in answer to his question, there are plenty of toasters that use microprocessors. Rival has had one on the market for two years, Proctor-Silex has one, and so do many others. I'm working on a PIC-based toaster-oven controller, and the microprocessor design ends up being cheaper than a thermostat, because we can eliminate some peripheral components.

I put a microprocessor into a steam iron to implement a new automatic shut-off mechanism (patent pending). That's the least complex appliance I know of that uses a micro.

Lawrence Lile
lilel@toastmaster.com

AND HOW MUCH MORE HAS CHANGED?

Steve's editorial about manufactures reps and distributors ("What You Get with a Handshake," *INK* 93) reminded me of two other changes worthy of note.

First is humor—or rather, where has it gone? Commercial publications have "matured" so much, they're basically humorless. Remember what used to be:

- the Signetics WOM
- Steve's April article in *BYTE* one year when he went to "raid" the fridge. The word picture was just great.
- Unix man pages used to have a section on bugs, which was more of a comment on how to do it right the next time. But, no one admits to bugs anymore, do they?

Second, consider year-2000 problems in embedded processors. I can't remember when the common engineer has had to think so much about project liability, when fear-mongers have predicted so much doom and gloom, and when there has been so little risk assessment.

Sure, there will be problems come Jan. 1, 2000 and Feb. 29, 2000. That's a given. What I don't see is a good solid risk assessment in the popular press. Maybe crying wolf sells more paper, or vendors and professional service providers see an opportunity to make some big bucks.

I'm looking forward to the day *INK* runs an April Fools' article. Steve could always write it under a pseudonym.

Larry Pajakowski
Gurnee, IL

STOP THE PRESSES

Thanks for Jeff's interesting article, "Proprietary Serial Protocols" (*INK* 92). I'd like to comment on one small point. He says a stop bit is used to "signify the end of the single character transmission". Not so.

As far as I understand it, a stop bit doesn't signify anything. Its presence (an electrical "high") merely ensures that, whatever the state of the preceding data or parity bit, there is always a high-to-low transition for the start bit of the next data byte. This transition notifies the receiver that a new data byte transmission is about to begin.

Terry Koh
Singapore

What I meant was the end (or last part) of the data words protocol. You are correct. Without the stop, we could only see a start bit about 50% of the time.

Jeff Bachiochi

PICARO CORRECTION AND UPDATE

Figure 1 in "Picaro" (*INK* 93, p. 31) incorrectly shows R9–R16 as 10 kΩ. These resistors should all be 1 kΩ.

Also, an updated version (April 14, 1998) of the Picaro software is available on *INK*'s Web site.

NEW PRODUCT NEWS

Edited by Harv Weiner

MULTIPLE VIDEO DISPLAY UNIT

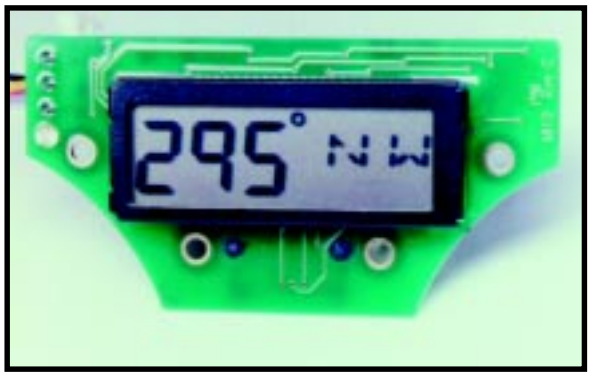
The **SuperView 500** is an advanced display input system that combines up to 10 computer screens and/or video signals on a single monitor or projector. It is compatible with virtually all PCs, workstations, and any monitor or data display projector up to 1600 × 1200 resolution. The SuperView 500 was developed for sophisticated applications where multiple video and computer sources must be displayed simultaneously, such as operations centers, control rooms, and teleconferencing and training facilities.

The SuperView 500 is an external stand-alone peripheral that connects between the host computer and display. It combines the multiple signals downstream of the computer so it doesn't burden the host CPU. Input signals may come from virtually any source like NTSC, PAL, S-Video, FLIR, or any computer signal up to 1280 × 1024 pixels. Video and computer inputs are shown as windows on the main screen. All windows can be positioned, scaled from icon size to full screen, overlaid with computer graphics, or overlapped with other windows. Also, each input can be panned and zoomed to emphasize areas of interest.

SuperView 500 supports software control to manipulate the video windows. X.TV software provides full integration under X Windows, and W.TV software for Windows 95/NT.

Pricing starts at **$10,990** for a camera with two NTSC/PAL inputs. Additional NTSC/PAL inputs and RGB/FLIR inputs are available.

RGB Spectrum
950 Marina Village Pkwy.
Alameda, CA 94501
(510) 814-7000
Fax: (510) 814-7026
www.rgb.com **#501**



ELECTRONIC COMPASS MODULE

The **Navifinder-200** electronic compass module is ideal for use in RVs, snowmobiles, motorcycles, boats, and van/truck conversions. It can easily be mounted in a dash or enclosed in a housing. This module uses advanced calibration algorithms that discriminate between the earth's magnetic field and those generated externally, such as from the metal and electronics in a vehicle. By electronically compensating for these external factors, the Navifinder-200 can provide highly accurate compass readings in all vehicle environments.

The Navifinder-200 is easily wired to any 12-V battery supply and ignition system or externally mounted on/off switch and battery. It outputs the compass heading on a LCD in 5° numeric digits and eight cardinal points (N, NE, E, SE, etc.), with an accuracy of 2°. To provide different viewing angles, the Navifinder-200 can be mounted with up to ±30° of tilt without losing accuracy.

The Navifinder-200 uses Precision Navigation's patented Magneto-Inductive magnetic sensor technology to provide high sensitivity, a large dynamic range, low power consumption, and a low price.

This compass module sells for **$75** in single units and **$32** in quantities of 1000.

Precision Navigation, Inc.
1235 Pear Ave., Ste. 111
Mountain View, CA 94043
(650) 962-8777
Fax: (650) 962-8776
www.precisionnav.com

#502

NEW PRODUCT NEWS

PCI CONTROLLER DEVELOPMENT KIT

A new **Motorola MPC860 development kit** simplifies design efforts to use the Anchor Chips AN3041Q CO-MEM PCI controller to interface between the PCI bus and the Motorola MPC860 family of processors. The kit features a development board with an AN3041Q device interfaced without glue logic to the MPC860. The add-in card contains no onboard memory and plugs directly into the PCI bus in a desktop PC. The cache memory in the AN3041Q caches the local processor's memory needs across the PCI bus to host memory. The designer defines how much host memory (up to 16 MB) is allocated to the MPC860 board through software.

The CO-MEM Mapper software, which runs on any Windows-based PC, defines the MPC860's memory map and automatically configures and displays the AN3041Q's internal registers. Once the AN3041Q is configured, the MPC860 processor can be released from reset to run the application. Mapper software contains debug tools to aid in the integration of the MPC860 processor and its application with the host system.

The AN3041Q uses a highly intelligent cache-memory architecture instead of the FIFO/DMA architecture to provide a high-speed communication interface from a local microprocessor or DSP to the PCI bus. It interfaces directly to most 8-, 16-, and 32-bit processors. The bus interface engine in the AN3041Q enables the local processor to have direct access to the PCI bus. The dual-mode caching controller enables the controller's internal memory to easily interface between the local processor, PCI bus, and host memory.

A design guide shows a step-by-step procedure for developing a system with the AN3041Q. Other reference materials are provided in the kit to make a complete development package.

The Motorola MPC860 development kit sells for **$495**.

Anchor Chips, Inc.
12396 World Trade Dr.
San Diego, CA 92128
(619) 613-7900
Fax: (619) 676-6896
www.anchorchips.com **#503**

NONCONTACT ULTRASONIC SENSOR

The Senix **ULTRA-30-VA** noncontact ultrasonic sensor provides a 0–10-VDC output proportional to the measured distance to target materials. The analog output spans easily between any two distances in its 3–80″ operating range. Packaged in a 30-mm threaded aluminum housing, the sensor features push-button operation to simplify setup and eliminate potentiometers, target status indication to convey target range and stability status, and stable output under lost target conditions. It offers dual-range operation for optimum performance at different target ranges and a user-selectable output response filter for stability under noisy or poor target conditions.

This sensor has a resolution of 2 mm (0.078″) and accepts 15–30-VDC input. It connects easily with PLCs, motor drives, or other controls.

The ULTRA-30-VA is designed for motion-control applications including the measurement of roll diameter and free loop positions for the purpose of tension, speed, and/or diameter control. It's ideally suited for medium-volume OEM machine controls in the pulp and paper, printing, and converting industries.

Senix Corp.
52 Maple St.
Bristol, VT 05443
(802) 453-5522
Fax: (802) 453-2549
www.senix.com **#504**

NEW PRODUCT NEWS

REMOTE I/O MICRO CONTROLLER MODULE

The **Model HCON-1** remote I/O microcontroller module accepts commands over the RS-485 serial bus to control output drivers and obtain inputs from quadrature encoders, temperature modules, and switch inputs. High-current output drivers directly control up to 20 relays or LEDs.

Applications include remote equipment control, home automation, temperature control, and remote relay control.

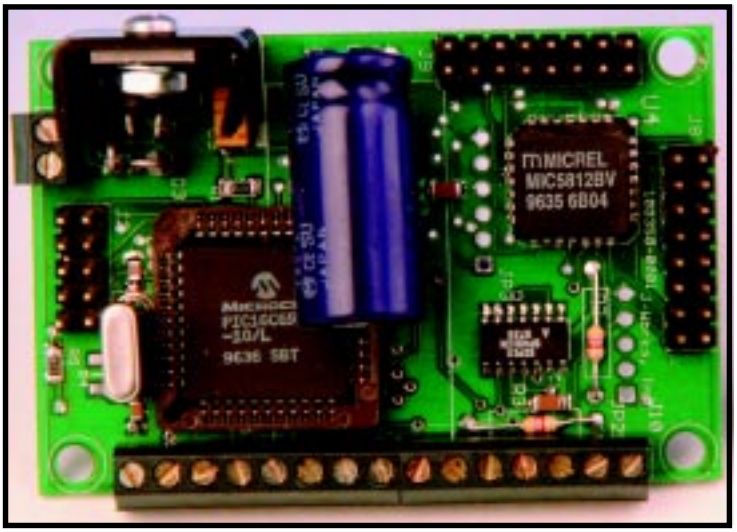
Up to 256 Dallas DS1820 or DS1821 digital temperature inputs can be scanned over the single-wire interface. Key switching can be from a 5 × 5 switch matrix or 10 discrete inputs. ASCII or binary commands can be sent over the RS-485 interface at a data rate up to 19.2 kbps. The compact board measures 1.8″ × 2.7″.

The supplied software includes sample interface code under Visual Basic, Visual C++, and ActiveX control.

The Model HCON-1 sells for **$119**.

J-Works, Inc.
12328 Gladstone St., Ste. 4
Sylmar, CA 91342
(818) 361-0787 • Fax: (818) 270-2413
www.j-works.com

#505



Steve Ciarcia **Jeff Bachiochi**

Gotcha!

Alarming the Alarm System

Alarm companies fall a little short if what you want is entry and exit printouts. Sure, they'll do them. But only at $20 a pop—or the cost of a new system. So, what do you do? If you're Steve and Jeff, you add a little electronic sleuthing to the system.

I was just about ready to pack it in for the night when Jeannette called down the cellar stairs.

"Steve, the alarm service is on the phone. The alarm at the office just went off! They called the police. Shall I say you're on the way to meet them?"

As I grabbed my coat and keys, I cast a quick glance back at Jeannette. Her expression said far more than any verbal exchange.

The gist of it was that if anyone was going to play hero tonight, it wasn't going to be her. She's happy to run a business with me, but gunslinger is definitely not in her job description.

There was a time when sharing the "business experience" might have prevailed, but after a real break-in at our office when the police actually dragged a burglar out in handcuffs, she decided this was one event she'd rather stay away from. Besides, anyone stupid enough to break into a building attached to a courthouse and surrounded by a half-dozen TV cameras probably isn't bright enough to listen to reason anyway.

As I ran for the car, I heard her yell, "Be careful...! Call me...!"

Like most businesses, we have a commercial alarm system. The reason isn't as much to deter crime as it is to qualify for discount on insurance rates. That's the good news.

The bad news is that, because I live closest to the office, I'm first on the call list when the alarm goes off. I get to greet the police and walk around a building with a lot of dark hiding places.

There really aren't a lot of options. If you want the police to treat the call seriously, you better meet them there. And, you also have to watch the false alarms.

Everyone had a sense of humor when someone set the alarm while Ken was still working on the third floor. Since then, the last one out is supposed to page the building or check the parking lot. There has to be a bit of seriousness. After all, the police did nab somebody that one time.

As I pulled into the parking lot, the lights from state and local police cruisers greeted me. All I could think was, please, let them find Jeffery Dahmer or someone, not another false alarm.

After the appropriate introductions, we trooped into the building—they with their guns drawn and flashlights blazing. I don't know why they didn't just turn on the lights, but they preferred to search each room in the dark.

I still don't understand the tactic. Maybe they presumed someone this dumb would invariably use tracer ammo or something else that's easy to see in the dark. Needless to say, I waited until the lights were on before roaming anywhere unescorted.

The results of the search were less than spectacular and somewhat embarrassing. Apparently, workmen had left an outside door to the furnace room unlocked and nobody checked it.

Besides the lock, the door needs a 3½″ thick wooden bar across the inside to keep it from opening. Of course, when someone leaves the door unlocked and only puts in a standard 2 × 4 (1¾″ thick) instead of the usual 4 × 4, the door can open almost 2″. Guess what happens when someone pulls on the door from outside? An unlocked door is an embarrassing predicament.

Needless to say, I apologized profusely. It was a false alarm. If we'd

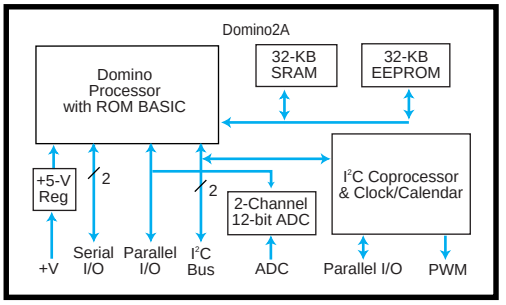


Figure 1—*Domino2 is a small encapsulated controller with a built-in floating-point BASIC interpreter and 32 KB each of EEPROM and SRAM.*

nabbed Charles Manson, wasting their time wouldn't be an issue. I probably could have even gotten away making police and donut-shop jokes. Under the circumstances, however, my only recourse was to thank them, tuck my tail, and return to the car.

I dialed the cell phone. When Jeannette answered, I said in a disgusted tone, "Someone left the damn door open! The only thing worse would be if they didn't set the alarm at all!"

"Well, Steve, I wasn't going to tell you, but I've had reports from the people who come in early that occasionally the alarm hasn't been on."

Next morning, I called everyone with alarm codes together and asked the pertinent who and when questions. I got back mostly blank stares, to be interpreted as, "Not me, man." Given all the people with independent access to the building, that was hardly reassuring.

The obvious answer was the alarm company. We pay them $30 a month to monitor the system and call the appropriate people if the alarm goes off. When it was first installed, we got a monthly opening/closing report that listed the date, time, and access code (these days I suppose we'd call it a PIN) for every alarm set or reset. This was the obvious answer.

Calling the alarm-monitoring service is an experience. They're contracted by your alarm installer and not typically selected by the alarm owner. And since they answer the phone "Monitoring station" and use the installer's name once you give them your account number, you might think they're just down the block. Only when you interrogate the autodialer or otherwise see where the call goes do you realize that your personal monitoring can be 2000 miles away.

Our monitoring company was at the other end of the state, but most large alarm companies deal with centralized service monitors that cover many states at the same time. Regardless of their location, aside from changes to the call list, they're like talking to a brick! Their pat response is that you should call your installer.

They usually charge the installer a flat rate based on a specific service level for all his customers. If he only contracts for alarm calling and none of the monthly printed reports, it's tantamount to bringing the mountain to Mohammed to get one from the monitoring company. Yes, I could get a report for a specific day—at $20 each!

Calling the installer reminded me again why we designed our own home-control system. These people have no vision at all.

"What would it take to get daily entry/exit reports?" I asked.

"I suggest you install a new alarm system," he answered matter-of-factly.

"Would a new system be better than our present ten-year-old hard-wired system?"

"Well, sir," he continued. (Subconsciously, I noted that people generally called me "sir" when they were trying to sell me something. Sometime I'll have to test the financial limits of this theory. Do they start at $100, $500, $1000…?) "It would have the latest technology and use wireless sensors."

"And after I replace the $2 battery in every sensor each year, would it do more than provide a contact closure to an alarm horn and autodial a digital code to the monitoring station like the present one?" I already knew the answer.

"It would use the latest technology to close the contact and autodial the modem, sir…and we could get you one with a printer output." Finally, he mentioned something we wanted!

I continued, "Do you know if it's a standard serial printer port? Do you have a schematic?"

"Well, I'm not sure what it is, sir, but I can supply you with the printer (extra cost). There are never any sche-

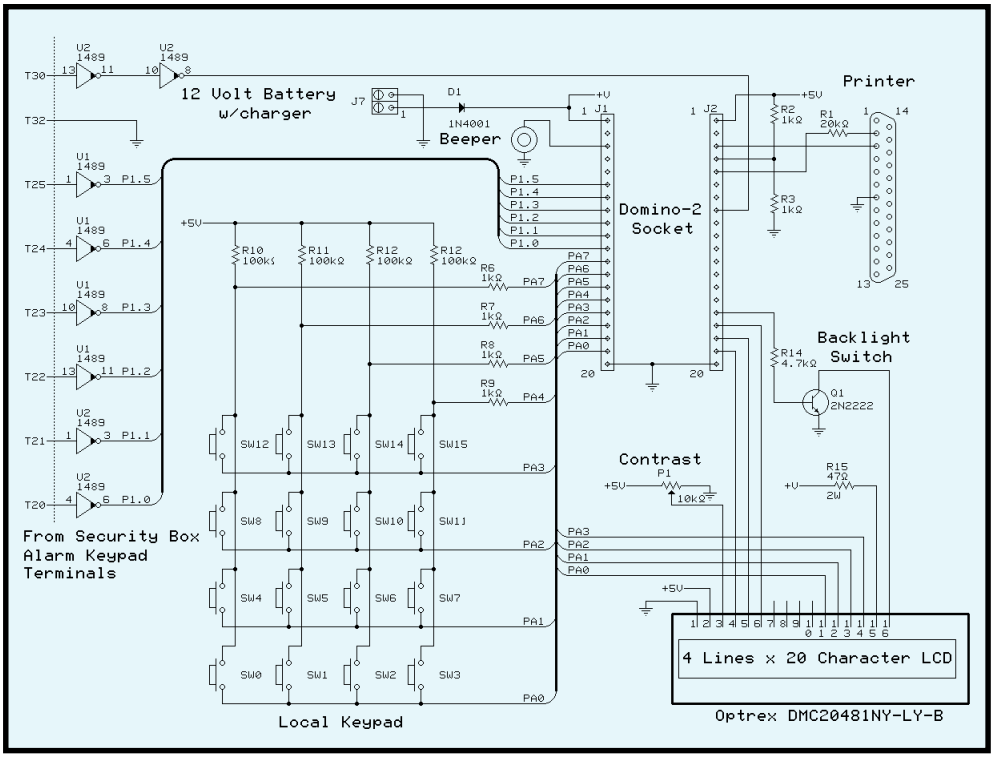


Figure 2—*This schematic shows the ASDL system and how it connects to the alarm signals.*

matics of any equipment, sir. I guess they're concerned about liability."

Liability, schmiliability. The only reason there are no schematics is that the alarm manufacturers don't want competition.

In the end, it's just a cost decision. "How much?"

"Well, sir, if we do just the same as your present system and add the printer, probably about $3500–4000."

I knew this "sir" thing was going to cost me. "Thanks, I'll have to get back to you."

My next stop was Jeff's desk.

"Jeff, this alarm guy is nuts. You can't believe how much it costs for an alarm-code printout. Worse yet, it only records successful entries. It can't tell who punches in a bunch of numbers, never actually sets the alarm, and then just walks out without checking. That's the guy I want."

I knew telling Jeff about such a ridiculous obstruction would be a technical challenge he couldn't refuse. They may not know how to produce a daily event record, but somehow *we* would. We gathered up the 'scope and headed for the furnace room and the alarm-system controller box.

There was nothing surprising about the system. It was your standard ten-year-old Silent Knight alarm system. Documentation was strictly at the installer level. Motion detectors and sensors connect to these terminals, the alarm horn connects here, the phone line goes in here.

The PCB hardware included two microcontrollers that shared the control tasks. Because they had house numbers, Jeff and I concluded they were probably ROM-programmed 8051-family devices. And because it was ROM coded, we didn't have a prayer of changing any internal operation. The best we could hope to do was monitor the external signals.

It would have been great if the alarm designers had taken a traditional engineering approach. Even a *predictable* approach would have been welcomed. In the world of high-volume consumer-quality electronics however, nothing works the way you expect it to, and "cute" is a term frequently used to describe the design technique.

The methodology is quite straightforward. Take a circuit that works the way you'd normally design it, throw away half the parts (or parts cost), and then make it do all the same control tasks. These are the guys who make a fine art of cycle-stealing, multiplexed operation, and hairy-edge qualification. Want them to design your next medical product?

This unit was no exception. The user interface consisted of a 3×4 keypad with a green LED (ready), a red LED (armed), and a single 7-segment LED display (zone). The keypad labels were 0–9, Door, and On/Off. Pressing any button makes an internal beeper sound. Photo 1 shows a close up of the alarm's keypad.

Similar to the operation of most alarm systems, the user looks for a green light indicating the system has no open doors and punches in a four-digit code followed by the on/off button. The alarm goes on, the red LED lights, and you have about 45 s to vacate the building. The process is reversed to shut off the alarm.

Jeff and I concluded the way for us to monitor entry and exit times and codes was to attach a circuit in parallel with this user interface and tap into its communication with the system box. Punch in 6637 and On, and we'd log it to a printer. The only sticky part was guessing where all these signals were so we could tell which key was pressed.

The keypad in the entryway connects to the system via an 11-wire parallel cable. The 3×4 keypad has four rows labeled 1–3, 4–6, 7–9, and Door, 0, and On/Off.

Electrically, we determined that the keys are scanned in two separate groups of six. Two opposite and alternating signals drive the two groups. Pressing none of the keys results in a logic 0 on the three column inputs back at the system board.

Pressing a key diode ORs one of the phases with one or more of the column inputs, like this: 1 = 001, 2 = 010, 3 = 011. A 7 also creates the 001 combination but in step with the opposite phase. The system knows which key is pressed based on the column inputs and the phase of the input signal.

Other signals to the entry panel enable the red or green LEDs and drive a piezo beeper when any key is pressed. All signals are at the 12-V level.

PACKAGED SOLUTION

Jeff and I now knew there were some logical signals we could monitor. The next task was to decide what kind of data-acquisition system we had to configure. But unlike our lightning device ("Ground Zero," *INK* 90), this wasn't just an illustrative magazine project. I wanted to use this thing.

We could have used anything from a PIC to a full-blown PC as the hardware. Our logging system needed a processor board to acquire and analyze the data, a real-time clock/calendar to time stamp the entries, an LCD to view records, a keypad to direct the logger's activity, and a printer interface for making hardcopies on command.

Beyond the strict hardware necessity, system selection is always a tradeoff of competing ideals:

- time (getting this much software done quickly enough to meet a magazine deadline typically rules out assembly language)
- I/O capability (obviously, we needed a serial port and a lot of parallel I/O)
- speed (just how fast does this thing need to be anyway?)
- cost (are we making a few or is it a volume-production device?)
- political bias (some designers will jam in a PC even if it can be done on a PIC)

This analysis pretty much fits half the board ads in *INK*. Fortunately, I get to apply a little political bias of my own.

While there's a little fancy footwork in the interrupt routines, most of the software is a lot of text shuffling among the peripherals (it's easy for the guy who doesn't write the software to say stuff like this). When we looked at the requirements, it seemed like a perfect application for a Domino—or more precisely, a Domino2.

As Figure 1 shows, Domino2 is a small encapsulated controller with a built-in floating-point BASIC interpreter and 32 KB each of EEPROM and SRAM. Best of all, it contains a serial port, lots of parallel I/O, and a real-time clock.

Even if you don't have a ten-year-old alarm, I'm sure you'll find that our method of solving the problem provides some interesting examples of using BASIC-52 in control applications.

ALARM-SYSTEM DATA LOGGER

Figure 2 is the schematic of the alarm-system data logger (ASDL). The circuitry was added to Domino's proto1 board, as shown in Photo 2. The first task was mating the 12-V level alarm signals with Domino's TTL input levels.

There are lots of ways to do this, but one cute way is to use MC1489 RS-232 input level shifters. These inexpensive inverters can withstand ± 30-V inputs while interfacing directly with TTL on the output side. Six keypad lines connect to the processor.

The next detail is determining when a key is pressed. We had two alternatives. We could create a falling-edge key-press interrupt by NORing the three column inputs together. Or, we could take advantage of the fact that pressing any key caused the beeper to sound.

Ultimately, Jeff chose to use the signal applied to the beeper as a key-press interrupt. The three column lines, two phase outputs, and on/off button are read anytime there's an interrupt. Together, they determine the physical key combination.

Normal entry and exit profiles run a total of 10 key presses a day. This plus a 7-byte time and date stamp amounts to fewer than 20 bytes of data-logging space required per day.

Our plan isn't to check this thing daily but to have a record of events available when we need it. If we have a good chunk of data storage space available, how often we dump the records will never really be an issue. Six months of data fits easily in less than 5 KB of memory. We have about 24 KB available.

Program development started with writing the time and date to the LCD. Instead of using a couple I^2C peripheral chips to connect the 4 × 20 LCD and 4 × 6 keypad as typically described for Domino, Jeff chose to minimize external hardware and scan the command keypad in software.

Using this number of I/O bits for the keypad necessitated using the LCD in 4-bit mode rather than the typical 8-bit parallel interface. The 4-bit mode requires two nibble writes to the LCD for each printed character, which entails eight physical operations for each character—set up the control register, raise the strobe line, set up the data register, drop the strobe line. The process repeats for the second data nibble.

The program executes a short initialization routine and then prints day of the week (DOW), month/day/year, and hour/minute/second on the LCD. Using the row and column positioning capability of the LCD (a control register routine), only new data needs to be updated. This makes a great idle screen that also indicates the system is running.

If the displayed time and date is incorrect, as it would be on initial powerup, the user enters the present DOW, month, day, year, hour, minute, and second. Maintaining the time during a power outage is a simple matter of adding a 4.5-V

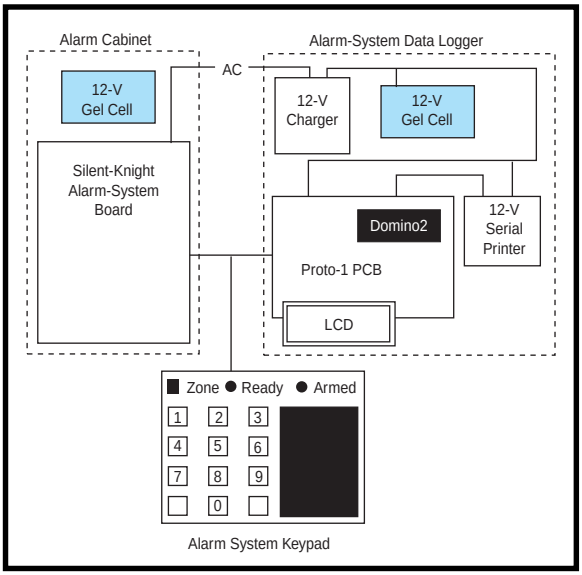


Figure 3—*The ASDL attaches in parallel with the existing alarm keypad. The Domino2-based monitor program registers and records keypad activity and outputs the data to an LCD and printer on command.*

back-up battery to the real-time clock.

Since the alarm system is normally battery-backed and may be needed during a power outage, we decided to battery-back the whole Domino2 circuit. Besides keeping the RTC alive, it maintains any collected data that hasn't been printed yet. Figure 3 gives an overall view of how the complete system was integrated.

Photo 1—*The alarm system PIN is entered via a keypad. The ASDL system monitors and records the time and date of all key presses.*

FANCY FOOTWORK

As I mentioned, it's easy to say it's just a matter of software when you don't have to write it yourself. One of the more challenging aspects of the software was catching the key presses while doing all the other system functions.

This task was accomplished via the BASIC-52 `ONEX1` command. This interrupt command temporarily redirects the normal program flow to a special subroutine.

`ONEX1` is triggered by a high-to-low transition on Domino's INT1 input pin. Obviously, it should be reserved for a high-priority function. Listing 1 illustrates the high points.

The piezo beeper output connects to INT1. When `ONEX1` is triggered, the routine samples the I/O port where all the row and column data is connected to the Domino. This beeper interrupt can occur during either phase of the row drivers.

To ensure the sample is valid, the program checks whether there is a high level on at least one column input line. If none of the first three bits are high, we assume it's the wrong phase and take another sample. Once there is a valid sample, the row driver levels (indicating the two phases) can be compared with the column inputs to determine which of the 12 keys was pressed.

Before leaving the interrupt routine, there are two more tasks. First, save the key-press code into the data-logging buffer and increment the buffer pointer. If the on/off key was pressed, a time/date stamp is added to the log.

Finally, we need to make sure the physical key-press action has ended (to prevent interrupt on the same key press). This is accomplished by repeatedly sampling the port and looking for invalid data (columns all low)

on both phases of the row driver outputs. When RETI is executed, the program resumes where it left off.

GETTING THE GOODS

Displaying and dumping the stored data log is a secondary function chosen from the command menu. The ASDL LCD can show a command list, the time and date, or the data. The data is printed to the LCD in blocks.

When the log is dumped, all the blocks up to and including the present time and date are dumped. We didn't see the necessity for individual date interrogation.

You want the data log? Here's everything, a block at a time.

While the data is being sent to the LCD, it is also sent out the console serial port. Although this could be connected to a PC and logged to a file, Jeff mounted a plain-paper 12-VDC–powered serial printer (measures only 5″ × 5″ × 3″) next to the Domino, so a hardcopy of the data could be printed and retained if necessary.

A short pause between each block lets the printer keep up with the data and provides time to study each block on the LCD before the next display.

Photo 2—*The prototype ASDL sytem is physically assembled using a Domino development board with an LCD and keypad attached.*

The final menu selections enable the user to clear all data from the data buffer and return from the menu screen to the idle screen (displaying time/date).

Since the capturing and logging of alarm codes is a security risk, it would be foolish not to try to prevent unauthorized use. Accessing the menu screen shouldn't be open to just anyone.

Any ASDL entry prompts the user for a PIN. An incorrect entry simply returns to the idle screen.

We also specifically chose to store the data log in SRAM rather than EEPROM. If the Domino is removed from the system, the data log disappears.

GOTCHA

While solving the problem this way gave me the satisfaction of not having to buy more useless equipment from an alarm company, it turns out there was no better alternative.

The monitoring station and a commercial alarm printer could only give me a list of successful alarm operation. But, my major irritation was with people who didn't look for a green LED (indicating that the building was clear) before blindly punching in a code (that the alarm doesn't accept) and blowing out the door.

The ASDL stores every key pressed. It only adds the time/date stamp when the on/off key is pressed. If the alarm isn't set, a quick review of the record should show what code was being entered, even if unsuccessfully.

All of our work may have been in vain. We haven't had any false alarms, and interestingly, there haven't been any mornings with an unset alarm.

Either everyone has become educated by involvement in publishing this article, or seeing Jeff and me constantly playing with the alarm system has made them aware that something is going on.

Regardless, they seem to recognize the seriousness of it all. Jeff and I just have to be careful that we don't let on that we really have fun.

Listing 1—*The highest priority was given to the monitoring of the alarm keypad's key presses. The BASIC* `ONEX1` *command branches to line 20000 to decode a key press and log it.*

```
120    ONEX1 20000 :  REM  CAPTURE ALARM KEY PRESSED
...
20000  REM  INTERRUPT ROUTINE - EXTERNAL BUTTON PUSHED
20010 LED=PORT1.XOR.255 : REM  INVERT DATA
20020  IF (LED.AND.07H)=0 THEN  GOTO 20010 : REM  CHECK FOR DATA
20025 LED=LED.AND.3FH
20026 BUT=LED.AND.37H : REM  MASK OFF ALL BUT BUTTON INFO
20030  IF BUT=15H THEN  GOTO 21000 : REM  BUTTON 0
20035  IF BUT=21H THEN  GOTO 21100 : REM  BUTTON 1
20040  IF BUT=22H THEN  GOTO 21200 : REM  BUTTON 2
20045  IF BUT=23H THEN  GOTO 21300 : REM  BUTTON 3
20050  IF BUT=24H THEN  GOTO 21400 : REM  BUTTON 4
20055  IF BUT=25H THEN  GOTO 21500 : REM  BUTTON 5
20060  IF BUT=26H THEN  GOTO 21600 : REM  BUTTON 6
20065  IF BUT=11H THEN  GOTO 21700 : REM  BUTTON 7
20070  IF BUT=12H THEN  GOTO 21800 : REM  BUTTON 8
20075  IF BUT=13H THEN  GOTO 21900 : REM  BUTTON 9
20080  IF BUT=14H THEN  GOTO 22000 : REM  BUTTON DOOR
20085  IF BUT=16H THEN  GOTO 22100 : REM  BUTTON ARM/DISARM
20090  RETI
20100 Z=Z+1 : XBY(Z)=0FFH : REM  TAG END OF BLOCK
20110 LED=PORT1.XOR.255
20120  IF (LED.AND.17H)<>10H THEN 20110 : REM  WAIT FOR NO DATA
20130 LED=PORT1.XOR.255
20140  IF (LED.AND.27H)<>20H THEN 20130 : REM  WAIT FOR NO DATA
20190  RETI
21000 XBY(Z)=30H
21010  GOTO 20100
...
22000 XBY(Z)=ASC(D)
22010  GOTO 20100
22100  REM  ARMED?
22101  IF (LED.AND.08H)=0 THEN  GOTO 22200
22110 XBY(Z)=ASC(N) : REM  NOT ARMED
22111  GOSUB 23000
22120  GOTO 20100
22200 XBY(Z)=ASC(A) : REM  ARMED
22201  GOSUB 23000
22220  GOTO 20100
23000  REM  STICK IN TIME/DATE STAMP
23005 Z=Z+1
23006 XBY(Z)=0AAH
23010 Z=Z+1
23011 XBY(Z)=MTH
...
23060 Z=Z+1
23061 XBY(Z)=SEC
23080  RETURN
```

Steve Ciarcia is an electronics engineer and computer consultant with experience in process control, digital design, and product development. You may reach him at steve.ciarcia@circuitcellar.com.

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

SOURCE

Domino2
Micromint, Inc.
4 Park St.
Vernon, CT 06066
(860) 871-6170
Fax: (860) 872-2204
www.micromint.com

I R S

401 Very Useful
402 Moderately Useful
403 Not Useful

FEATURE ARTICLE

Steve Mann

WearCam: Personal Imaging System

WearCam is a personal imaging system that lets you capture events as if the camera sits behind your eyes. It can even manipulate its shots—zooming in and out. The real world gets equal footing with the virtual.

Machine vision and artificial intelligence (AI) are problems that have challenged researchers for decades. Many easy tasks for humans can only be done by computers in carefully controlled settings, if at all.

Rather than emulating human capabilities in a computer, an alternative framework, humanistic intelligence (HI), creates a synergy between human and machine in which the human is part of the feedback loop of a computational process, as illustrated in Figure 1.

Instead of implementing an image-processing neural network inside a computer, we need to realize we have a good neural network sitting on our shoulders—and it is already outfitted with two very good imaging devices.

EMBODYING HI

WearComp, which originated in the 1970s as a tool for exploring vision, offers a new form of human-computer interaction in which the computer exists within the user's personal space, is controlled by the wearer, and has both operational and interactional constancy (i.e., is always on, ready, and accessible).

WearComp typically consists of a body-worn computer system, a visual display over one or both eyes with text and graphics display capability, and an input device with five or more push-button switches that may be operated by one hand. Other input devices may include microphones and one or more video cameras positioned to view the same subject matter the wearer sees.

Because the apparatus is worn on the body, input devices often include contacts that touch the body to determine skin conductivity or heart rate, as well as various transducers such as a respiration monitor.

To get a better feel for HI, consider this scenario: You're walking home wearing your WearCam. It's measuring your heart rate, respiration, footstep rate, and so on. (Dividing heart rate by footstep rate serves as a visual saliency index for controlling frame rate of video capture and transmission.)

Suddenly, an assailant wielding a sawed-off shotgun demands cash. You see him appear on the viewfinder concealed inside your glasses, and the computer, through intelligent signal processing, recognizes the increase in

Photo 1—*Multiple push-button switches (e.g., microswitches in this data-entry device from the 1970s and early 1980s) are pressed in different combinations to issue different commands.*

heart rate and decrease in footsteps and sends high-resolution image data to remote locations at maximal frame rate. Friends and relatives are automatically summoned to see the situation.

Essentially, the computer presents the image on its viewfinder screen and measures the human's response to that image. No high-level machine vision or AI is used. Instead, the human brain and human visual system are inserted into the feedback loop of a process that determines whether or not the situation is dangerous.

BUILDING WEARCOMP

Most modern cameras contain some computational capability. The computational capability in WearCam—WearComp—is more than just a micro for calculating exposure. WearComp is a complete Internet-connected multimedia computer system running a Unix-like operating system.

WearCam is for a technical breed that wants the challenge of becoming a camera and experiencing life through the screen of an image-processing computer. WearCam6/WearComp6 can be easily built from off-the-shelf components.

Obviously, since this system is in contact with your body, you need to take the utmost care to protect yourself from physical risk or danger by following careful engineering practices. As well, extensive wearing of the device could cause eye damage as well as damage from RF.

BRIEF HISTORY

Table 1 gives you a look at WearComp's evolution, including the microprocessor used and its expanding text and graphics capabilities.

WearComp0 and WearComp1 were specifically designed for controlling experimental body-worn photographic lighting equipment. They certainly weren't general-purpose computers.

But WearComp2 was. It could execute a general instruction set, and it even had a BASIC interpreter, making it easy to write programs to edit ASCII text files, exchange messages (e.g., an E-mail of sorts), do floating-point calculations, and other things that go beyond the kind of functionality you'd expect in a camera system.

WearComp3 was much less capable than WearComp2. However, it was smaller and could be better integrated into clothing (rather than being worn like a backpack).

It also marked the beginning of using the chest area as a display space others could see. This design choice arose out of the fact that WearComp3 put more emphasis on computer-supported collaborative photography than on the individual spirit WearComp2 had been based on.

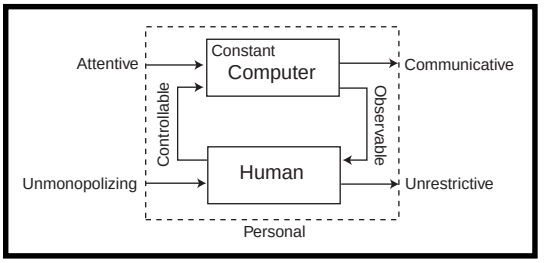


Figure 1—*A camera that embodies humanistic intelligence (HI) uses the human operator for its intelligence rather than trying to emulate human intelligence directly. Within the HI framework, camera intelligence arises as a result of the existence of the human in the feedback loop of some computational process. In the case of WearCam, the camera ceases to be intelligent as soon as it is no longer being worn.*

WearComp4 returned to a fully functional general-purpose computer, as did WearComp5, which was a much more powerful image-processing workstation running the Linux OS with X-windows (XFree86) and using a PCMCIA video-capture device.

In this article, I describe WearComp6 because it doesn't require any special nonstandard devices or custom ASICs. But, there's also a WearComp7, a covert version of wearable wireless webcam that was first completed in 1995, that looks like ordinary glasses and clothing.

And, a recently completed WearComp8, a covert version of wearable wireless webcam, embodies a new kind of mediated-reality experience with enough visual acuity so you can engage in fast-action activities (e.g., playing sports) while looking through the viewfinder and recording the experience.

BUILDING WEARCOMP6

Most of the research issues including the mathematical framework for personal imaging and mediated reality (MR) are described at <wearcam.org/research.html>. Here I simply describe how to build WearComp6. Go to the Web site for details on programming and integrating it into a WearCam6 system.

WearComp6 is built from standard PC/104 modules. The modularity of WearComp6 makes it easy to change the system's functionality quickly.

In a sense, you end up with a wearable image-processing lab, and you might want to add other boards such as an ADC board for an oscilloscope. I implemented an oscilloscope in WearComp2 and found it useful for diagnostic work. I could keep my eyes on the work while making measurements.

Isolation is important. You don't want high voltage going through your body when you display the signal on a high-voltage wire and the ground clip pops off.

BATTERIES

Early versions of WearComp used lead-acid batteries, whereas later (mid 1980s) versions used NiCd batteries.

For constant operation, I generally use at least two 12-V batteries. These batteries typically have lugs that connect to crimp-on connectors.

However, in wearable applications, the lugs are easily broken off or shorted by materials you might carry in a pocket with the batteries. To avoid fire and explosion hazards, I solder wires right to the lugs and insulate them well.

Be careful to place a fuse right next to one of the lugs of the battery, not in the cord going to the battery. Otherwise, if something wears through the insulation on the cord upstream of the fuse, you've got a problem.

The best fuses are the automotive type with solder lugs. I typically place a fuse right near the positive lug, as close as possible.

One lug of the fuse can be soldered to the positive lug of the battery, and a red wire soldered to the other end of the fuse. A black wire can go directly to the negative side of the battery.

I wrap both lugs in several layers of fiberglass tape and epoxy. You need to wrap all the way around both the positive lug and the fuse near it because general wear and tear on a wearable apparatus is much higher than for other embedded systems.

Name	Completed	Processor	Text and Graphics	Where on Body
WearComp0	1970s	electromechanical	—	back
WearComp1	1970s	SSI, MSI	ATV RS170	back + waist + shoulder
WearComp2	1981	6502	40 × 12, 280×, NTSC	back + waist + shoulder
WearComp3	early 1980s	8085	7-segment displays	waist + chest
WearComp4	late 1980s	80286	80 × 24,640 × 480	ordinary backpack
WearComp5	early 1990s	80486/33	80 × 24,640 × 480	large waistbag
WearComp6	early 1990s	PC/104,80*x*86	80 × 24,640 × 480	medium waistbag
WearComp7	mid/late 1990s	TMS320C3*x*/4*x*	RS170	underwearable
WearComp8	late 1990s	Pentium	RS170	underwearable

Table 1—*As you can see, WearComp has gone through quite an evolution on its way to becoming what it is today.*

Alternatively, you can buy a battery vest for about $600. You then have a ready-to-wear power supply with plenty of pockets for computational apparatus or additional components. These vests are designed for high-current output (e.g., video lights and large cameras), so it's a good idea to include an additional fuse of lower current rating, consistent with the use expected of WearCam.

LI-ION BATTERIES

In the early to mid 1990s, I began using lithium-ion (Li-Ion) batteries. Sony provided me with camcorder batteries before they were commercially available.

Now you can find Li-Ion camcorder batteries almost anywhere. You need a minimum of four batteries (two sets of two in series) for a constant-running 12-V supply.

My version of WearCam, for example, is a complete photographic studio and video editing facility, so it's not surprising that it needs more than just one camcorder battery. These batteries have built in minifemale banana connectors, so they're easy to connect to the system.

BRIDGING THE POWER GAP

Ordinarily, when you remove a battery from WearComp to insert a new one, there is a brief power gap. One or more large capacitors can keep power to the circuit during this time.

However, a better alternative is to put diodes in series with each set of battery terminals, as depicted in Figure 2. This power bridge has the following advantages over using a single battery:

- the new battery can be inserted before the old battery is removed, so the power gap is bridged
- multiple batteries may be bridged together for increased power capacity
- it protects against possible damage if battery polarity is incorrect
- mixed brands and types of batteries can be bridged together without damage resulting from one "charging" the other

In my typical usage, WearComp6 forms the basis of a complete Internet-connected multimedia video production facility, ham-radio television station, and various other things.

Because of the large amount of equipment I have powered up, the diodes dissipate some heat and must also carry the full current of the maximum anticipated load. I found that a bridge rectifier, by virtue of its larger surface area and the ease with which it may be heatsinked, dissipates the heat better and is easily sewn into my clothing.

VOLTAGE REGULATORS

The weight, for a given energy level, is much less for Li-Ion batteries compared to lead-acid and NiCd batteries. But, the output voltage of Li-Ion batteries varies widely and drops significantly with usage from a full charge.

Lead-acid batteries exhibit this inconstancy of output voltage to some degree (compared to NiCds, which are much more self-regulating). But, Li-Ion batteries are far worse, almost certainly needing a voltage regulator.

A single 12-V battery can power most of the apparatus, together with an integrated switching voltage regulator to bring the 12 V down to 5 V for powering the computational portion of the apparatus.

POWER SUPPLY

Isolation from any test probes to the rig is required, but I decided not to isolate the batteries from the rest of the rig. Therefore, I chose to use a nonisolated integrated switching regulator.

I selected the PowerTrends PT6302 (3-A ISR), which is much more efficient than the isolated regulators (e.g. Datel). This extends battery life and produces much less heat.

My WearComp6 is built around the Ampro CoreModule, along with various other modules that I swapped in and out depending on my plans.

I connected the power to the CoreModule rather than using a power header connector. This enabled me to cut off all the pins on the bottom board, saving considerable space.

It's worth the extra money to get the CoreModule development system, especially if this is the first unit you build. This development system includes the power connector (e.g., MX40) with a 10 (or 8) pin female connector—2 rows of 5 (or 4) to mate with the header pins on the CoreModule. I generally cut off the large connector and 12-V wires, leaving just the small MX40 connector.

It's important to use all three pairs of redundant 5-V wires in parallel for system reliability, especially when you use video-capture cards because they often introduce current surges on startup. Otherwise, you may find that, when you issue a frame-capture command, the computer system reboots itself.

Originally, I built enclosures using sheet metal and a metal-bending machine. Here, however, I'm putting together a system using a commercial off-the-shelf enclosure.

A suitable choice is the so-called half cube enclosure. With it, you can easily keep the power cables 2″ or less in length.

All three pairs of 5-V wires may connect to the various parallel pins of the PT6302 ISR. It is a nice coincidence that CoreModule's power connector and the ISR both have three redundant +5-V pins. Connect one of each red wire from the power cable to each of these pins.

The ISR has four redundant ground pins. Connect the three black wires from the CoreModule power cable to three of these, which leaves one ground connection for the 12-V input to the ISR

(higher voltage and correspondingly less current).

Connect a single twisted pair of wires to the input (conductors don't need to be so thick owing to the lesser current, as well as the fact that the ISR makes up for line losses). Make sure the twisted pair has tough insulation because it's outside the enclosure and subject to wear and tear against your clothing.

Here, I used a 100-µF output capacitor and a 47-µF input capacitor with leads soldered to the appropriate pins of the ISR for additional filtering. I selected an input capacitor with a high enough voltage rating to match the range of input voltage that the PT6302 could handle so the rig could run on a wider range of input voltages. I found this useful in the field because I could run the rig from nearly any battery source I might encounter (e.g., a 24-V system).

Next, mount the ISR inside the enclosure. This prevents it from being jostled around where it may touch and short out other components or obstruct air flow. It also helps with heat dissipation (i.e., heatsinking it to the case).

The PT6302 ISR comes in six variants, with and without mounting tabs (select the one with mounting tabs), and each of these comes in three variations—horizontal mount, surface mount, and vertical mount.

The vertical mount is preferable, but it's often out of stock. With surface mount, which is most readily available, the pins touch the case. But, if you use a small aluminum shim, the pins are kept sufficiently far away from the case.

Install the PT6302 ISR near the front of the enclosure, facing inwards. Locate it so that the power cable emanates from directly below where the power connector is located on the PC/104 CoreModule. Note that the ⅛″ aluminum shim keeps the pins from touching the case.

Be careful not to locate objects near pin 12 (the sense pin) of the ISR. For example, if the disk cable comes too close to pin 12, stray emissions will affect the ISR or set up a feedback loop that makes the whole system unstable.

Touching pin 12 when the computer is running generally causes a spike of sufficient strength to reboot the computer. If you don't need it, consider breaking it off or cutting it short so it doesn't act like a receive antenna.

Note the capacitors on the ISR as well as inline right near the MX40 connector. Sometimes I also insert series inductors, especially when using the ANDI-FG board, which seems to need them.

I brought the 12-V power leads out of the enclosure and threaded them through a ferrite bead (healthy paranoia). Then, I soldered on connectors for the battery.

HARD DRIVE

On the Ampro 100-MHz '486 CoreModule, it's best to put the hard drive on the bottom of the case, assuming you have three boards or fewer in the stack.

I normally cover the circuit-board side of the hard drive with cloth tape (thick gaffer's tape works best) as an precautionary measure. If you place the hard drive on the bottom of the case, put it upside down in the case and wire-tie it down.

With the hard drive underneath, you can make a straight run to the header

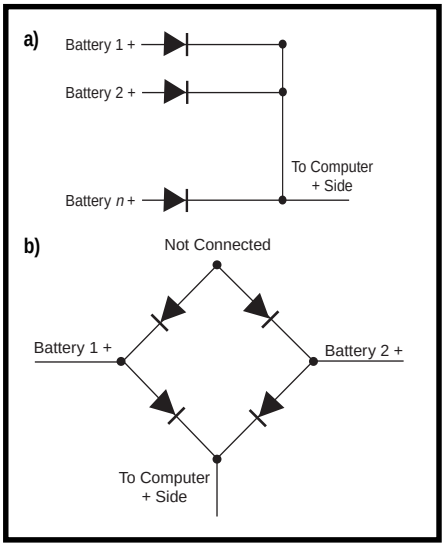


Figure 2a—*A large number of batteries may be combined, and in this way, there is also protection from accidental polarity reversal.* ***b***—*Where only two batteries are needed, a commercial bridge rectifier may be used. In this case, only two of the four internal diodes are used.*

on the CoreModule. Therefore, you can shorten the ribbon cable appreciably, making the insides of the rig neater and resulting in greater reliability and improved air circulation.

ASSEMBLING WEARCOMP6

The Ampro VGA board doesn't properly support 24-bit true color, or at least the color fidelity was not as good as I needed. Having a full 24-bit color display was essential for photographic work because the display ultimately became the camera's viewfinder.

Therefore, I generally use a VGA board from another vendor. The Advantech board uses the Tseng4000 chip, which is fully supported in Linux. I use Linux in WearComp because Microsoft Windows isn't sufficiently reconfigurable to adapt to completely new paradigms in computing.

Traditional computing is a primary task, but WearComp makes it secondary. I frequently respond to an E-mail message, type a manuscript, and shoot a personal documentary video, while walking. Windows cannot be easily used through a teletype interface because it forces its GUI on the user.

The beeping speaker annoys others, so use an earphone jack instead. Alternatively, I use a step-up transformer (e.g., to generate a mild electric shock) or a silent vibrotactile device.

I usually wear the computer in a waist bag or lumbar pack. The Mountainsmith daypack or tourpack is appropriate for the computer itself and leaves room for a good collection of other peripherals.

KEYBOARD

Data entry is typically through the camera (e.g., you can color mark each finger and track them using a PC/104 machine-vision system like Cognachrome ["Robots with a Vision," *INK* 92]). That way, you can use your finger to outline an object and select it by pinching two fingers together.

Alternatively, I use a collection of push-button switches. A few switches, appropriately arranged, may be connected to the parallel port of the computer and serve as a data-entry means (see Photo 1).

If you decide to use the switches as your keyboard, you need to write a small program that takes input this way. But if you write a device driver for both DOS and Linux, don't use LiLo (Linux Loader).

Use loadlin instead because there is no control of the computer at the LiLo prompt since your program or device driver which needs to accept input from the switches hasn't run yet. So, you can't use the keyboard to select operating systems at the LiLo prompt.

Photo 2 shows the completed WearComp6 on my workbench, next to a VGA-to-NTSC scan converter.

TELEVEYES

By connecting WearComp to a head-mounted display and plugging in some sort of keyboard, you have a complete computer system you can use while walking around doing other things.

The limited availability of reasonably priced, small, VGA head-mounted displays as well as their poor tonal range suggest NTSC as an alternative. Indeed, early versions of WearComp used NTSC, and there is a long history of availability of NTSC displays.

In particular, you can often salvage camcorder viewfinders and build them into eyeglasses. I usually find these units for under $20, and it's clearly the lowest-cost solution. Larger tubes (like the ones I have from 15 or 20 years ago) last for years and provide good resolution and excellent tonal fidelity.

Photo 2—*The completed WearComp6 is sitting on workbench next to the VGA-to-NTSC scan converter.*

There is a common misconception (due to cheap game displays and consumer television) that NTSC resolution is significantly less than VGA.

However, good NTSC camera viewfinders often use CRTs with 1000 vertical lines of resolution that adequately display VGA-resolution images or text. Some experimentation is needed since text modes in VGA are often not 60 Hz, but many camcorder viewfinders will sync at 60 or 72 Hz.

If you choose to use a low-cost wearable television set (e.g., VirtualVision), you may want to run XFree86 with increased font size (e.g., 30 × 12).

NTSC OUTPUT

If you know Linux, you'll probably be able to rewrite the `XF86Config` file to output a signal that can drive an NTSC display directly. Unfortunately, such a display only works inside XFree86 and not in DOS or while booting to change the BIOS settings.

A simpler approach, which gives you a display that works outside XFree86, is to use a VGA-to-NTSC converter.

Begin by purchasing a Pocket Scan Converter from AITech. This unit, which costs about $129, consumes a lot of power because of its inefficient regulator. However, you can roughly double its efficiency by replacing its regulator with a PowerTrends ST105VC integrated switching regulator.

Connectors on most devices, including the AITech scan converter, tend to be unreliable. So, I connected the red and black wires for 12-V input and a coaxial cable for video output directly to the circuit board. Routing these through the case, I installed locking inline connectors, which are much more reliable.

SET TO GO

Describing the full details of WearCam is beyond the scope of this article. However, if you want more information about the algorithms and theory behind it, check the Internet sites listed in the References section.

The pictures in this article were taken with another WearCam system I wore while building this unit. The visual record of this assembly is another example of the utility of personal imaging. Much of my work in this area is documented from the first-person perspective of the apparatus I wear.

Thanks to Kodak, Xybernaut, ViA, Kopin, DisplayTech, Liquid Image, HP labs, Thought Technologies, VirtualVision, Compaq, Sony, and Antonin Kimla, for making this work possible, and to Robert Kinney and Rich Landry of Natick for updating my ThinkTank/VibraVest version of my personal imaging apparatus.

Steve Mann, is a faculty member at the University of Toronto, Department of Electrical and Computer Engineering. His present research interests include quantographic imaging, lightspace rendering, and wearable, tetherless computer-mediated reality. He is currently setting up a new Humanistic Intelligence lab to invent the camera of the future. Steve may be reached at mann@eecg.toronto.edu.

REFERENCES

genesis.eecg.toronto.edu/research.html
www.hi.eecg.toronto.edu/hi/index.html
www.wearcam.org/historical/index.html
www.wearcomp.org/wearhow/index.html
www.pc104.org
www.tapr.org
www.radio.org

SOURCES

CoreModule
Ampro Computers, Inc.
4757 Hellyer Ave.
San Jose, CA 95138
(408) 360-0200
Fax: (408) 360-0222
www.ampro.com

PT6302, ST105VC
Power Trends, Inc.
27715 Diehl Rd.
Warrenville, IL 60555
(630) 393-6901
Fax: (630) 393-6902
www.powertrends.com

Enclosures
Tri-M Systems
1301 Ketch Ct., Ste. 6
Coquitlam, BC
Canada V3K 6X7
(604) 527-1100
Fax: (604) 527-1110
www.tri-m.com

VGA board
Advantech America
750 E. Arques Ave.
Sunnyvale, CA 94086
(408) 245-6678
Fax: (408) 245-5678
www.advantech-usa.com
www.advantek.com.

Pocket Scan Converter
AITech
47971 Fremont Blvd.
Fremont, CA 94538
(510) 226-8960
Fax: (510) 226-8996
www.aitech.com

Battery vest
NRG Research, Inc.
840 Rogue River Hwy., Ste. 144
Grants Pass, OR 97527
(541) 479-9433
Fax: (541) 471-6251
www.nrgresearch.com

I R S

404 Very Useful
405 Moderately Useful
406 Not Useful

FEATURE ARTICLE

Brian Kurkoski

Design Embedded Systems for Low Power

When it comes to battery management, conventional techniques don't always cut it. Brian shows how to get around traditional restrictions and still reduce power consumption. Find out how to get the most out of battery life.

You're ready to begin work on your latest design: a handheld measurement and data recorder. The unit must run off batteries for a reasonable length of time, so you want the design to use as little power as possible. A quick calculation shows that conventional techniques yield a battery life of only a few hours, but the system must run for several times that on a single charge.

By reducing the power consumption, however, you extend the battery life. In this article, I show you exactly that—how to reduce power consumption in embedded systems.

Advances in IC technology make it possible to create practical low-power designs with relative ease. The power your board consumes is determined mostly by the design's supply voltage and operating frequency. However, there are a number of other considerations for a successful low-power design.

THE POWER EQUATION

A reasonably accurate approximation of power consumption in an IC is:

$$P = V^2 \times f \times C + P_{static}$$

where $V^2 \times f \times C$ is the dynamic power consumption and is largely under the designer's control. P_{static} is associated with the IC's quiescent current and depends on the characteristics of the die, temperature, and supply voltage. V is the supply voltage, f is the operating frequency, and C is the capacitive load.

REDUCING SUPPLY VOLTAGE

That the power consumption follows a square law is good news: great power savings are realized for small reductions in supply voltage.

Compare a system operating at 5-V supply versus 3.3 V. Reducing the voltage supply from 5 to 3.3 V reduces the dynamic power consumption to:

$$\frac{3.3^2}{5^2} = 43\%$$

of the original, which is a power savings of 57%. Figure 1 shows this graphical square relationship.

To reliably run a system at a reduced supply-voltage level, you must have ICs specified for operation at the level in question. A few years ago, your component selection may have been restricted because the part you wanted wasn't offered in a low-voltage version.

Today, many parts are available for 3.3-V ±10% power-supply operation. (3.3 V ±10% is usually taken to be 3.0–3.6 V; in fact, this is a JEDEC standard for 3.3-V supply voltages.)

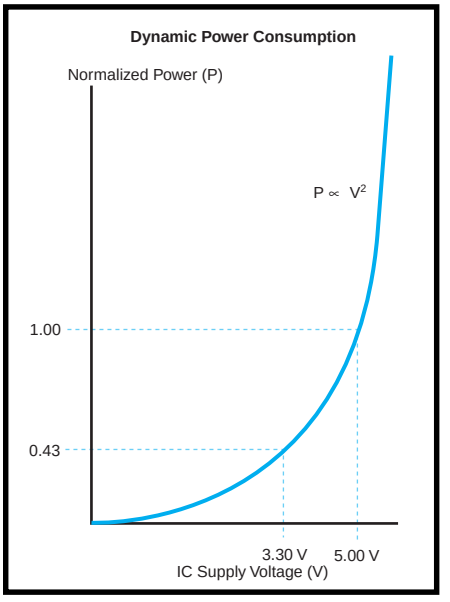


Figure 1—*The dynamic power consumed by an IC is proportional to the square of the supply voltage, so reducing the supply 5.0 V to 3.3 V results in a 57% power savings.*

You can also find 3.0-V ±10% parts (often referred to as 2.7-V parts; 3.0 V – 10%), but they're less common, which makes designing a 2.7-V system difficult.

FAST CLOCKS

Power consumption is proportional to operating frequency. As the operating frequency goes to zero, the dynamic portion of the power consumption also approaches zero. This situation leaves only the static power consumption, which typically is in the microwatt range for CMOS ICs.

Because power consumption depends heavily on clock speed, choose a processor speed as fast as your application needs, and no faster. Running the clock at a frequency higher than necessary wastes valuable battery power.

Also, substantial power savings can be realized from intelligently managing the CPU clock speed. If your CPU clock is fixed at a blazingly high speed to accommodate a compute-intensive task (e.g., data processing), then considerable battery power is lost when the system performs a less CPU-intensive task (e.g., acquiring data).

Enter a solution: the scalable clock. The microprocessor can program the frequency of its own clock to match the processing speed of the task being performed. Thus, for a minimal amount of additional hardware and software, the programmer can improve battery life by dynamically scaling the clock based on the computational load.

Simple on-chip scalable clocks, such as those found on Zilog's Z180, divide the crystal frequency to derive the CPU clock. Although this scalable clock is simple, the oscillator still operates at the nominal frequency, consuming a steady current.

Some larger microprocessors (e.g., Motorola's MC68328 DragonBall and AMD's Élan 386) have an internal PLL that can step up the frequency from a 32.768-kHz watch crystal to megahertz speeds. The PLL permits a good deal of control over the operating frequency, enabling complex power-management schemes. An external 32.768-kHz crystal requires lower quiescent current than megahertz-speed crystal oscillator circuits.

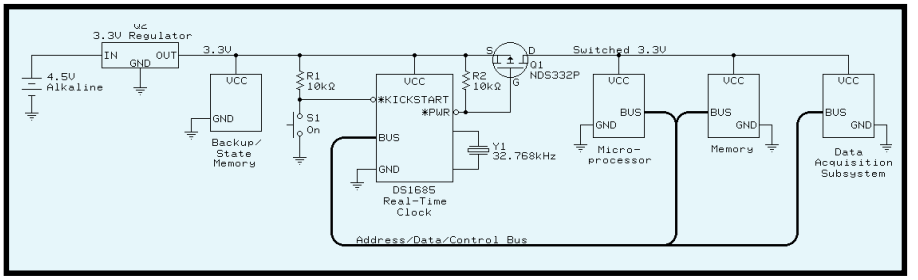


Figure 2—*The National Semiconductor NDS332P (a p-channel FET rated for operation at 3.3 V) gates the power supply to most of an embedded system under the control of a real-time clock with a power-control pin.*

SLOWING THE CLOCK

In many applications, you want to be able to stop the clock completely, and most microprocessors have this feature. Modes that turn off the clock (referred to as sleep, doze, snooze, shutdown and halt) are usually invoked by writing to an internal I/O register or executing a special instruction. Normal operation is restored by a stimulus event such as an interrupt (external, or internal if onboard peripherals are permitted to operate) or reset.

Let me describe three possible zero-frequency clocking modes based on their technical differences. I use generic labels because processor datasheets use no consistent terminology.

In mode 1, the oscillator continues to operate, but the core is not clocked. In mode 2, the oscillator is off, but the micro is still powered. And in mode 3, power to the processor is removed completely.

The advantage of mode 1 is that it can respond quickly to a stimulus event, even though current consumption is high for a sleep-type mode (~1 mA, depending on frequency).

In mode 2, current consumption is reduced to the quiescent current of the microprocessor and other components, but it takes a lot of time to restart the processor. It's usually controlled by the reset generator, which has a 50- or 200-ms pulse. That's a generous amount of time for the oscillator to restart, but a long time for a real-time event.

In mode 3, the quiescent current of the upowered components is eliminated entirely by disconnecting power to the system via a switch like a p-channel FET. However, some additional circuitry is required to switch power. Also, power-supply bus isolation becomes a problem.

Modes 1 and 2 rely on the features of the microprocessor, but the power-savings mode 3 can be implemented using any microprocessor and it affects the entire system.

In modes 1 and 2, the restart depends on the microprocessor. In mode 3, the restart is implemented in hardware controlling your switch.

Figure 2 shows an implementation example of mode 3. Power is controlled by the real-time clock, a Dallas Semiconductor DS1685. The microprocessor can set a bit in a DS1685 control register that tristates the *PWR pin, which disables the FET, effectively shutting off power to the system core.

If so configured within the RTC, the *PWR pin is pulled low by a real-time clock event (e.g., a preprogrammed alarm) or an external stimulus on the

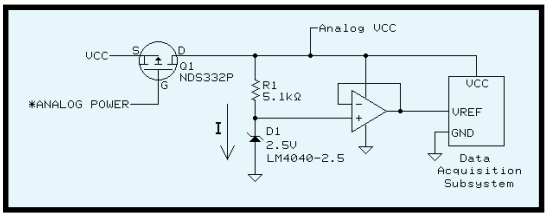


Figure 3—*A series shunt is an alternate to a low drop-out reference since the selection of the latter is limited. The series resistor must be small enough to permit the shunt reference to be biased to a region where the output is stable.*

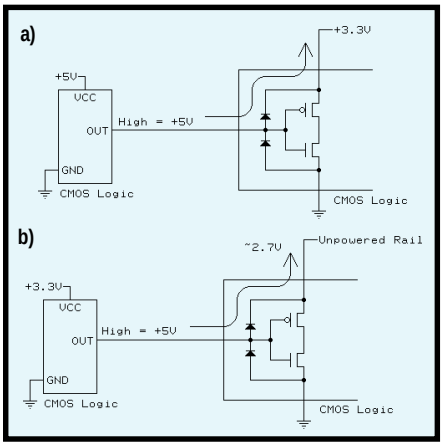


Figure 4a—*When a 5-V–powered output drives a 3.3-V–powered input, a large current flows through the input-protection diode into the 3.3-V supply rail.* ***b—****When a powered output drives an unpowered input, current flows into the unpowered rail, turning on ICs that were intended to be off.*

*KS (kickstart) input (e.g., the On button being pressed). The p-channel FET Q1 must be specified to be on with a V_{GS} of 3.0 V or less. A few FETs are available, such as National's NDS332P.

As a novel alternative to putting the unit to sleep as in mode 1, consider slowing the clock to the tens to hundreds of kilohertz range. Power consumption at these clock frequencies may be comparable to sleep-mode operation and the problems of oscillator restart time are avoided. While in slow-speed mode, the micro performs a simple sampling task, like monitoring the keypad for user input or sampling the real-time clock to wake the system at a specified time.

CAPACITANCE

Power dissipation is also proportional to capacitance. The term C from the power equation is the sum of C_{PD} and C_{IO}. C_{PD} is the effective, lumped internal capacitance that is associated with the power dissipated internally by the IC. C_{IO} is the capacitance associated with power dissipated outside the IC.

C_{PD} depends completely on IC characteristics and is beyond your control. You have control over C_{IO}. However, it's usually a function of PCB traces loading and other IC loads, which are constrained by other aspects of the design. Keep excess capacitive load to a minimum where possible.

COMPONENT SELECTION

Using a 3.3-V supply and reducing the clock speed to a minimum gets the biggest reduction in power consumption. However, careful IC selection, use of shutdown modes, power-supply design, and analog design also help you squeeze extra life out of the systems' batteries.

As opposed to 5 V, with low-power design, you must be concerned with the availability of 3.3-V parts and the specified power consumption. These two considerations make IC selection for a low-power design trickier than conventional 5-V design.

Not all ICs specified for operation at 3.3 V are appropriate for a battery-powered system. Compare the specified current consumption for parts you are considering for your design. Sometimes IC manufacturers are inconsistent about the conditions power consumption is specified in.

When you compare power-consumption figures, make sure the clock frequency, supply voltage, and temperature are similar. Some PLD manufacturers specify current consumption at 0°C. But, this is not meaningful for most systems since CMOS power consumption increases with increasing temperature.

POWER SUPPLY

Choosing a power supply is critical in a low-power system. Although you may think of linear regulators as inefficient, they're often a good choice for a battery-powered system, particularly because the difference between the input (your battery supply) and the output is typically low.

Some recent offerings by IC makers include low-dropout (LDO) and low-quiescent current parts. LDO linear regulators may remain in regulation when the V_{in} to V_{out} difference is 100 mV or lower, depending on the current. If your system supply voltage is 3.3 V, the input voltage from your batteries can drop as low as 3.4 V and the system operates normally.

The apparent efficiency of a linear regulator can be comparable to that of a switcher. For example, if your power source is three alkaline batteries, the nominal battery voltage is 4.5 V, and the nominal supply is 3.3 V, then the initial efficiency is:

$$\frac{3.3\ V}{4.5\ V} = 73\%$$

(The calculated efficiency increases as the battery discharges and voltage decreases.) A typical switching power supply is 60–80% efficient and needs more components than a linear supply.

Many LDO regulators require a high-value, low equivalent series resistance (ESR) capacitor on the output, such as a 4.7-µF tantalum, to stabilize the internal feedback. A noteworthy exception is the anyCAP line of LDOs (ADP330*x*) from Analog Devices. According to the manufacturer, the ESR of the filter capacitor is unrestricted and its value can be as low as 0.47-µF.

An LDO regulator may not necessarily have low quiescent current. If your design requires the availability of regulated power in sleep mode, pay attention to the quiescent current. If the power supply has high quiescent current and is continuously on, the batteries could have a shorter-than-expected shelf life.

If your design requires a battery supply voltage that is lower than the supply (e.g., two alkaline cells in series provide 3.0 V when fresh, less than the typical 3.3 V), a boost-topology switching power supply is your only choice.

Given your design requirements, it may not be possible to operate strictly at 3.3 V. A 5-V power supply may be needed for some subsystems. Components such as LCDs and certain analog ICs are not easily found at 3.3 V.

SHUTDOWN MODES

Many ICs providing I/O functions (e.g., RS-232 level shifters or ADCs) have shutdown modes that reduce the power consumption to microamps.

Some ICs have a pin dedicated to this function. In others, shutdown mode is engaged by writing to a specific register. Thus, the micro intelligently controls the use of power in the system by shutting down a subsystem not in use.

LOW-POWER ANALOG

Design of analog sections can be tricky with a reduced supply voltage. Component selection plays a significant role in your design.

Op-amps specified to operate from a single 3.3-V supply abound. With op-amps, keep an eye on the specified current draw. Generally speaking, op-amps trade off power consumption with frequency response.

To minimize op-amp power consumption, select one with the lowest possible frequency response that your design can accommodate. Table 1 lists some op-amps that trade current consumption with frequency response.

Another critical parameter for op-amps is maximum and minimum output voltages. With rails of only 3.3 V and ground, the op-amp must swing as close to the rails as possible to have a useful output voltage range. Fortunately, newer op-amps specified for lower voltage operation can also drive their output voltage within 10 mV of the rails.

A reference is a critical part of the analog subsystem. For a low-voltage design, you may select a reference voltage of 2.5 V. When operating from a 3.3 V supply ±10%, the reference must tolerate V_{CC} as low as 3.0 V, so a dropout of more than 0.5 V is unacceptable.

Analog Devices offers the ADR291, a 2.5-V, 5-mA output precision voltage reference with a maximum quiescent current of 15 µA at 25° C.

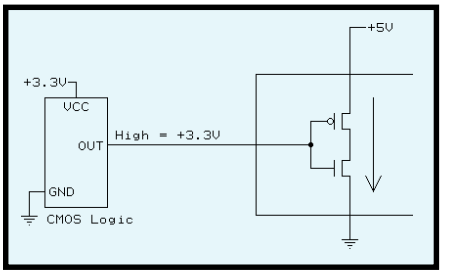


Figure 5—*When a 3.3-V–powered output drives a 5-V–powered input, both input stage transistors conduct, forming a current path between +5 V and ground and dramatically increasing power consumption.*

An alternative to sole-source buffered output voltage reference is a shunt reference. With shunt references, a series resistor supplies the reference. Careful selection is important here because some shunts require higher current to maintain the output voltage within tolerance.

For example, National Semiconductor's LM4040 needs an external resistor. The forward current across the IC is specified to be 100 µA to achieve a flat output voltage over temperature.

With a minimum V_{CC} of 3.0 V, the series resistor value should be:

$$\frac{3.0\ V - 2.5\ V}{100\ \mu A} = 5000\ \Omega$$

(but use 5100 Ω; see Figure 3).

Also, a p-channel FET can turn off the reference's power when it's not in use. An op-amp buffer provides sufficient current to drive the analog signal conditioning and converter.

An LDO linear regulator can be used as low-accuracy alternative to a conventional reference, although you have to program the output voltage using external using feedback/sense resistors.

A variety of 3.3 V-only ADCs and DACs are available from Texas Instruments, Analog Devices, Linear Technologies, and others. The TLV2543 from Texas Instruments is a low-cost, 12-bit ADC with 11 input channels as well as a serial interface and a software-programmable shutdown mode.

	Op-Amp	Typical Current Consumption per Op-Amp at 25°C (µA)	Gain Bandwidth Product (MHz)
Analog Devices	OP193	15	0.035
Burr-Brown	OP336	20	0.1
National Semiconductor	LMC6462	20	0.05
Analog Devices	OP191	220	3
Burr-Brown	OPA340	750	5.5

Table 1—*With op-amps, you can tradeoff current consumption with frequency response by way of part selection.*

DIGITAL LOGIC

Almost every design needs some logic to glue the processor and subsystems together. Modern CMOS discrete logic families have low dynamic power consumption and almost immeasurable static power consumption. Various logic families are available, but they usually trade power consumption for speed. Pay attention to the specified supply voltages.

The 74VHC family, available from National Semiconductor and Motorola, is a nice choice. It has low power consumption (see Table 2), it's fast enough for many microprocessor applications (~15 ns for buffers and latches, ~10 ns for gates), and it tolerates 7 V on inputs even if the supply is 3.3 V, which is a help when interfacing to 5-V logic.

You may be tempted to use a PAL or EPLD in your design. However, most programmable logic has high power consumption.

"Quarter power" and "zero power" PALs are available from some IC vendors, but since these have microamp static power consumption, the dynamic power consumption can be quite high (see Table 2). Some PLDs are designed to improve power consumption, but sometimes specifications are ambiguous or only specified at 0°C.

Table 2 compares manufacturer's specified power consumption for various logic parts, operated at 3.3 V. It gives a rough comparison of the power consumption for different types of logic.

Table 2 doesn't accurately compare logic implementations because each entry doesn't have an equivalent amount of logic and PLDs are not programmed with the same logic functions. Note: Philips specifies at 0°C, making their current consumption figures appear more favorable than the competition. And, AMD PALs are now sold under the Vantis name.

MIXED-RAIL INTERFACING

Mixed-rail interfacing (i.e., interfacing logic driven by both 5- and 3.3-V supplies) can be fairly tricky. It offers no inherent advantages, and single-supply systems are easier to design.

But, when component selection dictates you must have a 5-V rail, you need to think about mixed-rail interfacing. Both 3.3 V to 5 V and 5 V to 3.3 V have interfacing difficulties.

When a 5-V output drives the input of a 3.3-V–powered part, unnecessary current flows from the 5-V output pin into the 3-V supply via the input protection diode (see Figure 4a). Although solutions using open-drain outputs, series resistors, or diodes exist, a clean solution is to use a logic family such as 74VHC, which can tolerate inputs as high as 7 V independent of the supply voltage.

The same problem is encountered when interfacing between powered and unpowered parts. If a powered part with a high output drives an unpowered part, then current conducts through the input protection diode, powering the supposedly unpowered rail (see Figure 4b).

Interfacing a 3.3-V output to a 5-V–powered input, as in Figure 5, is also a problem but for a different reason. When the input to the 5-V–powered part is ~3.3 V, the input stage transistors are both in their linear region and a current path from V_{CC} to ground is formed within the part. This results in current consumption 10 or 100 times worse than the quiescent current.

In some cases, this current of 1–2 mA per pin may be acceptable. A solution is to use an open-drain output with a pull-up resistor, but speed and current drain through the pull-up resistor may still be unacceptable.

Probably the best approach to 3.3 V to 5 V interfacing is to use dual-supply interface ICs designed for this purpose. These include some of the 74LVX line from National Semiconductor.

MEMORIES

If your micro needs external memory, there's good news and bad news. The good news is that SRAMs are widely available and specified for 3.3-V operation, even at access speeds of 70 ns.

	Typical Supply Current at 0 MHz (µA)	Typical Supply Current at 15 MHz (mA)	Temperature Specified (°C)
National Semiconductor 74VHC373	4	1.5	25
Philips 32-Macrocell EPLD	10	3	0
Atmel 32-Macrocell EPLD	3000 (5 with shutdown pin)	30	25
Atmel Zero Power PAL	5	55	25
Vantis Zero Power PAL	30 (85°C)	30	25
Vantis Full Power PAL	40,000	45	25

Table 2—*Although there's a large amount of variation in the power consumption of programmable logic, it does consume more power than discrete logic.*

The bad news is that this is not true for ROMs. Low-voltage EPROMs and flash memory are easy to find, but the fastest access speeds aren't on par with their 5-V brethren. You may have to use a slower microprocessor clock or insert wait states to meet timing requirements for low-voltage nonvolatile memory.

ON TIME

It's difficult to predict the length of time your device will operate on a single battery charge. First, estimate the system's power consumption.

Usually, the maximum rated power consumption that you find in the datasheet is much worse than what you measure on the bench. Remember that the datasheet rating usually applies over the full temperature range and your bench measurement is performed at room temperature.

The other consideration is the amount of power available from your batteries. But, the capacity of the system's batteries can vary fairly widely, even between batteries in the same manufacturing lot.

The best way to predict time of operation is to measure the length of operation under a variety of different operating conditions with different batteries.

Battery capacity increases with decreased drain current. The lower the current draw from a battery, the more total power is available from that cell. Thus, as your design draws less and less current, the application runs even longer.

LIVE LONG AND PROSPER

Many designs are constrained to use batteries because the application requires portability or is located remotely. To maximize battery life, the system shouldn't squander power when it isn't needed.

If the application is a hand-held instrument, lower power consumption means that batteries can be smaller, lighter, and last longer. If the application is a remote data logger, the batteries do not have to be serviced as often. If you are designing a robot, more available battery power can be allocated to motors, resulting in a faster or higher torque device.

Brian Kurkoski is the hardware engineering manager for Z-World. He developed many of the ideas in this article when designing the LP3100, a C-programmable embedded controller with low power consumption. You may reach Brian at kurkoski@zworld.com.

REFERENCES

National Semiconductor, *National VHC Advanced CMOS Logic Databook*, 1996.

C. Small, "Batteries Explode into New Applications and New Chemistries," *EDN*, 13 October 1994.

J. Williams, "Low Voltage Embedded Design," Intel Corp., App. Note 272324-001, February 1993.

SOURCES

Élan 386
Advanced Micro Devices, Inc.
One AMD Pl.
Sunnyvale, CA 94088-3453
(408) 732-2400
Fax: (408) 732-7216
www.amd.com

OP191, OP193, ADCs, DACs
Analog Devices
One Technology Way
Norwood, MA 02062-9106
(781) 329-4700
Fax: (781) 461-4261
www.analog.com

EPLDs, PALs
Atmel Corp.
2324 O'Nel Dr.
San Jose, CA 95131
(408) 441-0311
Fax: (408) 436-4300
www.atmel.com

OP336, OPA340
Burr-Brown Corp.
6730 S. Tucson Blvd.
Tucson, AZ 85706
(520) 746-1111
Fax: (520) 889-1510
www.burr-brown.com

DS1685
Dallas Semiconductor
4401 S. Beltwood Pkwy.
Dallas, TX 75244-3292
(972) 371-4448
Fax: (972) 371-3715
www.dalsemi.com

3.3-V ADCs and DACs
Linear Technology
1630 McCarthy Blvd.
Milpitas, CA 95035
(408) 432-1900
Fax: (408) 434-0507
www.linear-tech.com

74VHC*xxx*, MC68328 DragonBall
Motorola Literature Distribution Ctr.
P.O. Box 20912
Phoenix, AZ 85036
(602) 244-6900
Fax: (602) 332-3944
www.mot-sps.com

74VHC*xxx*, LMC6462, NDS332P
National Semiconductor
P.O. Box 58090
Santa Clara, CA 95052-8090
(408) 721-5000
Fax: (408) 739-9803
www.nsc.com

EPLDs
Philips Semiconductor
811 E. Arques Ave.
Sunnyvale, CA 94088-3409
(408) 991-3737
Fax: (408) 991-3773
www-us2.semiconductors.philips.com

3.3-V ADCs and DACs, TLV2543
Texas Instruments, Inc.
34 Forest St.,MS 14-01
Attleboro, MA 02703
(508) 699-5269
Fax: (508) 699-5200
www.ti.com

AMD PALs
Vantis Corp.
995 Stewart Dr.
Sunnyvale, CA 94088
(408) 616-8000
Fax: (408) 616-8010
www.vantis.com

Z180
Zilog, Inc.
210 E. Hacienda Ave.
Campbell, CA 95008-6800
(408) 370-8000
Fax: (408) 370-8056
www.zilog.com

I R S

404 Very Useful
405 Moderately Useful
406 Not Useful

Design98 Contest

PIC of the Lot

"The designs submitted were quite professional and far more complex in nature than my wildest imagination. It was clear to me that all of the contestants in this contest are winners." Contest Judge John Martinelli, Director of Advanced Development, Interlink Electronics Inc.

Your response to **Design98** made this contest more enjoyable than any of the others that I've coordinated to date. I received a ton of E-mail with questions ranging from "How do I get an entry form" to "I designed this cool <big long explanation of the gadget> for my company. May I enter it?"

I got phone calls, "Hello, this is Alvin Tan from Singapore. Can internationals enter your contest?" My response, "Can you cash an American-currency check?"

I received letters. "Thanks for running such an interesting contest!!" "Thank you for promoting such a contest." "Thank you for running a great contest. I had fun designing this entry and seeing it finally work."

And then, to top it all off, I got a rabbit colony full of entries. Entries were bursting out of the corners. I was swimming in emptied FedEx envelopes.

In other words, YOU made **Design98** great! Your response made all of the difficulties of putting a contest together worth it. Thank you for stepping up to the plate and making this the very best *Circuit Cellar INK* contest we've ever had. You are what made it work.

—Janice Hughes
Very Proud Contest Coordinator

BEST OVERALL

$5000, HP Mixed Signal Oscilloscope, $500 for using PIC16F84

XY LCD Graphing Data Logger—Alberto Ricci Bitti, Lugo, Italy

Do you need sophisticated graphical display of data immediately regardless of whether you're in the lab, on the production line, or in the field? If so, you need to check out the XY LCD Graphing Data Logger.

Alberto brings us a hand-held data logger that is capable of displaying data graphically on a 64×128 LCD screen and of performing analytical (e.g., integration and derivation) and statistical (e.g., linear, exponential, polynomial regressions, mean, and deviation) math processing. At up to eight inputs, its voltage is read with 1-mV resolution in the 0–4096-mV range through a 12-bit ADC.

How does he do it? By using a Casio graphing calculator that sells for about the same amount as an LCD and interfaces through a miniature jack, a PIC16F84, and a MAX186 ADC. The software manages the communication protocol over a serial line while also combining sleep mode and watchdog techniques to achieve minimal power consumption, low-power ADC operation, and data conversion to the Casio format.

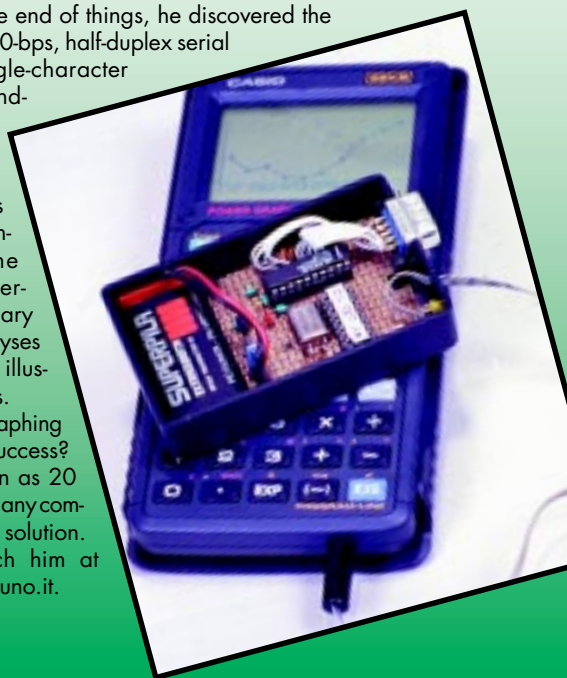
But first, let's talk about why he chose the PIC16F84. The chip supports data tables as long as the program memory thanks to its `RETLW` instruction. It is cheap and powerful enough to handle serial communications entirely in software.

The next most important chip was the MAX186 A/D converter. He realized that to get stable readings of the least significant bits, he had to keep the digital ground and analog ground separate, joined only at one point near the regulator. The MAX186's good pin layout helped divide input lines from data lines, making a stable reading easily obtainable.

To maximize battery life, Alberto used a low-voltage power regulator. By putting the micro and ADC in sleep mode, he found that the unit only requires 50 µA.

The entire circuit is assembled using a prototyping board that fits inside a small plastic box as large as the calculator. While operating, the box is affixed to the bottom of the Casio calculator via removable biadhesive strips. This arrangement keeps the calculator stable and on a comfortable slope.

On the software end of things, he discovered the Casio used a 9600-bps, half-duplex serial stream with single-character ACK/NACK handshake terminated by checksums. The logger collects data and passes tables of variable information to the Casio, which performs the necessary mathematical analyses and graphically illustrates the functions.



Is the XY LCD Graphing Data Logger a success? You bet. It rings in as 20 times cheaper than any comparable PC-based solution.

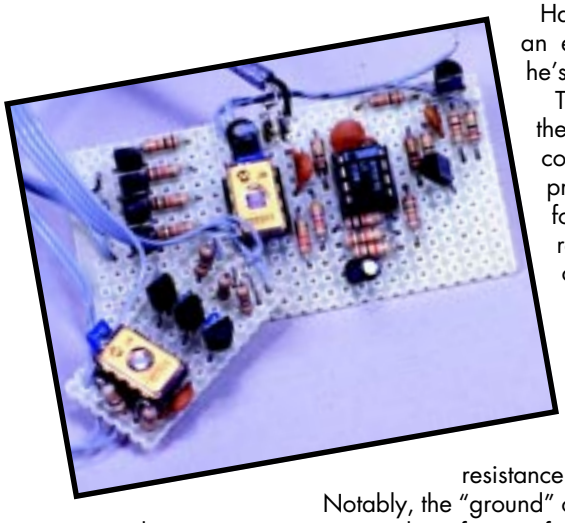
You may reach him at a.riccibitti@ra.nettuno.it.

PIC12CXXX

First Place

$3000, HP LogicDart, $500 for using PIC12C508

Guitar Effects Controller—Hank Wallace, Fincastle, Virginia



Hank is an avid guitar player as well as an electronics addict. With this project, he's able to bring both his hobbies together.

The problem: when playing live, one of the most frustrating tasks for guitarists is controlling their sound. Typically, the problem is solved by placing an array of foot switches on the stage. But, that restricts the player to one spot—an unacceptable constraint for rock and roll.

Instead, Hank mounted three inexpensive aluminum-tape pads to the pick guard. A small transmitter under the guitar's phone jack senses touches on the three metal pads. A transmission is triggered by the resistance of the body from the pad to ground.

Notably, the "ground" of the transmitter is actually the VCC of the microprocessor since the reference for the pads is not ground, but the +3-V supply of the transmitter. While this may sound like a problem, it's not since the transmitter output is AC coupled to the guitar audio path.

As the guitarist touches a pad, the contact conducts enough current to turn the transistor on and awaken the PIC12C508. After a 120-ms debounce delay, the processor qualifies the three inputs, produces a 50-kHz pulse train, and sends it along the guitar cable to the receiver mounted inside a commercial DSP-based guitar-effects processor.

The receiver decodes the databurst, determines which key pads were activated, and then connects one or more foot-switch terminals in the processor to ground for 250 ms, thereby simulating the foot-switch operation.

He may be reached at hank@aqdi.com.

Second Place—$2000

Internet Appliance—Myron Loewen, Niverville, Canada

Lots of companies today like to boast about having the smallest Internet server, and while theirs are impressive, Myron puts all them all to shame.

With an 8-pin PIC12C672, Myron handles all the tasks of dialing an ISP, negotiating PPP configuration and compression options, negotiating IP configuration options, authenticating itself with PAP, finding its IP address, implementing PING protocol, and implementing TFTP.

On the hardware end of things, the only real difficulties came in determining which port to use for which functions so that the number of analog inputs was maximized.

And in the software department, you'll find a lean-mean state machine. As packets arrive, they are examined to see if a reply or state change is required. And, if nothing arrives within a set amount of time, each state has its own job to do.

He may be reached at myron@norscan.com.

Third Place—$1000, $500 for using PIC12C508

R/C Brakes—Greg Mrozek, New Hope, Minnesota

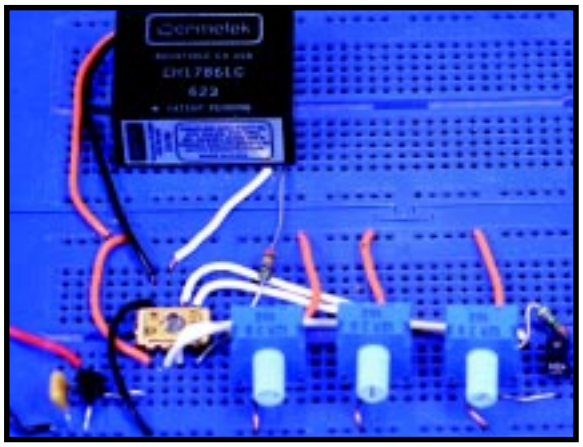
Even when you're racing radio-control cars, you can't forget to use the turning signal. And, woe betide the car that tailgates if your brake lights aren't up to snuff.

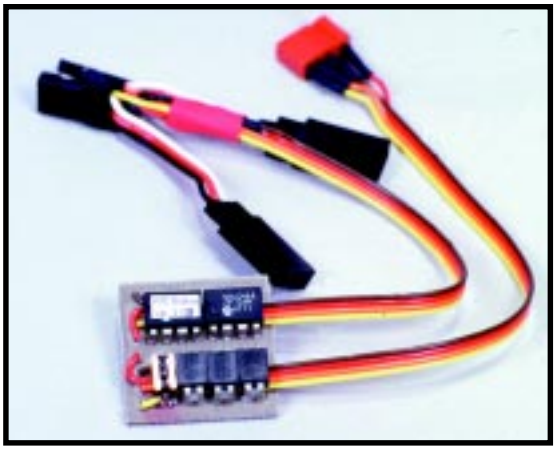
But, Greg fixes all that with R/C Brakes, a brake light and/or turn-signal controller for radio-control cars. The device takes input and power from the servo outputs of a standard radio-control receiver and uses these signals to control the brake lights and turn signals mounted on the body of a radio-control car.

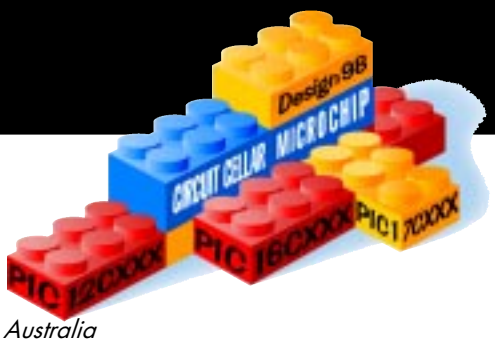
The RX signal from the servos is normally low with a high-going pulse length of 0.8–2.1 ms. A software capture algorithm captures this pulse width and then uses the information to control the tail lamps.

As well, R/C Brakes is completely configurable, enabling it to operate correctly on almost any radio-control car. You can program the device through a single push button, and the configuration data is stored in a serial EEPROM.

You may reach Greg at mroz0012@tc.umn.edu.



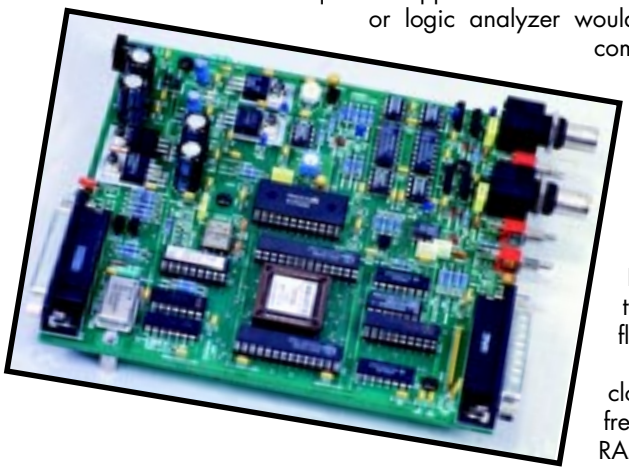




First Place

$3000, HP Logic Dart, $500 for using PIC16F84

BitScope Mixed Signal Capture Engine—Norman Jackson, Stanmore, Australia



The BitScope Mixed Signal Capture Engine is an RS-232 peripheral device intended for data-acquisition applications where a digital sampling oscilloscope or logic analyzer would be used. It is capable of complex triggering and simultaneous recording of analog data and digital logic states as well as time/frequency measurements. This 50 MS/s digital signal oscilloscope and logic-analyzer capture engine is constructed using a PIC16F84, a Lattice 1016 PLD, two 32-KB cache RAMs, and a flash ADC.

A PIC-controlled synchronous clock circuit allows single-step or free-running operation of sample RAM. High-performance op-amps and a wide-bandwidth ADC chip provide an analog bandwidth close to 100 MHz, so you can use subsampling techniques to reconstruct repetitive waveforms above Nyquist's sampling frequency.

BitScope is programmed via scripts from a host attached to the serial port. The scripts are virtual machine instructions that synthesize the required functionality.

The PIC communicates with the host via a standard serial communications link, so anything from a palmtop to a workstation can interface to it. According to Norman, this excessive computing power should enable the construction of a complex virtual instrument from a small set of capture primitives implemented in the BitScope microcontroller.

You may reach Norman at bitscope@discrete.net.

Second Place—$2000

Hydrocare Hydrotherapy System—Troy Harvey, Provo, Utah

Feeling a little tight and achy? Troy has the bed for you. The hydrocare Hydrotherapy system is a therapeutic massage bed that provides a full body massage via high-pressure warm-water jets without getting the user wet. A gel-filled sound-dampening waterproof membrane fully relays the water pressure and keeps the user dry.

Hydrocare's electronics are controlled by a PIC16C74A. The chip coordinates many functions: a hand-controller keypad and LCD, a real-time session timer, an external timer interface, a token-vending interface, a user-definable menu with hardware lockout and EEPROM backup, water-jet pump control, a variable-speed PWM water-jet drive motor control, a water-jet position LCD readout, a water-jet memory setting, a multilevel water-jet pressure control, a water temperature control, water-tank level sensing, and a nonvolatile EEPROM-based machine-use odometer.

With a PIC, the cost of the electronic subsystem fell from $562 to $143. What I want to know is how much would it cost to adapt it to my waterbed?

You may reach Troy at taharvey@itsnet.com.

Third Place—$1000

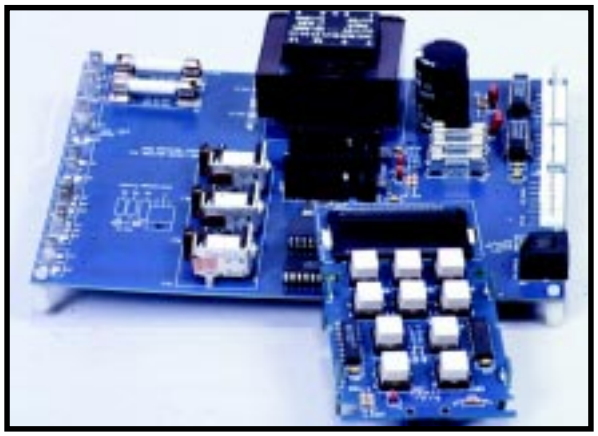
Mobile Environmental Control from a Wheelchair—Dan Leland, Coquitlam, Canada

Dan wants to enable people with little or no limb function to access common household appliances without assistance. His goal: to design and build a simple scanning input device using the PIC16C84 that would serve as the front end for a versatile universal IR remote. With the device, the user could access a variety of TV and stereo components as well as X-10-enabled devices from anywhere in the home.

Dan first built a Scanning User Interface (SUI) with a 4 × 10 LED array, corresponding to the buttons on the remote. The device has a variety of ability control switches and operating modes that let the user make a selection.

The user scans up, down, and across the LEDs via a sip-and-puff pneumatic switch (other switches—blink, finger-flex, head tilt, or oversize button/paddle—can also be used). Once a selection is made, the SUI sends the appropriate serial control command through a three-wire cable to the IR remote.

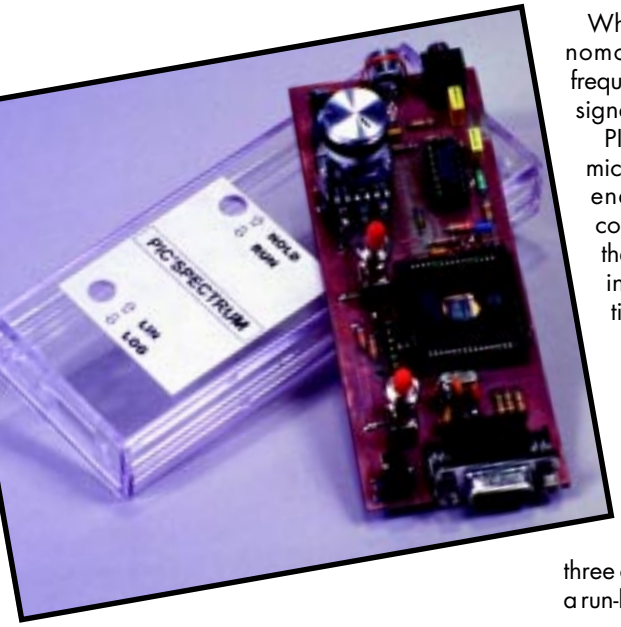
You may reach him at rd@mindlink.bc.ca.



First Place

$3000, HP LogicDart, $500 for using PIC17C756

PIC'Spectrum—Robert Lacoste, Chaville, France



What is PIC'Spectrum? It's an autonomous device able to display the frequency decomposition of an audio band signal.

PIC'Spectrum is a small box with a microphone or audio line input at one end and a standard VGA output connector on the other. It displays on the video monitor the spectrum of an incoming sound and updates it in real time (e.g., 10 refreshes per second).

Of course, PIC'Spectrum is a digital spectrum analyzer. The incoming signal is first digitized and its spectrum computed with a 256-point Fast Fourier Transform. The corresponding video image is software-generated on-the-fly.

This compact device has only three controls: a signal-level potentiometer, a run-hold switch, which freezes the display, and a log/lin switch, which switches the spectrum amplitude display between logarithmic and linear scales.

And, there's no PC. Apart from a voltage regulator and a quad op-amp for signal amplification, the only other active component is the PIC17C756. The PIC has enough internal resources to do everything from signal acquisition to video generation through real-time FFT. The PIC replaces the ADC, DSP, and video generator, which make up the classic design of a digital spectrum analyzer.

The PIC17C756 may not have an integrated video controller, but it does have enough horsepower and program memory to do software-generated video. All by itself, PIC'Spectrum directly drives the VGA CRT at a high level of quality.

You may reach him at rlacoste@nortel.com.

Second Place—$2000

Sky Scanner—Adam Lyness, Christ Church, New Zealand

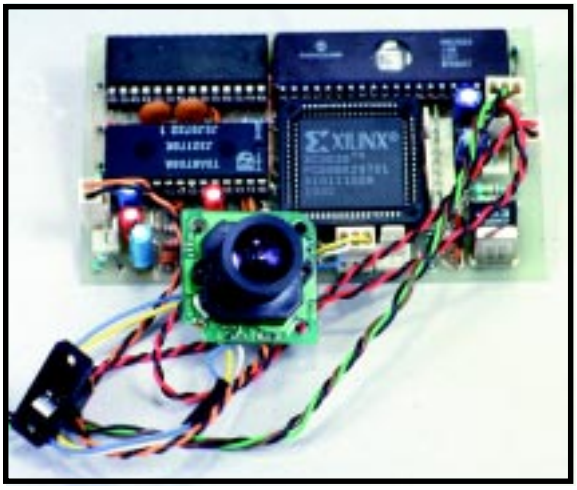
So just where does the sky meet the ground?

With Sky Scanner, you can find out. Adam uses a miniature CCD camera and logic to capture a video image of the horizon. Based on the slope and offset of this line, the PIC17C44 determines the attitude of the sensor.

While conventional tilt sensors can do essentially the same thing, Sky Scanner has a distinct advantage. It's immune to acceleration. It uses the horizon, not gravity, as a fixed reference. It measures 360° of roll, 70° of pitch, and processes about 25 updates per second.

Its relative accuracy stands at ±1° and its absolute accuracy is limited only by the presence of obstructions.

You may reach Adam at arl37@cad.canterbury.ac.nz.



Third Place—$1000

Spectra Post Processor—Reid Wistort, Westford, Vermont

Reid made this project not to win a contest, but to hear better. In 1989, he received a cochlear implant, which consists of a body-worn speech processor and a surgically implanted receiver/stimulator that includes an electrode array placed in the snail-shaped cochlea, deep in the skull. Essentially, a microphone in the earpiece picks up a sound, which the processor analyzes, digitizes, and encodes into an RF signal, which is then passed through the skin to the implanted receiver.

But despite many adjustments, Reid's speech recognition was sporadic and he could not use the phone. He realized that the EEPROM settings on his implant were wrong.

The solution: he uses a three-processor system (PIC17C44, '17C42, and '16C74), which intercepts, decodes, and modifies the RF datastream from his bodyworn implant speech processor before reconstructing a modified datastream that is applied to the implanted portion of the cochlear implant.

He may be reached at reid@btv.ibm.com.

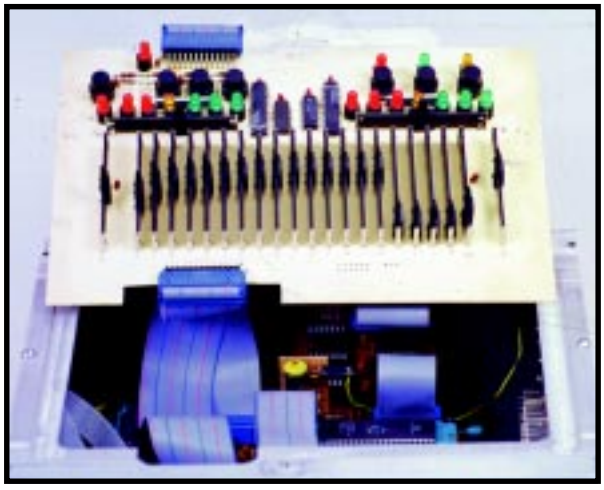


Photo courtesy of QNX Software Systems, Ltd.

NPC

PCI DATA-ACQUISITION BOARDS

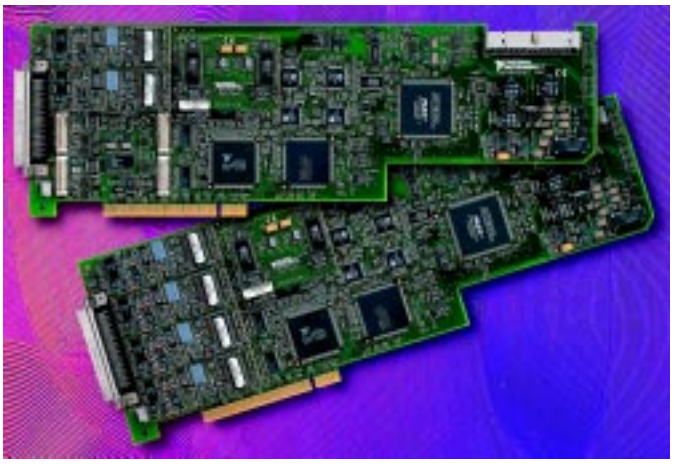
The **PCI-6110E** and **PCI-6111E** are high-speed, simultaneous-sampling, multifunction plug-in data-acquisition boards that have a 12-bit ADC. The boards are PCI-bus masters and sustain 5 MS/s per channel without burdening the CPU. They're ideal for applications requiring high-speed, simultaneous acquisition, such as vibration analysis, transient captures, phase measurements, spectral analysis, and electronics testing. Both boards are compatible with a wide variety of industry-standard application software, including LabVIEW, LabWindows/CVI, ComponentWorks, Measure, and VirtualBench.

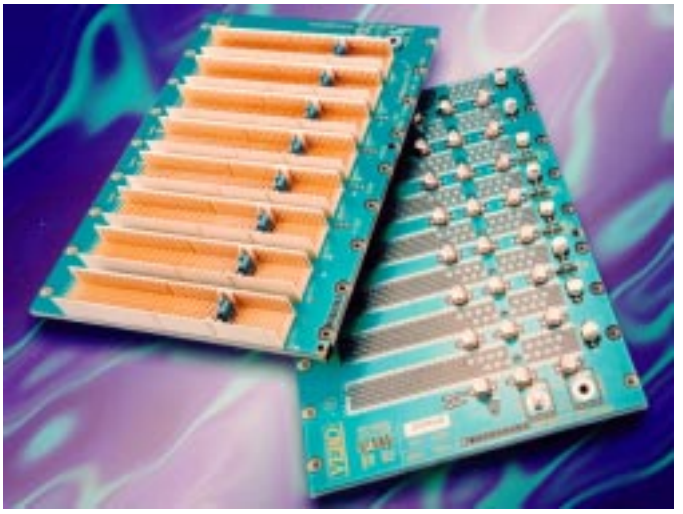
The PCI-6110E and PCI-6111E feature four or two differential analog inputs, respectively, with a 12-bit ADC per channel. The voltage input range is bipolar and software-configurable up to ±42 V. Software-programmable gains include 0.2, 0.5, 1, 2, 5, 10, 20, and 50. Both boards feature two 16-bit analog output channels, eight lines of TTL-compatible digital I/O, and two up/down 24-bit counter timers. All of these features can be user synchronized for precise timed acquisition and generation. The boards use a custom-programmable instrumentation gain amplifier to ensure high-quality measurement and excellent accuracy.

The PCI-6110E and PCI-6111E sell for **$2995** and **$2195**, respectively.

National Instruments
6504 Bridge Point Pkwy.
Austin, TX 78730-5039
(512) 794-0100
Fax (512) 794-8411
www.natinst.com

#510





CompactPCI-COMPLIANT BACKPLANE

VERO's new family of 10-layer **cPCI backplanes** consists of four versions in 3U and 6U formats. All models support geographic addressing, hot-swap pin sequencing, and 32-/64-bit interoperability, with pin allocations and lengths defined for future insertion (currently at Draft 0.5 status) and with clock routing changes according to V2. Standard models operate at 33 MHz and are selectable for operation at 66 MHz. Models are user-selectable for operation between 5 and 3.3 V. Power can be brought onto the board using power bugs, a bus bar, Fastons, or a standard ATX psu–format connector header.

The entry-level backplane is a 3U, 32-bit, eight-slot unit, which reduces cost by limiting the power-on option to Fastons only. For powerful computing needs, a 64-bit backplane is available in 3U and 6U formats without support for rear plug-up modules. This unit, which offers bandwidths up to 264 MBps, is useful for semicustom applications where additional space is useful in combining a proprietary bus into the standard cPCI on a monolithic backplane.

The development systems, which can be fitted with any of the cPCI backplanes, consist of a VEROtec half-width case, a KM6-RF subrack, and an ATX psu. Thermal management and EMC screening are available. The systems also feature a power-sequencing module to ensure that power rails are presented and removed in the correct order at powerup and shutdown.

Fully specified cPCI backplane and development systems start at **$400**.

VERO Electronics, Inc.
5 Sterling Dr.
Wallingford, CT 06492
(203) 949-1100 • Fax: (203) 949-1101
www.vero-usa.com **#511**

NouveauPC

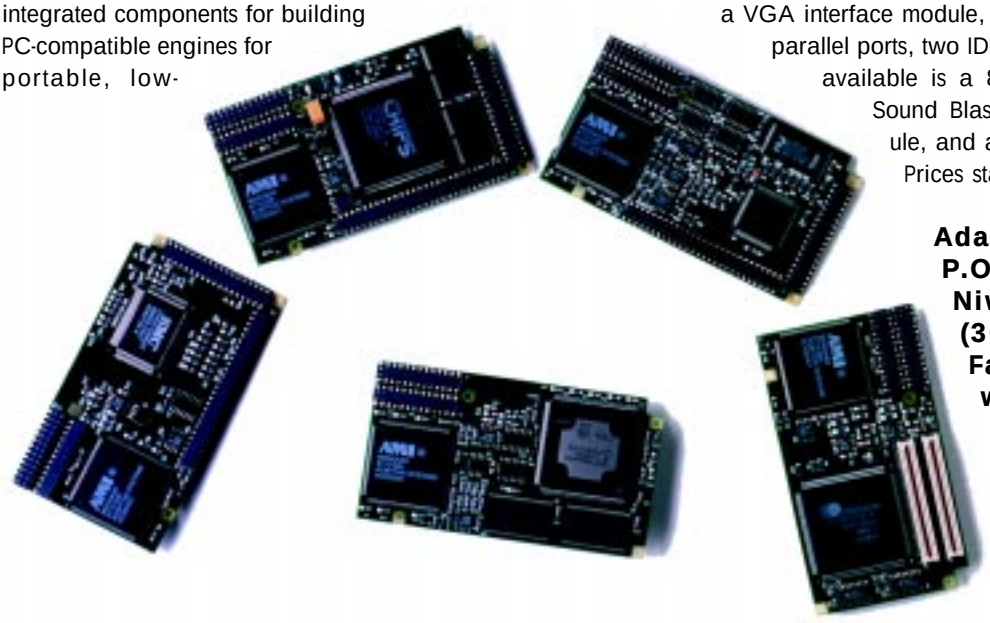
edited by Harv Weiner

PC MODULES WITH NEW FORM FACTOR

Adaptive Systems has designed a set of off-the-shelf 2″ × 3.5″ printed-circuit assemblies that can be combined to offer a custom PC "motherboard" in a variety of physical configurations. This patented interconnect strategy provides unparalleled mechanical flexibility and improved system reliability by eliminating backplanes, internal cables, and complex mounting assemblies. The **nC52** provides increased performance over PC/104 modules in a quarter of the space, while reducing power consumption.

The nC52 product line features highly integrated components for building PC-compatible engines for portable, low-power, and small form-factor applications. Passthrough connectors enable modules to be interconnected by stacking them onto an array of pins in four basic orientations. The nC52 interface chip senses the orientation and automatically configures itself to properly swap signals in order to present a consistent interface to the remaining circuitry on the module. This technique keeps signal lines short, so the bus can operate at high speeds without termination concerns.

Available products include '586 and '486 processor modules, a VGA interface module, an I/O module supporting serial and parallel ports, two IDE devices, and two floppy drives. Also available is a 8051 system-management module, a Sound Blaster-compatible audio subsystem module, and a dual-port PC Card interface module. Prices start as low as **$80** per module.



Adaptive Systems, Inc.
P.O. Box 1150
Niwot, CO 80544
(303) 247-1317
Fax: (303) 247-1318
www.adaptivesystems.com

#512

FLAT-PANEL LCD DRIVER BOARD

Arcom has introduced the **AIM-104-LCD-3**, a PC/104-format module that runs either VGA CRT monitors or VGA LCD flat-panel displays. The board drives a wide range of commercial flat-panel display types like analog CRT monochrome single- or dual-scan STN, dual-scan color STN, and color TEF.

The AIM-104-LCD-3 interfaces through a standard 50-way box header ribbon-cable connector. The board is supplied with 512 KB of video memory that provides the user with video formats up to 1024 × 768 in 16 colors and 640 × 480 in 256 colors.

A supplied utility disk includes standard drivers supplied by Cirrus, including CLMODE and a TSR (terminate and stay resident) application that allows the user to swap between CRT, flat panel, and simultaneous mode.

The AIM-104-LCD-3 sells for **$220**.

Arcom Control Systems, Inc.
13510 S. Oak St. • Kansas City, MO 64145
(816) 941-7025 • Fax: (816) 941-7807
www.arcom.co.uk **#513**

NouveauPC

NPC

PENTIUM II SBC WITH VIDEO OR CREATIVE SOUND

Interlogic has introduced a Pentium II SBC that comes in two versions: **Creative Sound Blaster 16 and Roland MPU 401 UART mode-compatible audio** or **ATI 264VT 64 bit PCI Graphic Accelerator** with 2 MB of display DRAM. Both versions include full support for the Pentium II processor at speeds up to 333 MHz. They also feature 512 KB of onboard L2 cache, memory up to 256 MB of fast page mode EDO or BEDO DRAM, and the 440-series chipsets with dual bus-master PCI EIDE controller interface for fast data transfer.

The Award flash BIOS with plug-and-play is auto-configurable and provides a full selection menu for custom setups, including master/slave interface, watch-dog timer, page mode, and mode-4 timing. Also included is support for USB and IrDA, two high-speed 16C550-compatible serial ports, and one bidirectional parallel port with EPP/ECP functions. As well, these SBCs feature a PS/2 mouse port; BIOS support for LS-120, ZIP, and CD-ROM drives; and a socket for a Disk-On-Chip flash disk.

The boards are available for **$495** quantity one, no video, and for **$595** with video.

Interlogic Industries
85 Marcus Dr. • Melville, NY 11747
(516) 420-8111 • Fax: (516) 420-8007
www.infoview.com **#514**

NouveauPC

Greg Bergsma

Building Web-Enabled Mobile Devices

Want to design a Web appliance into a hand-held unit? If so, Greg shows you how to easily change the form and content of your dynamic HTML without recoding and retesting the Web server. He predicts it'll be a boon for the vending industry.

All of us have used the Web on our desktops. It's become a standard part of the way we work.

But what about using the Web in embedded devices? It can fit anywhere you need to serve up or display data using classic client/server architecture. The key is being able to readily customize the technology for your embedded application.

Many embedded devices use Web technology today—phones, PLCs, PDAs, TVs, photocopiers. At the same time, mobile computing is on the rise. Hand-held devices are finding their way into healthcare, telecommunications, manufacturing, security, banking—you name it.

These devices display information, so they're prime candidates for browsing technologies, be it to download information from a remote database, send E-mail to the head office, or troubleshoot a black box. From the embedded developer's perspective, the opportunities are endless.

But, there's a problem. On the desktop, a Web browser is something that lets you, well, browse. One browser works much like the next. But if you build a browser into a hand-held medical device, that browser needs to look and behave quite differently.

Similar issues apply to any embedded Web server a browser-enabled HHC might talk to. It has to be small and configurable, and it needs to let you easily change the form and content of your dynamic HTML without having to recode and retest the Web server itself.

What kind of tools do you need to build this kind of Web technology? To get an idea, follow along as I build the Web into a vending application.

DYNAMIC VENDING, INC.

Here's the scenario: I've been asked to design a new delivery solution for a highly successful—and entirely fictitious—vending machine company, Dynamic Vending Inc. (DVI). Management at DVI has decided that the next generation of vending machines and delivery systems will be highly functional and Web based.

Each vending machine will support remote queries, whether from head office or from any delivery truck. As a result, delivery staff can get detailed, up-to-the-minute information on how many candy bars—and how much cash—is in each machine.

Each machine will automatically report problems to DVI's head office. These messages could include "stock running low," "out of change," and "tilt." (You never know when Homer Simpson will attempt brute force to elicit candy from the machine.)

Each delivery person will carry a hand-held computer (HHC) that can:

- instantly retrieve stock details from any vending machine
- gather restocking information, then transfer that information to DVI's central database and invoicing system
- contact the head office directly
- print service details for the customer

Since DVI has some pretty big customers, the system must also provide remote

access to sites with hundreds of vending machines. And the system shouldn't force DVI to dial up each vending machine separately.

One more thing—I have three months to get a jump on DVI's archrival, Vending Dynamics Inc. (there's still an outstanding lawsuit over their swiping the company name).

OFF-THE-SHELF TECHNOLOGY

Three months! Are these guys nuts? The only way to do this is if everything is off-the-shelf. Well, at least, most of the software is—particularly the Web technologies.

To start, I need an embedded Web server so DVI can remotely query each vending machine. I need to equip each one with an embedded Web server, along with a mechanism for generating dynamic HTML.

This mechanism will retrieve the current status of the machine (stock levels, cash levels, etc.) and format that information in HTML. The Web server can then serve that HTML to anyone querying the machine from a remote browser. Of course, the machine needs a modem or network interface.

The delivery staff needs to query the vending machines from their HHCs. So each HHC needs a Web browser.

Obviously, a conventional desktop browser won't do. I need something small and easily controlled from a pen-sensitive display. I'll probably need to customize the browser interface—the delivery staff doesn't have time to mess with lots of menus and buttons.

I'll use an embedded E-mail client both in the vending machine and in the HHC. The E-mail client in the vending machine will automatically transmit any problems to DVI's head office. The E-mail client in the HHC will, among other things, keep the head office updated on the delivery person's schedule.

In addition to these technologies, each vending machine should:

- be able to take advantage of the customer's Ethernet backbone (to transmit data) if possible, otherwise be accessible via modem
- optionally connect to the customer's E-mail server (to report problems)

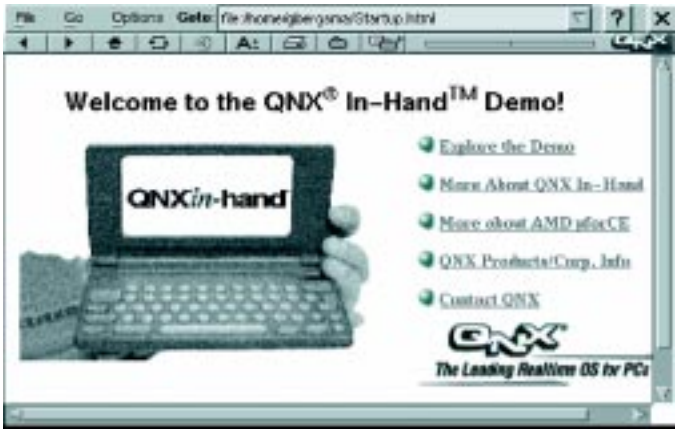


Photo 1—Before customization, the QNX In-Hand demo browser shows on-line content from a flash-resident demonstration unit.

- permit the head office to gather statistics on individual machines and on groups of machines

The HHC has some requirements, too. For one thing, it needs some cellular-phone technology to connect to vending-machine modems. It should have a local database with ODBC functionality so its database can synchronize with head-office systems. And, it has to support standard PDA components as well as a bar-code reader and receipt printer.

When I put everything together, it looks like Figure 1.

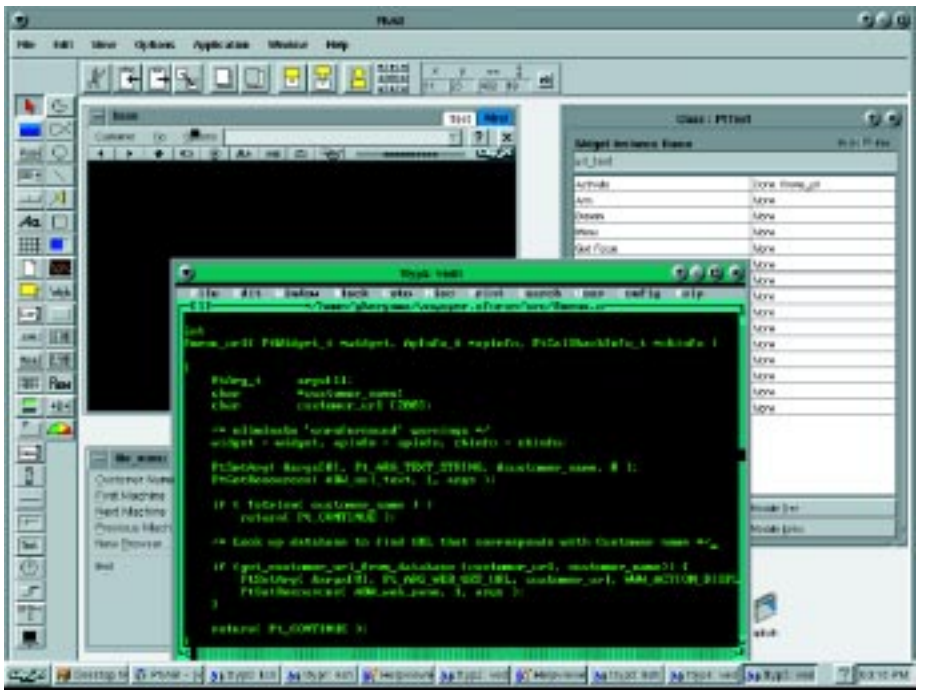


Photo 2—Customization of the standard Voyager browser is done through the Photon Application Builder (PhAB). Here the labels in the menu have been changed so that the user can enter a customer name and other appropriate options. The program listing in the forefront shows the coding changes made to the function `fmenu_url()`***. With this, it is possible to retrieve the customer's URL by searching via the customer's name in the database.***

BUILDING THE HHC

Given the time frames (who came up with this schedule, anyway?), the biggest challenge is to adopt a hardware and software design for the HHC.

Perhaps the most difficult task is getting browser software that can be readily customized. Fortunately, the QNX In-Hand Toolkit offers a suite of off-the-shelf software and hardware that lets me customize the HHC.

The kit provides a suite of software on CD and the µforCE, a new hand-held reference design from AMD. Plug in the power, the system boots from onboard flash memory—and presto! There's the display shown in Photo 1.

The whole development environment is on the CD, which I installed on a partition on the desktop PC. The CD includes the QNX RTOS, the Photon microGUI, and all the tools and demo applications that were used to build the demos that ship in the flash memory.

The Voyager browser runs under Photon and is customizable through PhAB

(Photon Application Builder), an integrated GUI development tool that lets you add buttons, labels, windows and other objects known as widgets.

Widgets can have code attached to them, which is executed as a result of some sort of event (e.g., a mouse click). This callback code is a complete C function.

All graphical demo applications that come in the flash memory are customizable through PhAB. Let me give you an idea of how this is done by customizing the browser for this DVI HHC.

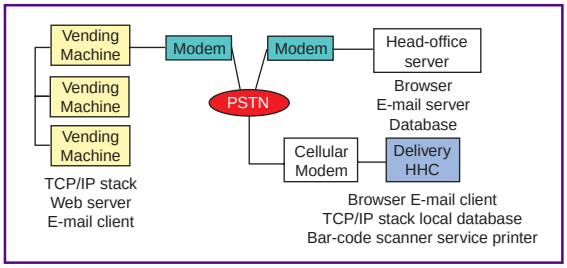


Figure 1—This system overview shows the various components needed by vending machines, HHCs, and head-office servers. Access to vending-machine data can be via modem (including cellular) and optionally via the customer's Ethernet backbone.

THE HHC BROWSER

The browser is actually two programs. The interface has been separated from the underlying browser engine to create a client/server architecture.

The interface (client) talks to the engine (server) through a GUI widget called `PtWebClient`. In turn, the engine informs the interface of progress, status, and other information. The interprocess communication between the two programs is handled by a callback mechanism in `PtWebClient`.

With an architecture like this, customizing the browser becomes straightforward. You can freely modify the browser interface without having to understand the complexities of the browser engine.

The default browser expects the user to enter a URL. But DVI users are too busy to enter a URL for each customer and each machine. Instead, I'll let them simply enter the customer name. The HHC database can look up the appropriate URL.

When you bring up the browser's "File" menu under PhAB and choose the "Goto URL" item, you see a dialog that lets you enter a URL. When the user hits Enter, a C callback function `fmenu_url()` executes.

`fmenu_url()` sets a widget resource `Pt_ARG_WEB_GET_URL` in the `PtWebClient` widget through another function, `PtSetResources()`, which tells the browser engine to retrieve the Web page.

Changing the File menu so users can enter a customer name consists of two steps. To make an intuitive interface, I change the title of the "File" menu to "Customer" and rename "Goto URL" to "Customer Name."

Next, I want to retrieve the appropriate URL when the user enters a customer name. To do this, I make a simple modification to the callback function `fmenu_url()`. The user types their customer name into the `customer_text` widget.

To retrieve the text entered by the user, send a call to `PtGetResources()`. Next, look up the URL in the database that corresponds with the customer's name and then set the `Pt_ARG_WEB_GET_URL` resource, which causes the browser engine to retrieve the Web page. The modified callback code appears in the forefront of Photo 2.

These changes are made through PhAB. Once the code is modified, it regenerates the widgets (so resource updates take effect) and recompiles all relevant components of the application. You see the PhAB environment in Photo 2.

Once the code recompiles, the customized browser looks like Photo 3. As you see, entering the customer name "QNX" retrieves a page from a vending machine's Web server.

Once you get used to this environment, you can develop custom browsers in a matter of hours. The modifications took me about an hour, including the database code and a caffeine break. To modify any other QNX In-Hand applications (e.g., the E-mail client and Internet dialer), you can use the same environment.

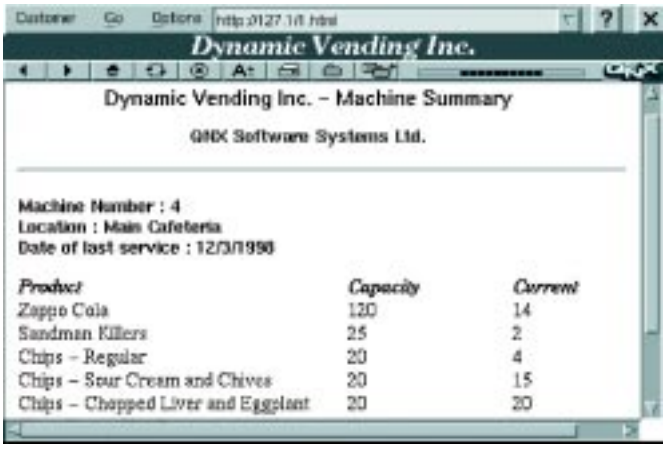


Photo 3—After customization, the browser shows typical Web content supplied by the Web server in the vending machine.

µforCE HHC HARDWARE

The µforCE board, pictured on the *EPC* cover (p. 37), includes all the hardware components you need—a pen-sensitive LCD screen, a PCMCIA slot, and the Élan-SC400, which was designed specifically for power-sensitive devices. Its audio support also offers audible notification for delivery staff.

Just plug a bar-code scanner into the serial interface (to scan candy bars), add a parallel interface (to support a receipt printer), and leave out anything you don't need. Next, submit your design to a custom PCB/board manufacturer and they'll make a swag of them to spec (see Figure 2).

BUILDING A VENDING MACHINE

So what technology is needed for the vending machine? It has to support several interfaces. It needs an LCD or an LED display. It should accept and return coins and let users select the products they want. And, of course, the products have to get to the user.

The circuitry for all these devices can be readily driven from a PC-based hardware design—something in a PC/104 form factor, for example.

The machine needs to track stock quantities for each product, the coin levels in collection and change canisters, and any machine-specific status values.

As the system overview in Figure 1 shows, each machine also needs an embedded Web server to provide status values, an E-mail client to report problems, mechanisms for generating dynamic HTML, and Ethernet and/or modem connections.

QNX features a low-overhead sendmail client, a Web server called Slinger (only 14 KB of code), and a dynamic HTML generator. The same API and development tools work for both the HHC and the vending machines, which saves precious development time.

IMPLEMENTING DYNAMIC HTML

So how exactly can you implement dynamic HTML in something like a vending

machine? The classic approach is to add a function directly into your embedded Web server. When a dynamic HTML page is being accessed, it queries the hardware and generates new HTML.

The Web-server technology in the QNX In-Hand toolkit implements a unique and flexible approach that doesn't require any modification of the Web server. You can implement the dynamic HTML pages as filenames in the filesystem's pathname space (see Figure 3).

Rather than being stored on disk or in flash memory, the named files exist as connections to an HTML-generator process that runs separately from the Web server. Whenever the named files are read, this generator probes the hardware of the vending machine and generates HTML that reflects the current state of the system.

Let's look at the dynamic HTML generator and see how it works (see Listing 1).

First, the generator lets the OS kernel know what part of the pathname space it is responsible for. In this case, I'm dealing with the `qnx_prefix_attach()` function call, specifying the file `/usr/httpd/1.html`. This action causes all I/O requests (`open()`, `close()`, `read()`, `write()`, etc.) for `/usr/httpd/1.html` to be directed to the HTML generator.

These requests are delivered as standard messages with a standard format, through which the generator replies. The main requests are `open()`, `read()`, and `close()`. My system doesn't support writing to the HTML file.

When the Web server opens the file, a message with a type of `IO_OPEN` is delivered to the generator. The generator checks the file permissions against the permissions of the calling process and, if OK, allocates an OCB (open control block) that becomes the "handle" with which subsequent I/O requests are based. It allocates a buffer for the data, and then calls

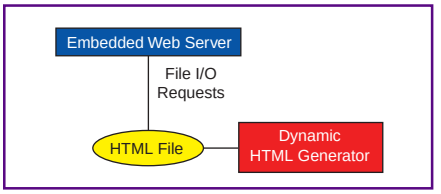


Figure 3—Under QNX, dynamic HTML is implemented through adopting a portion of the "pathname" space and handling the subsequent file I/O requests without having a physical HTML file on permanent media.

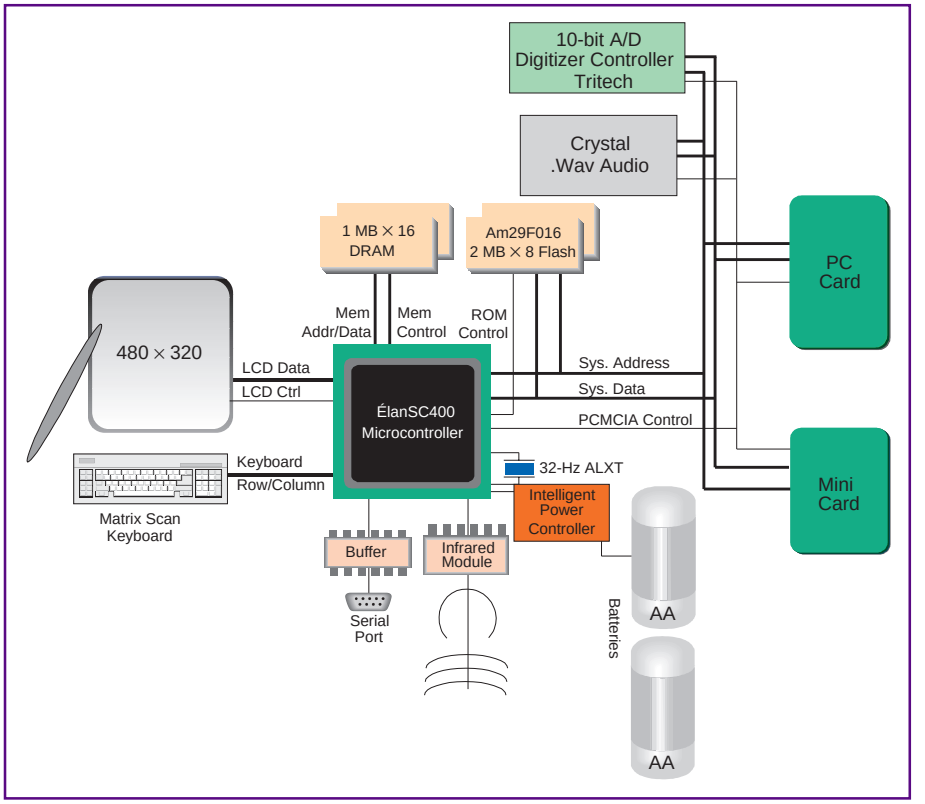


Figure 2—The µforCE demonstration includes the AMD ÉlanSC400 microcontroller and all the associated chipsets that you would find in most HHC designs.

`fill_buf()` to put the HTML content into that buffer (see Listing 2).

Messages of type `IO_READ` are received when the Web server is retrieving the data. Simply check to see if the data is exhausted. If it isn't, we reply with the amount of data requested or whatever data is remaining (whichever is smaller).

Data is returned via `Replymx()`. Each time you retrieve data, keep track of where you are up to in the buffer by updating the offset member of the OCB structure.

`IO_CLOSE` tells you the file is closing. So, you should deallocate any objects allocated as a result of I/O calls on the file.

`IO_WRITE` is not supported, so `ENOTSUP` is returned. It ends up being passed back through in `errno` to the program trying to write to the file.

There are a few other functions that need to be supported, but this is the guts of it.

HOW IT ALL FITS

So far, I've shown you some of the components required for this project as well as how I might implement some of them. Now, let's see how the whole scheme works in practice.

It's three months later, and DVI has equipped the delivery staff with my spiffy new HHC. It has also deployed the Web-enabled vending machines at several major locations.

But, exactly how do delivery staff use the new system? Let's look at a typical workday for Dave, one of the senior members on the delivery team.

Dave's first delivery run is in a high-tech business park, where he delivers cartloads of super-caffeinated cola to hundreds of overworked engineers. Before he starts out, he queries the machines at each site (using his HHC or a remote desktop PC) and gets detailed information on how much product is required for each machine, as well as an overall summary across all machines. Based on this information, Dave determines how many locations he can cover in a single trip and how to stock his truck.

At each location, Dave again interrogates each machine to check the latest status. He then restocks the machine, removes the money, and uses the HHC's custom browser to gather the status details (provided by the vending machine's embedded Web server and then saved in the HHC's database). Before he leaves, Dave uses the HHC to print out a service receipt for the customer.

Listing 1— The dynamic HTML generator has four main types of I/O requests to handle— *open(), read(), write(),* ***and*** *close()****. These requests are handled through a well-defined message structure already in place for regular file-system I/O.***

```
void
main(int argc, char *argv[]){
  struct _mxfer_entry mx_entry[2];
  int          devno;
  off_t        endmem;
  pid_t        pid;
  msg_t        type;
  int          nbytes;
  mode_t       mode;
  struct ocb   *ocb;
  int          size;
  if (qnx_prefix_attach("/usr/httpd/1.html", NULL, 0)){
    perror("prefix_attach");
    exit(EXIT_FAILURE);}
  for (;;) {
    pid = Receive(0, &msg, sizeof(msg));
    type = msg.type;
    msg.status = ENOSYS;
    nbytes = sizeof(msg.status);
    switch (type){

      case _IO_OPEN:
      // check access permissions are OK
      mode = perm_mode[msg.open.oflag & O_ACCMODE];
      __get_pid_info(pid, &psdata3, fd_ctrl);
      mode &= (mode & 077) |  /* reuse lower 6 bits for handle */
        (psdata3.euid == 0) ? S_IRWXU :
        (psdata3.euid == fstat_reply.stat.st_uid) ?
        (fstat_reply.stat.st_mode & S_IRWXU) :
        (psdata3.egid == fstat_reply.stat.st_gid) ?
        (fstat_reply.stat.st_mode & S_IRWXG) << 3 :
        (fstat_reply.stat.st_mode & S_IRWXO) << 6;
      if ((mode & S_IRWXU) == 0){
        msg.status = EPERM;
        break;}
      // permission check OK—allocate control block
      if (msg.open.path[0] != '\0' {
        msg.status = ENOENT;
        break;}
      if (!(ocb = (struct ocb *) malloc(sizeof(struct ocb)))){
        msg.status = ENOMEM;
        break;}
      memset(ocb, 0, sizeof(struct ocb));
      ocb->mode = mode;
      ocb->count++;
      // attach a handle (fd) to the ocb for use later on
      if (qnx_fd_attach(pid, msg.open.fd, 0, 0, 0, 0, (unsigned)
           ocb) == -1){
        free(ocb);
        msg.status = errno;
        break;}
      if ((ocb->buf = malloc(BUFLEN)) == NULL){
        free(ocb);
        msg.status = errno;
        break;}
      ocb->offset = 0;
      fill_buf(ocb->buf);     // fill data buffer with HTML data
      msg.status = EOK;
      break;

    case _IO_CLOSE:
      // release all allocated objects - ocb/buffer/fd
      ocb = (struct ocb *) __get_fd(pid, msg.close.fd, fd_ctrl);
      if (ocb == (struct ocb *) 0 || ocb == (struct ocb *) - 1) {
        msg.status = EBADF;
```

(*continued*)

Listing 1— continued

```
        break; }
      if (--ocb->count == 0){
        free(ocb->buf);
        free(ocb);}
      qnx_fd_attach(pid, msg.close.fd, 0, 0, 0, 0, 0);
      msg.status = EOK;
      break;

    case _IO_READ:
      ocb = (struct ocb *) __get_fd(pid, msg.read.fd, fd_ctrl);
      if (ocb == (struct ocb *) 0 || ocb == (struct ocb *) - 1){
        msg.status = EBADF;
        break;}
      if ((ocb->mode & S_IREAD) == 0){
        msg.status = EBADF;
        break;}
      if (ocb->offset == -1){    // Already at end of file
        msg.read_reply.status = EOK;
        msg.read_reply.nbytes = 0;
        msg.read_reply.zero = 0;
        _setmx(&mx_entry[0], &msg, sizeof(struct _io_read_reply) -
               sizeof(msg.read_reply.data));
        Replymx(pid, 1, &mx_entry);
        nbytes = 0;
        break; }
      // reply with the minimum of either the size of data request
      //   or size of data remaining in buffer
      size = min(msg.read.nbytes, strlen(ocb->buf) - ocb->offset);
      _setmx(&mx_entry[1], ocb->buf + ocb->offset, size);
      msg.read_reply.status = EOK;
      msg.read_reply.nbytes = size;
      msg.read_reply.zero = 0;
      _setmx(&mx_entry[0], &msg, sizeof(struct _io_read_reply) -
             sizeof(msg.read_reply.data));
      Replymx(pid, 2, &mx_entry);
      nbytes = 0;
      // update data offset and access time
      ocb->offset += size;
      if (ocb->offset >= strlen(ocb->buf)) ocb->offset = -1;
      fstat_reply.stat.st_atime = *timep;
      break;

    case _IO_WRITE:
      msg.status = ENOTSUP;
      break;}
    if (nbytes){
      Reply (pid, &msg, nbytes);}}}
```

When the vending machine has problems, they automatically dial the DVI head office (or an ISP, for that matter) and send an appropriate E-mail message. Now, you might be wondering, what if there are dozens of machines at one site? You need modems and phone lines for each one, right?

With inherent distributed networking, a network of machines looks like a single logical machine. So if I network the machines together, they can all share the same modem and even the same IP address.

Finally, Dave drives back to the warehouse and updates the central database. To do this, he simply hooks up his HHC to the network, which then transfers the data via ODBC.

A SWEET ENDING

As you can see, adopting off-the-shelf Web technologies can significantly reduce development time for hand-held devices. In fact, deciding you can use Web technology is probably the biggest hurdle to overcome.

Having the flexibility to customize both browser and server opens Web technologies to a myriad of embedded applications. So, keep an open mind and ask yourself whether you could use Web technologies in your next implementation. EPC

Listing 2—The dynamic HTML content is written into a data buffer through a series of *sprintf()* ***function calls. The data is retrieved from the buffer through*** *I/O_READ* ***messages.***

```
void
fill_buf(char *buf){
  char *p = buf;
  int i;

  // set up HTML header
  p += sprintf(p, "<HTML>\n<HEAD>\n<TITLE>Dynamic Vending
       Inc.</TITLE>\n");
  p += sprintf(p, "</HEAD>\n<BODY LINK=\"#800080\"
       VLINK=\"#c0c0c0\" ");
  p += sprintf(p, "BGCOLOR=\"#ffffff\">");
  p += sprintf(p, "<FONT FACE=\"Arial\"><H4
       ALIGN=\"CENTER\">Dynamic Vending Inc.");
  p += sprintf(p, " - Machine Summary</H4>\n");
  p += sprintf(p, "<H5 ALIGN=\"CENTER\">");
  p += sprintf(p, detail->customer_name);
  p += sprintf(p, "</H5><hr>\n");
  p += sprintf(p, "<h5>Machine Number : ");
  p += sprintf(p, "%d", detail->machine_number);
  p += sprintf(p, "<br>Location : ");
  p += sprintf(p, detail->location);
  p += sprintf(p, "\n<br>Date of last service : ");
  p += sprintf(p, "%d/%d/%d", detail->day, detail->month, detail-
       >year);
  p += sprintf(p, "</h5>\n");

  // set up table header
  p += sprintf(p, "<TABLE CELLSPACING=0 BORDER=0 CELLPADDING=0
       WIDTH=480>\n");
  p += sprintf(p, "<TR><TD WIDTH=\"50%%\" height=16>");
  p += sprintf(p, "<B><I>Product</B></I></FONT></TD>\n");
  p += sprintf(p, "<TD WIDTH=\"25%%\" height=16>");
  p += sprintf(p, "<B><I>Capacity</B></I></FONT></TD>\n");
  p += sprintf(p, "<TD WIDTH=\"25%%\" height=16>");
  p += sprintf(p, "<B><I>Current</B></I></FONT></TD>\n");
  p += sprintf(p, "</TR>");

  // set up each row
  for (i=0; i < detail->number_of_products; i++) {
    p += sprintf(p, "<TR><TD WIDTH=\"50%%\" VALIGN=\"TOP\"><P>");
    p += sprintf(p, "%s</FONT></TD>", detail->product
         [i].product_name);
    p += sprintf(p, "<TD WIDTH=\"25%%\"><P>");
    p += sprintf(p, "%d</FONT></TD>", detail->product
          [i].capacity);
    p += sprintf(p, "<TD WIDTH=\"25%%\"><P>");
    p += sprintf(p, "%d</FONT></TD>", detail->product
      [i].current_level);
    p += sprintf(p, "</TR>");}

  // page trailer
  p += sprintf(p, "</table><HR>");
  p += sprintf(p, "<H5 ALIGN=\"CENTER\">&copy; Dynamic Vending Inc. ");
  p += sprintf(p, "1997</H5>\n</BODY></HTML>\n");}
```

Greg Bergsma serves as senior technology analyst at QNX Software Systems. He has also worked on real-time transaction processing and embedded gaming terminals, as well as developing a variety of demanding real-time applications from plastics injection molding systems to children's television game shows. You may reach him at gbergsma@qnx.com.

SOURCE
QNX In-Hand
QNX Software Systems
175 Terence Matthews Cres.
Kanata, ON
Canada K2M 1W8
(613) 591-0931
Fax: (613) 591-3579
www.qnx.com/inhand

IRS

410 Very Useful
411 Moderately Useful
412 Not Useful

EPC

Pascal Dornier

'386 Processor Survey

Part 1: '386-Class Embedded CPUs

How do you know which CPU is best suited to your embedded-PC application? Take some cues from Pascal as he examines the strengths and weaknesses of '386s from Acer Labs, AMD, and Intel.

Once upon a time—or at least about ten years ago—PC motherboards consisted of a pile of TTLs and separate packages for the interrupt, timer, and DMA controllers, not to mention a memory controller based on (shudder) delay lines. This construction always mystified me. Even most home computers at that time (e.g., the Commodore 64) were already quite highly integrated.

Then along came PC chipsets, which simplified motherboard design and enabled the high-volume, low-cost PC-clone market. With some exceptions (e.g., Cyrix MediaGX), further integration of chipset functionality into the CPU hasn't happened in the volume-PC market due to the steadily increasing resource requirements of "productivity" software.

On the other hand, the embedded-PC market has readily accepted integrated CPUs. Many embedded products are manufactured in relatively low volume, and the reduced design time and component count greatly lower the overall cost. Embedded CPUs are usually based on mature technology, which is a nice way for chip manufacturers to keep their old fabs busy.

But of course, each CPU has particular strengths and weaknesses. How do you decide which one best suits your needs?

In the first of this two-part series, I review '386-class CPUs from several manufacturers. Next month, I'll cover the '486 class of CPUs. As always, there are trade-offs to be made, so with the information I present, I hope you'll be better able to select the right part for your application.

ALI M6117C

To get started, let's take a look at the ALI M6117, a device based on the Acer Laboratories '386SX CPU core. As you see in Figure 1, it includes a DRAM controller, real-time clock, and keyboard controller. Compared to its competitors, this part is quite simple and should be the easiest of the group to design with.

At 40 MHz, the M6117 is the fastest CPU in the group. The DRAM controller supports fast-page mode and EDO DRAM. True, it's a little dated (e.g., banks can't be assigned arbitrarily), but the M6117 is still more flexible than other CPU options on the market.

The M6117 has a power-management unit, but unless you add external circuitry to slow down or stop the CPU clock, the power savings don't turn out to be all that significant.

For most designs, it should be sufficient to execute the `HLT` instruction when idle. The power-management unit can generate *SMI, NMI, or a regular interrupt.

The M6117 has a keyboard/mouse controller, an IDE interface, and a programmable watchdog timer on chip. However, it doesn't feature serial or parallel ports on chip, so you need to add an external super I/O controller.

Unfortunately, the watchdog timer misses the mark. According to the documentation, it must be stopped during the update sequence. Clearing the watchdog requires a tedious sequence of configura-

tion register accesses. The watchdog can be programmed to generate a system reset, *SMI, NMI, or regular interrupt when it times out. I hope ALI plans to improve this feature.

Memory refresh is a performance leak in this design. ISA-refresh timing is used for DRAM refresh as well, which wastes more bandwidth than necessary. Modern ISA peripherals don't require refresh. The M6117 uses the refresh to read an optional 16-bit input buffer on the SD bus. This data can then be read through configuration registers.

Another problem: accesses to internal registers are not passed to the outside ISA bus. When you're trying to debug your system with a logic analyzer, this feature can be a real nuisance.

Despite any of these hassles, however, the pluses of the M6117 definitely outweigh the minuses. The M6117 is found on PC/104 single-board computers, and it forms the core of the ZF Microsystems SMX/386 PC module.

DESIGNING WITH M6117C

My sample design is for a PC/104 module with a CPU, 2–4 MB of DRAM, and 128–512 KB of flash memory. This design isn't final, so please contact me if you're interested in any updates.

The 80-MHz CPU clock and 14.31818-MHz OSC clock are synthesized by clock generator U7, using a 14.31818-MHz crystal as a time base. Power-on reset is generated by U1, a DS1811 packaged in a SOT-23. The built-in real-time clock requires a 32,768-Hz crystal.

Series resistors are placed close to the driver for all clock, DRAM, and ISA-bus strobe signals. Depending on layout, though, you may be able to eliminate series resistors for the DRAM signals.

The ALI M6117 also includes a keyboard controller. The keyboard signals are connected to a four-pin header. Ferrite beads minimize EMI on the signal and power lines, and keyboard power is fused to comply with safety standards.

The M6117 is a 5-V device. Therefore, it doesn't require any additional voltage regulators.

One or two 1M × 16 EDO DRAM chips can be used for a total of 2 or 4 MB of DRAM. The 512-KB EPROM, U3, is organized as a paged flash disk.

On reset, M6117 strobes the power-on value (pullup) of the XD bus into the flash page register, U2, which enables access to the top 64 KB of the device. Once the BIOS comes up, it can copy itself to shadow memory and reprogram the page register to access the remainder of the flash memory for disk emulation.

The IDE interface in this design isn't buffered, and it should be used with a very short cable only. The I/O decoding is done by the M6117.

When system power is off, RTC power (VBAT) is supplied by lithium battery BT1 through series resistor R22 (current limit required by safety standards) and diode D2. When power is on, VBAT is powered from +5 V through diodes D3 and D4.

AMD ÉLAN SC300

Another '386-class processor—AMD's Élan SC300—was originally designed for hand-held computers. This one includes just about everything but the kitchen sink.

As Figure 2 illustrates, the SC300 features a '386SX-33 CPU core, a CGA-resolution grayscale LCD controller, a PCMCIA controller, one serial and one parallel port, and a real-time clock. Everything runs off a single 32,768-Hz crystal.

Given the Élan SC300's hand-held-computer legacy, its power-management unit is very complex. This processor has multiple PLL clock generators for the CPU clock—both high and low speed. Depending on system activity, the CPU can run at full (33 MHz) or low (4.6/2.3/1.15/0.57 MHz) speeds.

In doze mode, the CPU stops completely and wakes up after

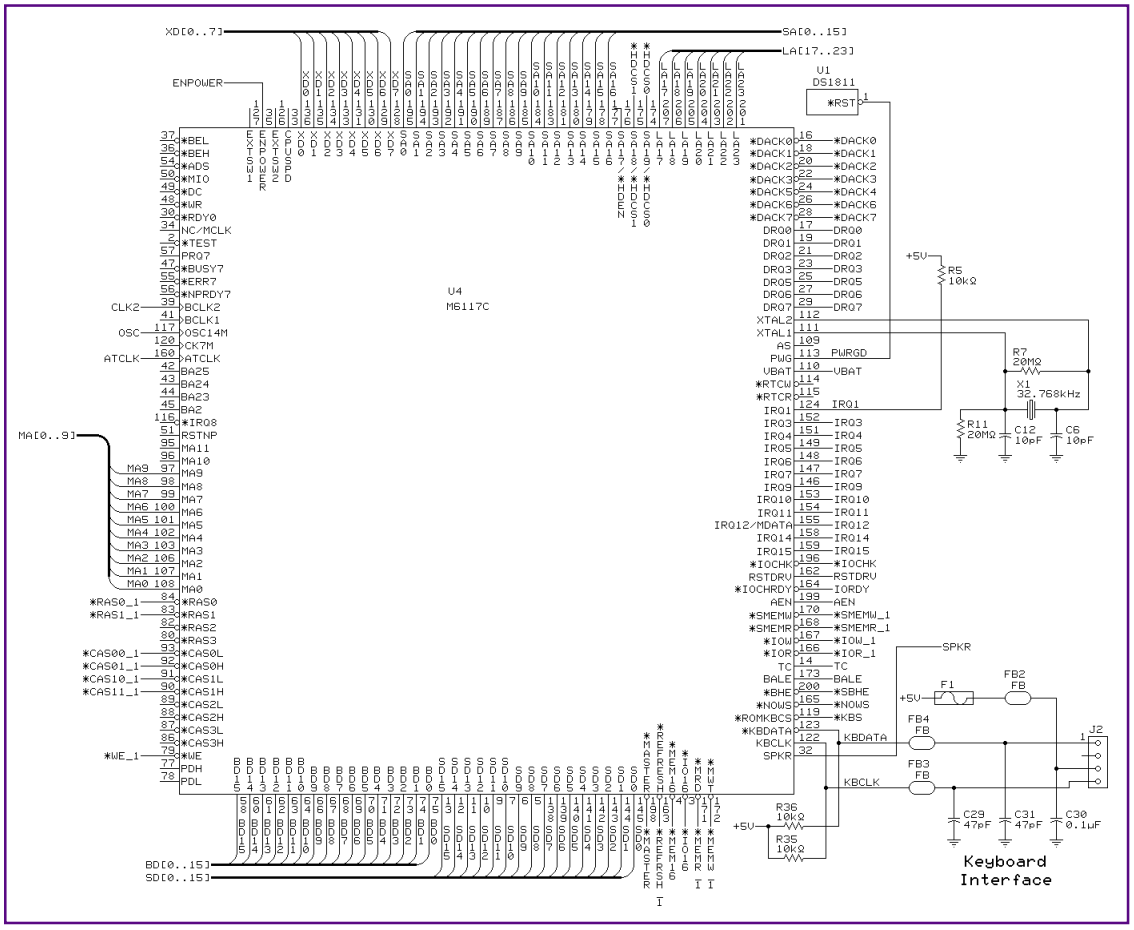


Figure 1a—ALI's M6117 makes this PC/104-based single-board computer tick.

each timer interrupt. Unfortunately, the high-speed PLL is always turned off in doze mode and can take 120–250 ms to recover. The AMD '386SX core doesn't support automatic power-down on HLT.

Many pins in this processor are multiplexed. For example, if you're using the LCD controller, many ISA-bus pins are no longer accessible. The LCD controller requires an external SRAM as a frame buffer. The PCMCIA controller requires external data and address buffers and power switches for each card slot.

Configuration bits for unrelated functions are combined together in the same register. Fortunately, the programmer's reference manual provides many cross references.

AMD's Élan CPUs don't support bus masters of any kind, which I think is quite strange. AMD's Ethernet controllers work best as bus masters.

Also on the downside, the real-time clock connects to the same power

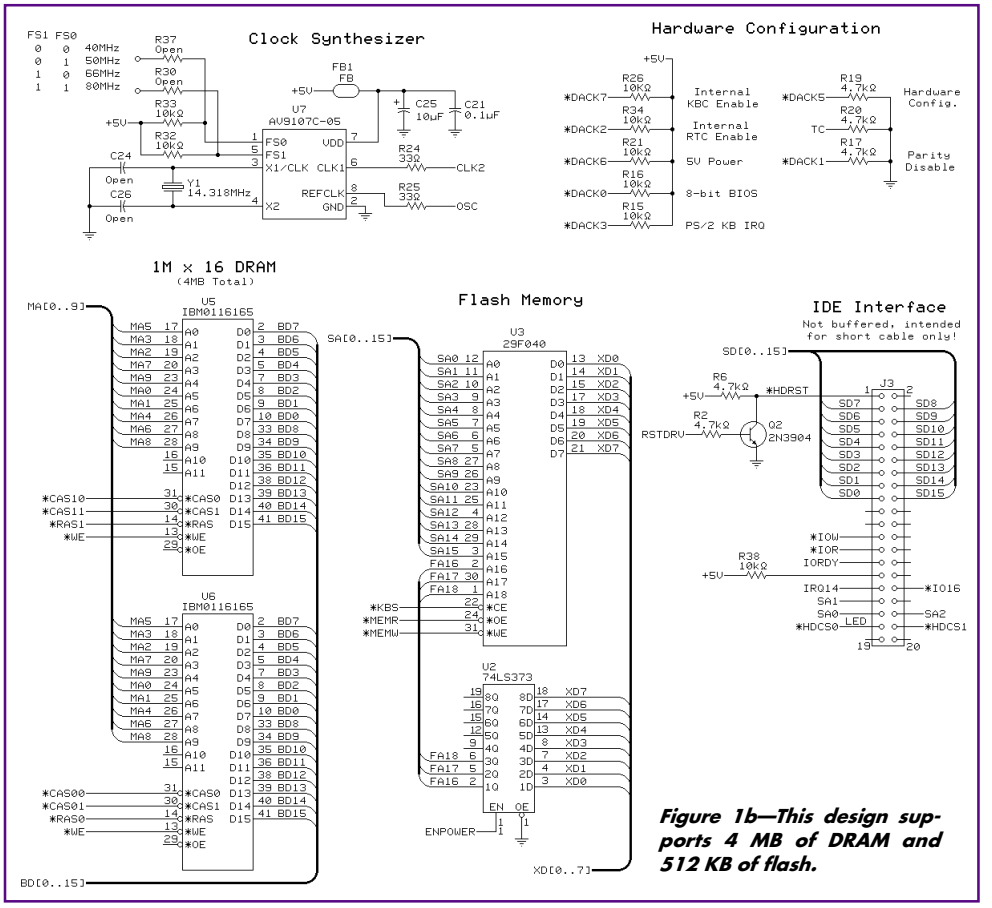
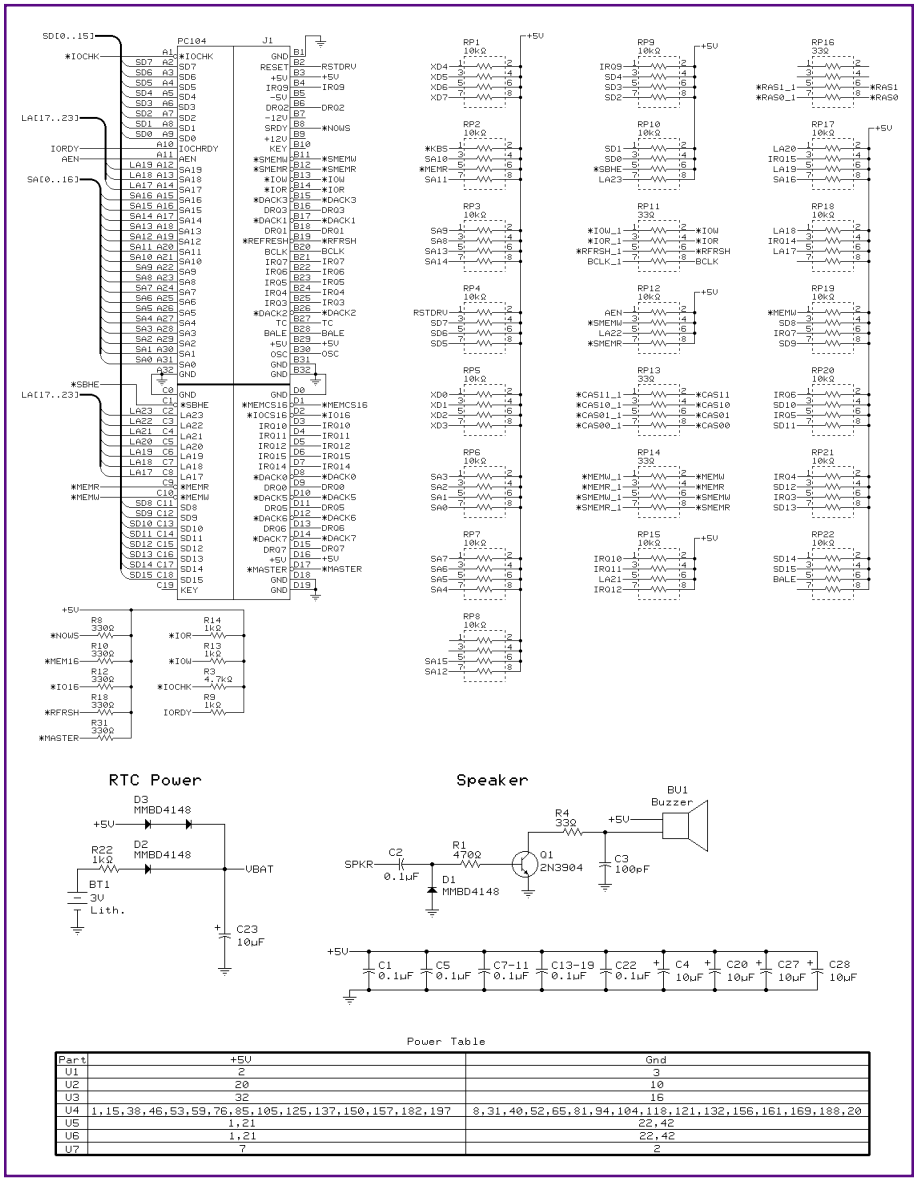


Figure 1b—This design supports 4 MB of DRAM and 512 KB of flash.



Part	+5V	Gnd
U1	2	3
U2	20	10
U3	32	16
U4	1,15,38,46,53,59,76,85,105,125,137,150,157,182,197	8,31,40,52,65,81,94,104,118,121,132,156,161,169,188,20
U5	1,21	22,42
U6	1,21	22,42
U7	7	2

Figure 1c—The M6117 connects directly into the PC/104 and IDE interfaces.

plane of the CPU core. This results in a relatively high leakage current of 25 µA (typical).

As well, a tricky switch-over circuit is required to provide battery backup. This doesn't meet my expectations for a reliable and economical design.

The internal RTC can't be disabled. An external RTC module must be decoded at a nonstandard address. The BIOS can then copy its contents to the on-chip RTC at powerup.

The Élan SC300 has been designed into many portable devices, such as Norand and Symbol hand-held terminals, the Brother Geobook low-cost notebook, and the Nokia 9000 cellular phone.

Another AMD processor—the Élan SC310—is a stripped-down version of the SC300. The LCD and PCMCIA controllers are disabled, but everything else is the same.

INTEL '386EX

The last '386-class processor I want to discuss here is Intel's '386EX. This CPU combines Intel's '386SX core with two asynchronous and one synchronous serial ports, a programmable chip-select unit, and some digital I/O. There is no RTC or keyboard interface. The timer and interrupt controllers are mostly PC compatible.

The DMA controller is a 8237 superset (26-bit address counter) but only supports two channels. The DMA signals are multiplexed with one of the serial ports. The DMA controller is not fully compatible with standard floppy controllers, so BIOS implementations use polling instead of DMA for floppy transfers.

The '386EX supports the full industrial temperature range. This range can even be exceeded if you can ensure sufficient cooling or a reduced duty cycle in your design.

Despite its simplicity, the power-management unit of the '386EX may be the most usable of the parts I've reviewed. Depending on how the part is set up, it enters either idle or power-down mode when a HLT instruction is executed.

The timer and UART can run off a separate clock input while the CPU core is completely stopped. The '386EX can't match the low power consumption of the Élan SC300/SC310 in standby mode, however. And if an external DRAM controller is needed (running on 2× clock), the situation is likely to be much worse.

External peripherals can interface to '386EX bus signals or to a simple *RD/*WR bus interface. Unfortunately, the timing on this bus interface is too tight. For example, according to the timing spec, there is no set-up time from chip-select valid to *WR low.

As well, in a worst-case situation of 33 MHz, the data-hold time on write cycles is 5 ns. This timing is likely to cause problems with many ISA peripherals.

Additionally, the bus turnaround time is too short. A data buffer is needed to prevent contention for slower devices like EPROMs.

The '386EX has a integrated watchdog timer. However, it may not work for your application because it stops when the CPU is in idle or power-down mode.

The '386EX doesn't include an integrated DRAM controller, so unless your RAM requirements are small enough to live in SRAM, you're going to have to add an external DRAM controller. This can be implemented in a CPLD or by using the RadiSys R300EX or R380EX controllers. But once you add a DRAM controller, the cost advantage of the '386EX is gone.

Additionally, don't forget to download the "specification update" (a.k.a. bug list) from Intel's Web site. There are quite a few issues you need to know about that can complicate designs using this part. Be sure to check the Circuit Cellar Web site for schematics of a minimal design using the Intel '386EX.

DESIGN SUGGESTION

I recommend eliminating the battery-backed real-time clock if at all possible. Batteries are guaranteed to fail over time.

Many designs can get the current time of day through a network, if they need it at all. A true embedded BIOS shouldn't have to rely on the CMOS RAM for configuration information.

If your application needs a reliable RTC, I recommend using an RTC module. They're much cheaper than the lifetime cost of battery replacements.

IMPROVEMENT OPPORTUNITIES

Most embedded CPUs still require too many external components. The ALI M6117 and AMD's Élan SC300 and SC310 require a lot of external pull-up resistors. Most of them could be integrated on chip at no cost (bus keepers or current mirrors).

Also, for many applications, the AMD Élan SC300 and SC310 and the Intel '386EX require external data/address buffers. In the case of the '386EX, it is due to insufficient bus turnaround time between write cycles and read accesses to relatively slow devices.

Additionally, the '386EX requires too much external glue logic to connect to just about anything other than fast SRAM and flash memory.

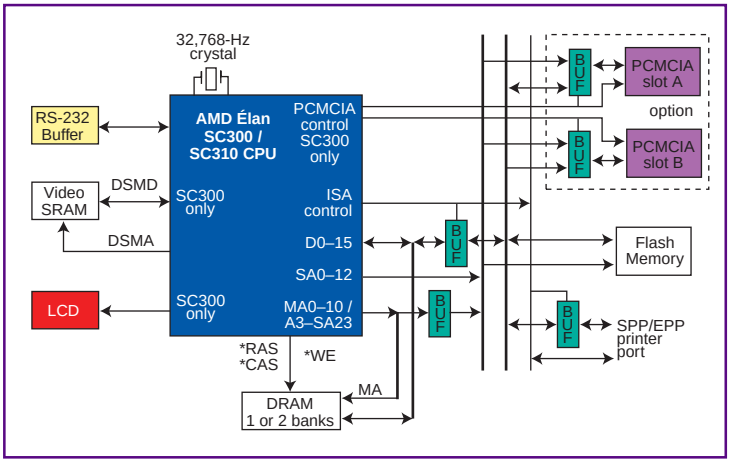


Figure 2—The AMD SC300 can implement a complete handheld computer in one chip.

Embedded applications frequently use a multinode RS-485 bus. After transmitting data, the transceiver must be turned off (normally controlled by RTS). The timing for this task is rather critical. You can provide a simple enhancement by adding a UART mode that enables and disables the transceiver automatically.

For embedded real-time designs, most power-management units miss the mark. SMI isn't desirable because it causes excessive interrupt latency and is hard to program. Ideally, executing the HLT instruction should be enough to get low power consumption.

Clocks should be stopped either immediately or after a programmable timeout, if a PLL recovery time is involved. Most activity timers are a waste of silicon. For example, most modern hard disk drives can take care of power management themselves.

As well, a watchdog timer shouldn't stop during updates, powerdown, and so forth. It should be accessible through a separate I/O port, not through indexed configuration registers (which require interrupt disable to protect the critical period between index and data register access).

And finally, I think most embedded CPUs need some improved debug support. In debug mode, all bus cycles to internal registers and DRAM should be visible on the DRAM data bus, and the DRAM address could be driven out to

	ALI M6117C	AMD SC300	AMD SC310	Intel '386EX
CPU core	ALI 386SX-40	AMD 386SX-33	AMD 386SX-33	Intel 386SX-33
Input clocks	32,768-Hz crystal (RTC) 80-MHz clock (CPU)	32,768-Hz crystal	32,768-Hz crystal	66-MHz clock
Coprocessor	External option	Not supported	Not supported	External option
Maximum power	1.45 W (5 V, 40 MHz)	0.86 W (3.3 V, 33 MHz)	0.86 W (3.3 V, 33 MHz)	0.46 W (3.3 V, 25 MHz); 1.6 W (5 V, 33 MHz)
Voltage range	4.75–5.25 V	3.0–3.6 V	3.0–3.6 V	3.0–3.6 V, 4.5–5.5 V
Idle power	250 mW (5 V) in halt state	73 mW doze mode; 0.12 mW suspend mode	73 mW doze mode; 0.12mW suspend mode	165 mW (3.3 V), 550 mW (5 V) halt; 0.5-mW power-down mode
Temperature range	0–70°C Tcase	0–70°C Tambient comm.; –40–85°C Tambient ind.	0–70°C Tambient comm.; –40–85°C Tcase ind.	–40–108°C Tcase
Price	$29 @ 100, $25 @ 1k (ALI)	$38.16 @ 100,1k (Hallmark) comm.	$30 @ 100,1k (Hallmark) comm.	$17.85 @ 100,1k (Hallmark)
Package	208QFP	208QFP	208QFP	132QFP/144TQFP
DRAM support	Up to 64 MB, fast page mode or EDO	Up to 16 MB, fast page mode	Up to 16 MB, fast page mode	Needs external DRAM controller (refresh controller is integrated)
Timer channels	3 (8254)	3 (8254)	3 (8254)	3 (8254)
Watchdog	Yes (limited)	No	No	Yes
Interrupts	11 (2 × 8259) + *SMI	11 (2 × 8259) + *SMI	11 (2 × 8259) + *SMI	10 (2 × 8259) + *SMI
DMA channels	7 (2 × 8237)	7 (2 × 8237)	7 (2 × 8237)	2 (8237 superset)
Keyboard interface	AT interface (8042)	XT interface	XT interface	No
RTC	Yes	Yes	Yes	No
Serial port	No	1 × 16450	1 × 16450	2 × 16450, 1 × sync.
Parallel port	No	1 × SPP / EPP	1 × SPP/EPP	DIO
Programmable chip selects	IDE interface, DIO (limited)	4 I/O, 1 memory	4 I/O, 1 memory	8, programmable wait
LCD controller	No	Yes, CGA resolution, needs external SRAM	No	No
PCMCIA	No	Yes, needs external buffers; not 365 compatible, but supports PCMCIA 2.0	No	No
Bus interface	ISA	ISA, '386SX optional	ISA, '386SX optional	'386SX, *RD/*WR
5-V tolerant I/O	No	Most	Most	No
Drive capability	8–24 mA depending on pin	Buffers recommended	Buffers recommended	4–8 mA
PC compatible	Yes	Yes	Yes	Limited
Testability	None	JTAG (only 1 MHz, very limited test features)	JTAG (only 1 MHz, very limited test features)	JTAG (10 MHz)

Table 1—Compare all '386-class embedded CPUs at a glance.

the ISA address bus for easier logic analysis. Disabling this for normal operation reduces both EMI and power dissipation.

CPU DECISIONS

Of course, your choice really depends on necessity. The requirements of your application dictate which CPU is best for you. Table 1 helps you compare these processors.

If you need to get your design done fast, I suggest going with the ALI M6117. Its simplicity won't get in your way. The full ISA-bus support also helps.

A basic design requires the right amount of glue logic devices—none. Sure, you'll have to add external peripherals, but they'll be your choice.

On the other hand, if power management is critical, the AMD Élan SC300/SC310 is the best choice in this group. Expect to spend some time with its jumble of configuration registers.

The SC300 shines in hand-held applications. By contrast, the SC310 is most suitable for deeply embedded applications that don't require a lot of peripherals on the ISA bus.

The Intel '386EX is strong in deeply embedded applications that have to support the full industrial temperature range (e.g., automotive applications). Other than that, I see it mainly in cost-sensitive applications that need very little RAM and can be implemented using SRAM.

Once you add a DRAM controller to the '386EX, however, it is likely to cost more money and board space than its competitors. Interfacing ISA peripherals such as network controllers also requires additional glue logic.

If you need a faster CPU, high-performance floating point, video display, or PCI bus, a '486 class CPU is going to be a better match for your application.

So, watch for all the information coming up in Part 2. In that article, I'll examine the AMD Élan SC400/SC410, Cyrix MediaGX, National NS486SXF/SXL, and SGS STPC Consumer. EPC

Thanks for reviewing this data goes to Klaus Flesch of Forth-Systeme (Élan SC300/SC310) and to Jim Stewart of JK Micro (Intel '386EX). All mistakes are mine.

Pascal Dornier is president of PC Engines, a design house for embedded PC hardware and firmware. You may reach him at pdornier@pcengines.com.

SOFTWARE
Schematics for the M6117 design, as well as for a minimal (untested) Intel '386EX design, are available via the Circuit Cellar Web site.

SOURCES
ALI M6117
Acer Laboratories, Inc.
1830B Bering Dr.
San Jose, CA 95112
(408) 467-7456
Fax: (408) 467-7474
www.acerlabs.com

Élan SC300/SC310
Advanced Micro Devices, Inc.
One AMD Pl.
Sunnyvale, CA 94088
(408) 732-2400
www.amd.com

'386EX
Intel Corp.
2200 Mission College Blvd
Santa Clara, CA 95052
(408) 765-8080
Fax: (408) 765-9904
www.intel.com

R300EX and R380EX controllers
RadiSys Corp.
5445 N.E. Dawson Creek Dr.
Hillsboro, OR 97124
(503) 615-1100
Fax: (503) 615-1150
www.radisys.com

IRS

413 Very Useful
414 Moderately Useful
415 Not Useful

Real-Time PC

Ingo Cyliax

Multimedia in Real Time

Multimedia is an ongoing rage. But as Ingo reveals, a lot of multimedia doesn't quite make the "real" part of real time. Read on to determine how to quantify your project needs since excessive throughput adds up to more expensive design costs.

Multimedia applications primarily concern encoding and decoding video, images, and audio. This month, I want to discuss some of the tools that are available to real-time multimedia system designers.

In particular, I want to look at the Pentium MMX architecture, which has been developed to address multimedia applications. Using reconfigurable hardware for multimedia apps seems like a good fit, and I discuss a possible solution.

MULTIMEDIA FOCUS

First question: what do I mean by multimedia? With all the hype, it's hard to get a clear definition.

For now, let's define multimedia as a combination of audio and video data to process in a system. The system may be a set-top box or video server. It doesn't matter.

Often, multimedia systems are real-time because of their deterministic response and throughput. When you generate video and audio, the system must send each frame of video at a certain time. It can't be early or late.

Even worse, the audio stream must synchronize with the video. An observer may not notice a dropped or duplicated video frame, but missed audio samples are more drastic, showing up as nasty clicks.

So, how real time is multimedia? In a system generating standard NTSC video and CD-quality stereo audio, the video requires a field every 1/60 of a second (~16 ms) and a whole frame (i.e., two fields, because NTSC is interlaced) in 1/30 of a second (~33 ms).

Of course, video is typically generated in hardware, but there's usually an interrupt for each field or frame so the software can update the frame buffer. However, a NTSC video line is only about 63 µs long, so a 50-µs interrupt latency can clobber most of a video line!

CD-quality audio is 16 bits at 44.1k samples per second, so

Photo 1—Traftech's Zephyr decodes an MPEG stream into NTSC or PAL analog video as well as audio. It also sports a PC/104+ interface, which brings PCI speeds to the PC/104 form factor.

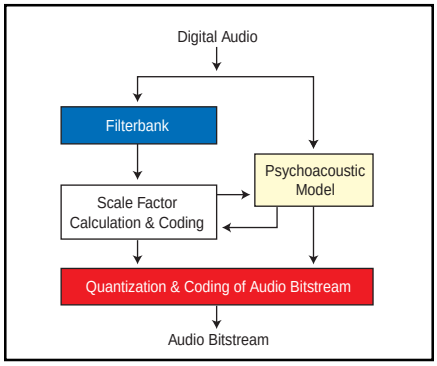


Figure 1—The psychoacoustic model decides what sub-bands of the audio are important. It may be adaptive depending on the type of programming involved. For example, dialog and music scores in a movie may require different sub-band scaling.

you have to generate a sample roughly every 20 µs. Of course, audio converters usually have buffers.

And then, there's the encoding and decoding, which is extremely CPU intensive. Encoding MPEG is about 100 times more CPU intensive than decoding and is often done offline. The video is encoded and stored on disk or tape, before being played back as a datastream.

An MPEG decoder has to perform the decoding in real time, while synchronizing the decoded video with the audio. If the video decoder gets behind or ahead, it may drop or freeze frames.

As you can see, it's important to quantify the computing resources and I/O needs so the required throughput is guaranteed and the various components remain synchronized.

COMPRESSION

If you're interested in learning about multimedia, there's no shortage of information.

In "HDTV—The New Digital Direction" (*INK* 86), Do-While Jones covers multimedia standards, offers some TV history, and overviews MPEG, a common encoding standard. Mike Podanoffsky ("Compressing Audio and Video Over the Internet," *INK* 86) describes JPEG decoding and gives insight into multimedia compression algorithms.

I also recommend *Video Demystified*, which is an excellent reference on all aspects of digital video. If you still feel understimulated, check out *Digital Audio* and *Digital Video*. These books have almost everything about digital audio and video, the conversion from analog to digital and back, the transmission of digital information, as well as compression techniques, error detection, and correcting algorithms.

Now that you know where to go for the details, what things are you likely to find in a multimedia system? Let's look at digital video first since it's fairly complex.

Video requires high data rates to transmit digitally. However, its information content is low, so you can apply compression algorithms that greatly reduce the amount of data you need to transfer or store.

The cost of this compression is, of course, the compression and decompression itself. In general, the higher the compression ratio, the more computing resources required.

Some tradeoffs are usually possible, depending on the application. In teleconferencing systems, we worry about the end-to-end latency of the videos communication path. We need to compress a real-life video feed, transmit it over a long-distance channel (e.g., a wide-area network connection, phone lines, or the Internet), and decompress the video at the other end without too much latency.

If the video stream has much latency, the audio needs to be delayed as well so they remain synchronized. But, communicating with delayed audio is annoying. So, it might be worthwhile to use less aggressive video compression and sacrifice frame rate and resolution to make it fit through the channel.

Two teleconferencing standards—H.320 (ISDN) and H.324 (telephone line)—encompass components that describe video and audio codec algorithms. They also address signaling, so teleconferencing equipment from different vendors can communicate.

At the other extreme sits the entertainment industry, which uses digital video and audio to offer programming to consumers. In this application, the distribution medium—satellite, CD-ROM, DVD, or even radio broadcast from local TV stations—is the bottleneck.

To fit the particular programming source and the consumer, spend some time analyzing the video information, choosing a compression strategy, and adapting the algorithm and the available resolution.

A feature-length movie, which may be distributed using DVD, needs to be high resolution and 16:9 (i.e., letter box) format,

Instruction	Data Types	Description	Instruction	Data Types	Description
PADD[BWD]	B,W,D	add with wraparound	PUNPCKHWD	W->D	unpack high word to double
PADDS[BW]	B,W	signed add saturated	PUNPCKHDQ	D->Q	unpack high double to quad
PADDUS[BW]	B,W	unsigned add saturated	PUNPCKLBW	B->W	unpack low byte to word
PSUB[BWD]	B,W,D	subtract with wraparound	PUNPCKLWD	W->D	unpack low word to double
PSUBS[BW]	B,W	signed subtract saturated	PUNPCKLDQ	D->Q	unpack low double to quad
PSUBUS[BW]	B,W	unsigned subtract saturated	PAND	Q	AND
PMULH[W]	W	packed multiply high	PANDN	Q	AND NOT
PMULL[W]	W	packed multiply low	POR	Q	OR
PMADDWD	W->D	packed multiply-add	PXOR	Q	XOR
PCMPEQ[BWD]	B,W,D	compare equal	PSLL[WDQ]	W,D,Q	shift left logical
PCMPGT[BWD]	B,W,D	compare greater than	PSRL[WDQ]	W,D,Q	shift right logical
PACKUSWB	W->B	pack word to byte, unsigned saturated	PSRA[WD]	W,D	shift right arithmetic
PACKSSWB	W->B	pack word to byte, signed saturated	MOV[DQ]	D,Q	move to/from MMX register
PACKSSDW	W->D	pack double to word, signed saturated	EMMS	—	empty MMX state
PUNPCKHBW	B->W	unpack high byte to word			

Note:
B -> packed byte (8 bytes in 64-bit MMX register)
W -> packed word (4 words in 64-bit MMX register)
D -> packed double (2 double in 64-bit MMX register)
Q -> 64-bit MMX register

Table 1—Here we have the MMX instructions. Although Intel claims there are 57 new instructions, many are just variants of the instructions for the different packed data types. Source operand to MMX instructions can be any valid Pentium address mode, but the destination must be an MMX register [1].

yet have a frame rate of only 24 fps. A news program, by contrast, might be OK at a lower resolution and in 4:3 aspect ratio.

The two MPEG standards address these issues. The first, MPEG-1, deals with compressing video and audio for use on CD-ROMs. MPEG-2 concerns multiple formats.

MPEG, however, comes at a cost. Even though it has aggressive and adaptable compression possibilities, much of the technology and effort in MPEG is in the encoder.

MPEG uses spatial and temporal compression techniques. It analyzes each frame of the video sources and computes the differences between frames as well as compressing much of the information in a single frame.

The data is then sent as groups of picture (GOP) frames. Each frame contains some reference and some incremental frames that reconstruct each frame in sequence. The GOP, in a sense, defines the compression algorithm itself. Do-While's article reviews this in more detail.

The encoder's job is to analyze the programming and come up with a good strategy of encoding the video data. Sometimes, operators have to select parameters so video quality is preserved.

In a nutshell, MPEG is optimized for decoding. On average, the encoding process is about 100 times more expensive than decoding.

Program insertion is a tricky area that falls under real-time processing. Consider a network transmitting MPEG-2-encoded programs to stations rebroadcasting the programs to end users. Because the program is MPEG-2 encoded and the compression algorithm is partially based temporal compression, which mostly involves sending incremental frames through the system, how does the local digital TV station overlay local programming on the network feed?

They can decode the stream into uncompressed or even analog video, overlay the information, and then encode the information before sending it to consumers. While this may work, encoding MPEG is expensive and TV stations can't afford to add latency to the programming.

Some systems can do this now, but the details are proprietary. As we move into the digital-TV age, there will be a lot of opportunity to work on such systems. And, PC-based real-time multimedia processing systems will likely get a big chunk of the market.

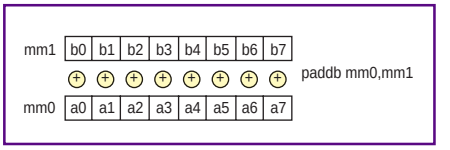


Figure 2—The `paddb` instruction adds eight bytes packed into a quad-word (64 bit) `mm1` to another byte-packed quad-word `mm0` and stores the result in `mm0`. The actual computation is eight byte-wide adds in parallel: a(0) = a(0) + b(0), a(1) = a(1) + b(1), ... a(7) = a(7) + b(7).

Of course, there are other opportunities for cool applications of real-time processing of video and multimedia information using PCs. Consider the special effects movie industry.

I recently visited a special-effects company, and while there is a big presence of SGI in this field, this industry is fully aware of the economy of using PC architecture in their rendering systems.

The stuff they do is pretty serious. Film is digitized at 4000 pixels per line and at 16:9 aspect ratio, so we're talking about 2250 lines or 9 million pixels.

To get good color resolution, you might use 12-bit RAMDACs and 30-bit color. Now imagine doing this at 24 fps, and we're now up to 200+ MHz video rates and about 800+ MBps of data….

Later, I'll provide some code examples of using Pentium MMX for chroma-key processing. This special effect is commonly used to make small models appear against a background—like a spaceship against a picture of a planet, or your local weather reporter standing by the weather map.

Compressing digital audio, while not as compute-intensive as video compression, is tricky. Humans can deal with noise in video (remember there's not much information there), but dislike noise, aberrations, or artifacts of our compression in audio—especially if the audio is in multiple channels.

There are two methods for compressing audio. One is to reduce the number of bits in each audio sample by using nonlinear quantization. Here, we use many quantization steps at lower sound levels than at high sound levels.

At high sound levels, humans aren't as sensitive to different sound levels as they are at lower sound levels. If you go to a concert, you don't know exactly how loud it is—it's just loud. However, it's possible

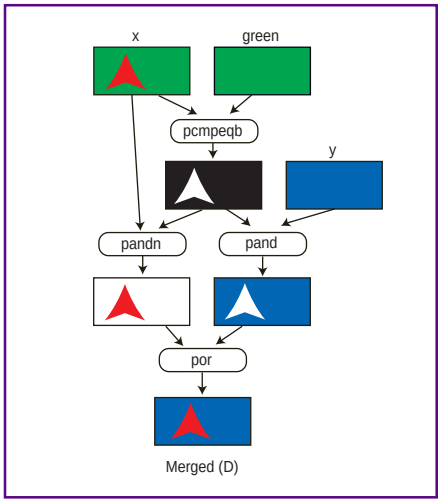


Figure 3—The image containing the chroma key (x) is compared against the key color (green), and a mask is generated. The mask is inverted and ANDed with the x to generate only the foreground image and ANDed with the y image to generate the key background. Finally, the two images are combined using a logical OR.

to discern whispering, even with some amount of ambient sound.

Nonlinear quantization is used in telephone systems to squeeze 12-bit audio into 8-bit samples. The encoding is usually exponential, and two different encoding tables are used today—μ-law and a-law.

Another method to reduce or compress audio is masking. It can be done when two tones are close together—say, 1000 and 1010 Hz—we can't hear the second tone if one is significantly louder.

To take advantage of this, some compression algorithms used in MPEG filter audio into 32 sub-bands and then analyze them to determine which frequency components to send on. The parameters for analyzing the bands and scaling the components are based on a psychoacoustic model, which is an experimentally derived part of the encoder, and adapt themselves to the audio program. Figure 1 shows the block diagram of an MPEG audio encoder.

With MPEG standards, the encoder can use proprietary components to analyze video and audio sources and find optimum compression parameters for a particular program. The decoder, however, can just be a dumb piece of hardware that simply decodes the MPEG datastream. The datastream is then much closer to a stream of instructions to perform than digital data values.

VENDOR OFFERINGS

Now that we've reviewed the kinds of things we are likely to deal with in a multimedia-based system, mostly decoding or encoding video or audio, let's look at some of the components and tools available.

Whenever we're dealing with fairly standard encoders and decoders, we can use off-the-shelf hardware components. Recall I mentioned that the MPEG encoder is fairly complex and sometimes proprietary, but the decoder is simply a machine that reconstructs video frames from the MPEG stream and either outputs them digitally for further processing or converts them to an analog format using a frame buffer. The same has to be done with audio, although the technique differs slightly.

One such board is Traftech's Zephyr shown in Photo 1. The Zephyr is a PC/104+ based MPEG decoder board that decodes MPEG-1 into video and audio streams without processor intervention, except for managing the datastream into the board. However, at MPEG-1, the input data rate is only 1.5 MBps (196 kBps), which is the raw data rate from a CD-ROM.

Intel, by contrast, chose to include DSP-like instructions on a general-purpose processor—the Pentium MMX. Both Adastra and Teknor have announced or are shipping Pentium MMX-based PC/104 CPU boards.

MMX adds eight 64-bit registers and 57 new instructions, listed in Table 1, to the Pentium architecture.

These instructions target the data manipulation used in multimedia and signal processing, such as finite impulse response (FIR) filters, discrete cosine transforms (DCT), and the logical operations of image processing.

More importantly, however, the MMX architecture introduces new data types that increase the data throughput for these applications. These data types let many of the new MMX instructions operate on more than one data item at the same time (i.e., SIMD; single instruction/multiple data).

Contrast this to typical microprocessor instructions, which operate on one data item at a time. The new data types in MMX are packed byte, which packs eight bytes into a 64-bit MMX register, packed word (four per 64 bit), and packed double word (two per 64 bit).

The MMX instructions then operate on the packed words as if they were individual data words. Figure 2 shows what a `paddb` (packed add byte) instruction does.

Let's look at an example. A common special effect for TV and film is chroma keying. Everybody's seen the weather reporter standing in front of the weather map on your local TV station. Here's how they do it.

The weather reporter stands in front of a screen with uniform color, usually green or blue, and gives the forecast. The background—here, the weather map—is a still

Listing 1—The inner loop of a chroma-key algorithm is implemented via Pentium MMX instructions. It performs one cycle, which encodes eight pixels of video-frame data at a time.

```
/* The chroma-keying function
 * x points to the frame with the chroma background
 * y points to the frame with overlay
 * g points to the key color, this is constant for the whole frame
 * d points to the destination of the result */
chroma(x,y,g,d)
register unsigned char *x,*y,*g,*d;{
  _asm{
    mov       edx,g
    movq      mm0,[edx]    /* load green constant */

    mov       edx,x
    movq      mm7,[edx]    /* load x, the image with chroma */
    mov       edx,y
    movq      mm6,[edx]    /* load y, the background */

    pcmpeqb   mm0,mm7      /* compare x with green mask -> mm0 */
    pandn     mm7,mm0      /* mask off chroma background */
    pand      mm6,mm0      /* mask off foreground in background */
    por       mm6,mm7      /* combine image */

    mov       edx,d
    movq      [edx],mm6}}  /* save combined image */
```

image filmed with a different camera.

Now comes the magic. A special box, usually integrated into the video switch, takes the image of the weather person and substitutes the background image whenever it finds the color in the image.

In an analog video system, you simply filter the video image for the key color. NTSC shows up as a pure sine wave that's phase shifted from the color-burst signal.

You can also do this special effect digitally, but it's a bit different. In digital video or film, each pixel has a word indicating the color hue and saturation, either by explicitly encoding the RGB value or by using a color map. The word size usually varies between 8 bits for NTSC video and 24 bits for deep-color spaces. In any case, a single value in the color space now represents the color key.

Let's do this with MMX instructions. The foreground image (i.e., the weather reporter) is image *x*, and the background is image *y*.

The algorithm first constructs a mask by comparing each pixel value in *x* to the chroma color (green). When it's green, the mask has a one bit and nongreen is masked as zero. The `pcmpeqb` instruction does this for a byte-packed word on eight pixels at a time.

The mask is ANDed with the *x* and *y* images. However for *x*, an inverse mask extracts the foreground. MMX has a special instruction of packed AND NOT (`pandn`), which implements an AND with the logical NOT of one of the operands. Image *y* is simply ANDed with the mask using the packed AND (`pand`) instruction.

We now have two images—a foreground and a background with a hole the shape of the foreground image. The final operation is a logical OR to combine the images.

Figure 3 shows what this looks like, and Listing 1 shows the heart of the algorithm. With the MMX architecture, eight 8-bit pixels can be processed at one time, using four instructions. Compare this to the approximately 24 regular Pentium instructions, which you have to use without the MMX extension to process the pixels.

A solution somewhat between software implementation and fixed hardware uses configurable hardware. Virtual Computer Company has a PCI-based FPGA card that can be configured dynamically to perform hardware-based computation.

VCC's card also sports local fast SRAM, so it functions like a array processor or implements encoding or decoding functions. Their development environment enables the design of hardware objects that are implemented on the card interface with library routines to C and C++ programs.

Algorithms like DCT, which form the inner loop of the MPEG and JPEG compression techniques, can be substantially accelerated over standard processors. For example, a VGA-resolution JPEG decoder implemented on this card FPGA runs at ~50 fps, compared to ~6.5 fps on a 233-MHz Pentium [2].

However, the RTOS system designer is still left wondering how to tie it all together. One solution is Microware's multimedia system architecture, Maui, for its OS9 RTOS.

Maui enables the application developer to program against a common API, which gives abstractions for video and audio devices. You then don't have to worry about what kind of hardware is present in the system.

It's even possible to implement some of the device as software components without recoding the application program. Figure 4 diagrams this system architecture.

WRAP UP

Multimedia is an exciting area with a lot of new development. Processor vendors are developing architectures to accelerate multimedia applications, and hardware vendors are offering specific board solutions.

As well, a new breed of computational element—reconfigurable hardware—is changing the way we solve some of these problems with "soft" hardware. Watch for developments. RPC.EPC

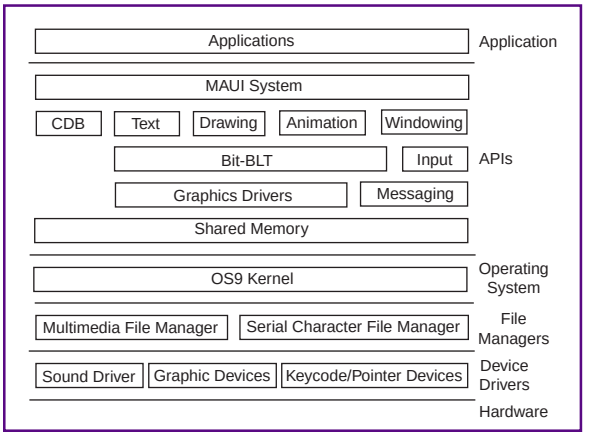


Figure 4—Microware's Maui architecture lets the developer program against a consistent API, regardless of how the devices are implemented. For example, the application program doesn't care if an MPEG decoder is implemented in hardware or software.

Ingo Cyliax has been writing for INK *for two years on topics such as embedded systems, FPGA design, and robotics. He is a research engineer at Derivation Systems Inc., a San Diego-based formal synthesis company, where he works on formal-method design tools for high-assurance systems and develops embedded-system products. Before joining DSI, Ingo worked for over 12 years as a system and research engineer for several universities and as an independent consultant. You may reach him at cyliax@derivation.com.*

REFERENCES

[1] Intel, *Intel MMX Technology Overview*, 1997.
[2] J. Heron, D. Trainor, and R. Woods, *Image Compression Algorithms Using Reconfigurable Logic*, Queen's University of Belfast, Northern Ireland, and Integrated Silicon Systems, Belfast, Northern Ireland, 1997.

RESOURCES

K. Jack, *Video Demystified*, Hightext, San Diego, CA, 1996.
K.C. Pohlman, *Principles of Digital Audio*, Sams, Indianapolis, IN, 1989.
J. Watkinson, *The Art of Digital Audio*, Focal Press, Oxford, UK, 1994.
J. Watkinson, *The Art of Digital Video*, Focal Press, Oxford, UK, 1994.

SOURCES

Zephyr MPEG decoder PC/104+ Card
Traftech, Inc.
6981 Millcreek Dr., Ste. 30
Mississauga, ON
Canada L5N 6B8
(905) 814-1293
Fax: (905) 814-1292
www.traftech.com

VNS-686M Pentium MMX CPU Card
Adastra Systems Corp.
3988 Trust Way
Hayward, CA 94545
(510) 732-6900
Fax: (510) 732-7655
www.adastra.com

Viper820 Pentium MMX CPU Card
Teknor Microsystems, Inc.
616 Cure Boivin
Montreal, QC
Canada J7G 2A7
(561) 883-6191
Fax: (561) 883-6690
www.teknor.com

PCI6200, PCI reconfigurable processing unit
Virtual Computer Corp.
6925 Canby Ave., Ste. 103
Reseda, CA 91335
(818) 342-8294
Fax: (818) 342-0240
www.vcc.com

IRS

416 Very Useful
417 Moderately Useful
418 Not Useful

Applied PCs

Fred Eady

A New View

Part 2: Data Acquisition with Virtual Instruments

Virtual instrumentation is only a good deal if it can handle real-world problems. So, Fred looks beyond the software of last month and puts together a data-collection device. Next month, he'll show you how to measure this month's data.

Since I talked to you last, lots of "instrumental" events have occurred. Fortunately, these events weren't virtual like the National Instruments VIs I introduced to you last time.

There's National Instruments *hardware* in Circuit Cellar's Florida room! That means hands-on-plug-it-in-load-drivers-and-run time.

Last time, I discussed the software side of the National Instruments LabVIEW. We saw that, by combining prewritten and specialty instrumentation components, just about any data-collection device within reason could be realized.

As you're know, we live in a physical world. It's difficult to collect "virtual" data to correlate with real-world events. Hence, "real" hardware must compliment the virtual mechanisms of the LabVIEW instrumentation software to collect real usable data. So, let's go into the goodie box I got from National Instruments and put together the beginnings of a real-world data-collection machine embedded style.

ENEMY IDENTIFICATION

For the soldier, proper identification of enemy forces is key to battlefield success. For the data-collection science officer, identification of the data to be collected is just as important. So, before we throw hardware at a problem, we need to know what we're throwing at.

Do we want to collect analog or digital data? What types of sensors will we need? Do the sensors need conditioning? What embedded engine will be used? Is the operating system 16 or 32 bit? Do we need standard peripherals like hard disks or CD-ROM drives?

Rather than go through an embedded lifetime wondering, let's start at the beginning and work out the answers as we go.

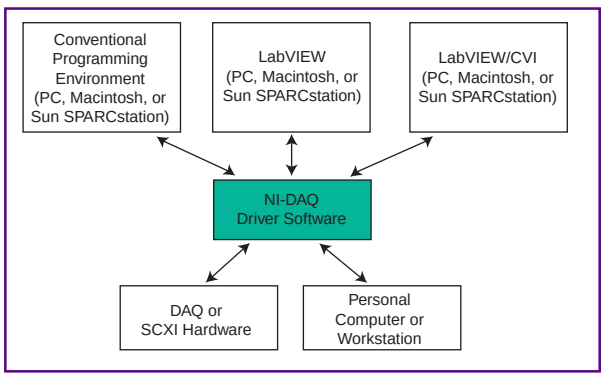


Figure 1—This is a little busy but it shows how all of the products are interwoven.

SELECTING EMBEDDED ENGINE

The first logical thing to do is to select the embedded platform we'll base the rest of the data-collection hardware on. We could start with any data collection components, but remember this is an *Embedded PC* column.

The data-acquisition card I pulled from the goodie box is AT compatible and ISA based. That means that the embedded engine we select must be AT compatible and accept ISA expansion cards. Another plus would be to have the capability of using PC Card technology.

A deeper look into the National Instruments goodie box offers some

PCMCIA data-collection and control possibilities. I'm getting old and not seeing as well as I used to, but I think I see some PC Cards in the box, too.

The very first and preferred platform that came to mind was the new EXPLR2. It has ISA slots and a PC Card socket onboard. It can do IDE, PS/2 mice, and floppy drive, too. Perfect!

Not. The EXPLR2 is '386 based. Although LabVIEW can run in a 16-bit environment, I don't feel like fooling with Windows 3.11 right now. If you want to, that's OK. I'll watch.

As you probably ascertained, there's a bunch of embedded components in the Circuit Cellar Florida room, so I decided to think about this a little more and come up with a more suitable platform.

After some thought and file-cabinet searching, I came across the Tempustech VMAX SBC 301. This embedded system is based on a 66-MHz '486 and is capable of playing with other components via a passive ISA backplane. I can add an IDE drive and CD-ROM as well as boost the memory up to Windows 95 levels. This is it!

Hmmm…. Seems I need to add a video card to make the system complete. No problem. I look for and find a spare video card. Well, this ain't it. The video card is PCI not ISA.

The Tempustech backplane has no provision for PCI. Being originally from Tennessee, the Grandpa Jones "Hee Haw" tune came to my lips just a little bit skewed, "I searched the world over and thought I found true love, but you were PCI and psssst you were gone."

Great idea, wrong interface. This is getting down to where I'm gonna have to really do some thinking if I don't find the right stuff pretty soon.

A little more digging brought out things I thought I had lost or loaned out and a Teknor VIPer-806. I've ridden this baby many times and put it up wet. There was a hint of moisture in the antistatic bag even.

As far as the VIPer is concerned, everything is there with the exception of the ability to incorporate an ISA daughter card. There's onboard video, onboard IDE and floppy capability, and plenty of memory for Bill's memory-hungry software.

Problem is, how do I get an ISA data-acquisition card linked with this puppy? Wait a minute—the VIPer806 hardware resides on an ISA-compatible card.

I dug out my VIPer manuals, and sure enough, "The VIPer806 is a multipurpose computer. It can be used as a single-board computer in conjunction with a passive backplane." Enough said.

I pulled the Tempustech out of its native backplane and inserted the Teknor. A few pulls and pushes and a whap to the power switch kick-started the Teknor BIOS, and up came an intelligent message on the display.

Yes! No smoke is good. Things are beginning to shape up. I'm sure that Tempustech backplane never thought it would see VIPer gold contacts.

After a few more tugs and grunts, a floppy, a 2+ GB IDE drive, and a CD-ROM drive join the virtual-instrument party. Don't you love this embedded stuff?

If you've read my previous columns, you know that in the middle of articles I sometimes park the truck and take a walk. Well, let's park the truck here and traipse (that's Tennessee for "walk") into the bushes.

A while back, I mentioned some of the stuff I use to make embedded development life better? You'll recall I have multiple machines all over the Circuit Cellar Florida room and I hate to run multiple keyboards, mice, and monitors all over the place.

Photo 1—This is more like it. Notice that three DMA channels are used by the DAQ card.

This particular application is another good example of how useful the Vetra MegaSwitch is. Within a few minutes, I had the VIPer and all its peripherals marching to 60 Hz. All I had to do was hook the VIPer up to an unused set of MegaSwitch cables and punch a button to select it.

The VIPer is now embedded platform 6 of 8. Hmmm. Sounds like a Borg thing to me. Anyway, back in the truck.

FUELING THE EMBEDDED ENGINE

We're at the point where we are ready to load one of Bill's operating systems, Windows 95. We could go back in time and use Windows 3.11, but I don't think so. When they put 3.11 on CD-ROM, I'll load it.

As it turns out, loading any version of Windows, CD-ROM or not, would have been easier than figuring out why I couldn't get the IDE drive to come ready. Seems the drive I selected didn't like 32-bit or block transfer mode with the VIPer.

I really don't have time to play with that now. Lucky for you, you're reading this in "relative" time, but it took a day or so for me to reach that conclusion and get a good load of 95.

A load of LabVIEW with the appropriate NI-DAQ equiva-

Photo 2—The "wordy" descriptions are nice. The temperature measurement box is checked for use with the PIC14000.

lent drivers is the next logical step. While it's loading, let's talk about what NI-DAQ and LabVIEW have in common.

NI-DAQ software ships with every piece of National Instruments DAQ hardware. NI-DAQ includes an extensive library of callable functions for the VI programmer. These functions include routines for analog input (A/D conversion), analog output (D/A conversion), buffered data acquisition, waveform generation (timed D/A conversion), digital I/O, counter/timer operations, calibration, messaging, and acquiring data to extended memory. Whew!

NI-DAQ also includes high-level DAQ I/O functions as well as low-level DAQ I/O functions to aid the user in producing a workable data-acquisition solution and at the same time maintain high performance and maximum flexibility.

High-level functions include acquiring particular numbers of data points or streaming data to disk. Low-level functions are much like the ones us bit-bangers are accustomed to, like writing to registers on the DAQ hardware card.

(Do you get the idea that DAQ is short for Data AcQuisition, and maybe NI is short for National Instruments? Hmmm….)

Summing it up, NI-DAQ software is the mediator between the embedded computer and the DAQ hardware. All the housekeeping, like programming interrupts and DMA controllers, is done by the NI-DAQ software.

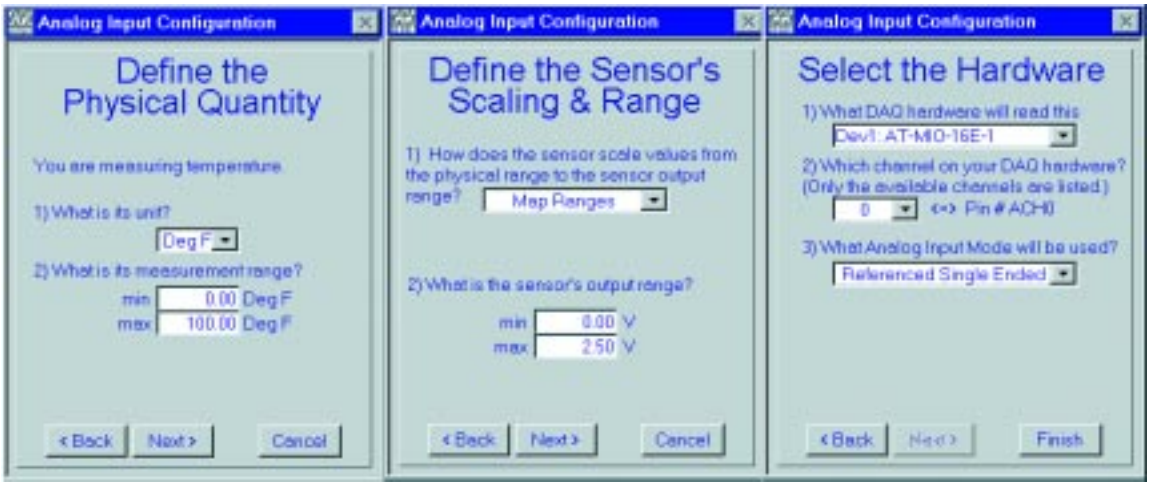


Photo 3—Is this flexible or what? a—We can choose any Centigrade or Fahrenheit temperature range to match our particular application. b—Not only can we select a temperature range, but we can tailor the measurement parameters to match our sensor output. c—Pretty straightforward. Every channel on every DAQ card in the system is easily identified.

If all that isn't enough, NI-DAQ also maintains a consistent software view among all of its versions that lets your data-acquisition applications move between many platforms with minimal coding effort. Figure 1 is a picture worth almost as many words as I have used to describe it.

INSTALLING DAQ AND LABVIEW

I promise, we're going to get back to it and install LabVIEW on the VIPer, but before we continue, let's talk about the data-acquisition hardware I pulled out of the goodie box.

We will be installing and configuring the AT-MIO-16E-1. How do I know that? It's silkscreened on the card. How did I decide which card to use? Well, that's what was in the box! Actually, I did ask for this particular card because I thought it would be of interest to most of you.

The AT E-series boards are software configurable as well as plug-and-play capable. To complete our installation, we must perform two types of configuration: bus related and data-acquisition related.

Bus-related configuration consists of setting the base I/O address, DMA channels, and interrupt channels. Data acquisition-related configuration has to do with the analog input polarity and range and other related entities.

Installing the DAQ board was just a push into the Tempustech backplane. LabVIEW's install was more involved.

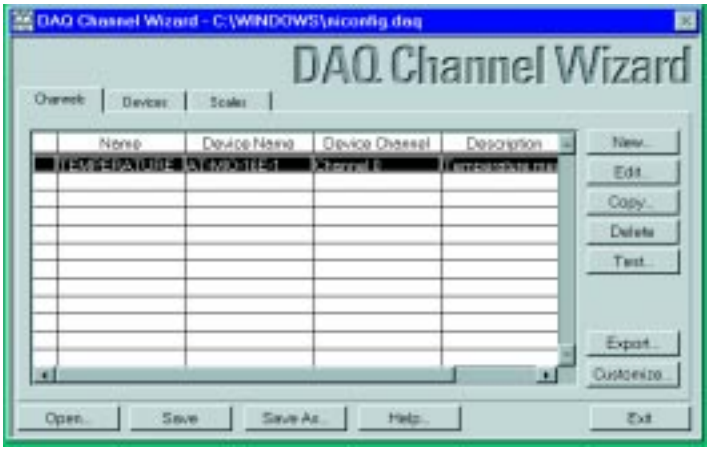


Photo 4—As you can see, repeating the definition process for every channel would automatically make another entry into the channel table.

Although the software-installation process was similar to other installs, I was glad I read the doc before jumping in. There are a few options that the install program asks you to choose during the install process. It helps to know some of the acronyms like VISA and GPIB.

GPIB is probably obvious to most of you, but VISA isn't a credit card. VISA in this case is Virtual Instrument Software Architecture, a single interface library for controlling GPIB-based instruments. Yep, LabVIEW not only is the instrument, it can control the instrument, too. The install went smoothly, but I noticed it took longer with the 100-MHz VIPer than my 233-MHz yin-yang desktop.

Now that Bill's OS and LabVIEW coexist on the spinning disk, it's time to do that configuration thing. I've always hated to configure things. Especially something that looks as complicated as this DAQ board.

But, someone at National Instruments must have taken my "It doesn't have to be complicated to be embedded" motto to heart. From the LabVIEW perspective, DAQ configuration isn't complicated, either. (It's great to know that somebody at National is reading my stuff.)

Configuration is accomplished by using NI-DAQ configuration utility, Niconfig. I had a few problems getting the VIPer and Win95 to recognize the DAQ card. So, I went back and read the install instructions again. And again. And again.

Finally, out of desperation, I deleted the DAQ card from the Win95 hardware config and performed a refresh. I noticed before that the DAQ card was addressed in the FF80–FF9F area. I thought it odd, but that's what Bill's software saw, so I let it go.

After I deleted the card and refreshed the Win95 hardware config, the new address was 0120–013F, and the IRQ had changed from 11 to 15 with three DMA channels tagging along. Also, I noticed that the second time around the National Instruments logo appeared in the Win95 device manager window, and the serial number of the DAQ card was picked up as well. Wow!

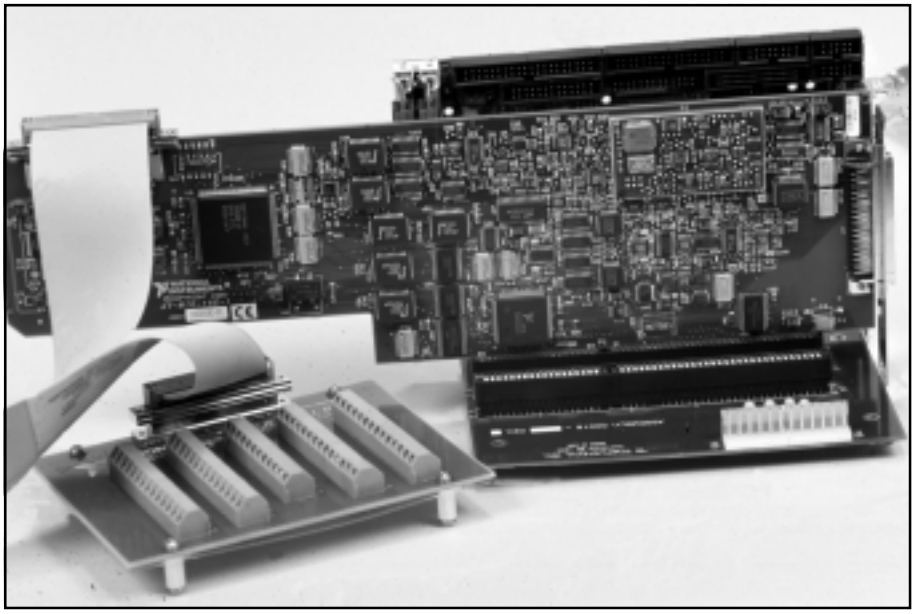


Photo 5—Here are all the goodies that I found in my box. The large DAC board also comes in a PC Card version.

With all that new stuff showing up, this card had better pass the hardware test. It did. I chalked the mishaps up to Bill's software not National's. Photo 1 is the fruit of my labor.

That about does it for the bus-related config. Keeping with the uncomplicated idea, the DAQ Channel Wizard guides us through the data-acquisition configuration. We can assign a real name to our inputs.

Photos 2 through 4 show how easy it is to configure an analog input channel. This process is repeated for the number of channels we need for our application. Once the DAQ card is configured, there's even some ready-to-roll apps included.

SUMMING JUNCTION

We now have a combination of hardware and software components that make up the basis of a powerful data-acquisition and control tool. Only one last piece.

There's a funny looking 68-pin connector on the back of the DAQ card. I can't stuff wires into it to make my interface connections. Back into the goodie box.

Aha! An interface cable. And look—a screw-down interface block for each wire! Check it out in Photo 5. The hardware is complete, and just in time. I'm out of paper.

Next month, I'll put the ready-to-run virtual instruments (National Instruments Virtual O-scope and DMM) to work looking at input and output associated with the DAQ platform we just built. Remember, it doesn't have to be complicated to implement data-acquisition and instrument control the embedded way. APC.EPC

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

SOURCES

LabVIEW
National Instruments Corp.
6504 Bridge Point Pkwy.
Austin, TX 78730
(512) 794-0100
Fax: (512) 794-8411
www.natinst.com

VMAX SBC 301
Tempustech
295 Airport Rd.
Naples, FL 34104
(941) 643-2424
Fax: (941) 643-4981
www.tempustech.com

VIPer806
Teknor Microsystems, Inc.
616 Cure Boivin
Boisbriand, PQ
Canada J7G 2A7
(514) 437-5682
Fax: (514) 437-8053
www.teknor.com

MegaSwitch
Vetra Systems Corp.
275-J Marcus Blvd.
Hauppauge, NY 11787
(516) 434-3185
Fax: (516) 434-3516
eemoline@comvetrasyst

IRS

419 Very Useful
420 Moderately Useful
421 Not Useful

FEATURE ARTICLE

Brian Millier

Digital Attenuators

Given the CD quality we've come to expect, volume control needs to be handled digitally. Brian shows us how using Dallas digital attenuators, TI op-amps and virtual grounds, and a musical digital interface. Tune in.

The mechanical potentiometer has been around for as long as many other electronic components. Until recently, however, it hasn't been widely available in a digital format.

For set-and-forget apps (e.g., calibration via DC voltages), you could use a DAC, assuming its voltage range matched the application. A four-quadrant multiplying DAC lets you handle bipolar signals over a wide range. For a number of years, Xicor produced a line of EEPOTS, which were low-resolution (100 step) DACs coupled with EEPROM memory and a serial interface.

Probably one of the most common uses of potentiometers is for volume control on the numerous consumer audio items available today. Human hearing covers an extremely wide range of amplitudes. How much of this dynamic range is actually needed varies, but for listening to music, CD quality is now the norm. A 16-bit digital storage format gives you a dynamic range of about 94 dB.

Volume controls for such applications must possess high resolution and low inherent noise so that the high-quality signal is not degraded. To match the human ear's response characteristics, a volume control—called a fader in the music industry—must include a logarithmic curve or taper.

For music, it's often necessary to make gradual changes in sound amplitude. Unless these changes are made in very small jumps, we perceive them as a form of distortion commonly referred to as "zipper noise" because it sounds like a zipper being moved.

In the past, these characteristics made it difficult to control volume digitally, and many of the schemes were quite complicated. High-resolution multiplying DACs were expensive, and the presence of many digital signals going to the device tended to produce some noise in the audio signal.

You could use a DAC to produce a DC voltage which then controlled a voltage-controlled amplifier (VCA). Unfortunately, VCAs are tricky circuits to design, and they have their own dynamic range and linearity problems. This method is complex and expensive, but it performs well.

Recently, Dallas Semiconductor devoted some of their engineering prowess to attack this problem. As a result, they now offer a full line of digital attenuators that address these problems. Let's take a look at how they work and see how they can be used for a common music-studio application—an automated mixing device.

THE DS1800 FAMILY

Members of the DS1800 family vary greatly in features, but they share a basic design. Figure 1 represents the block diagram of one member of the family—the two-channel DS1800.

A resistor ladder connects to a multiplexer, which acts like the wiper in a mechanical potentiometer. The resistors are equal values in the linear models, while the logarithmic models have scaled resistors to provide a pot with a logarithmic taper.

Figure 2 shows the attenuation-versus-position relationship of the DS1800, which is a log device. Notably, the wide dynamic range of +20-dB gain goes all the way down to 80 dB of attenuation, which suits musical applications well.

A control logic unit handles the interface of the parallel multiplexer inputs with the various digital control schemes of the DS1800 family. For all

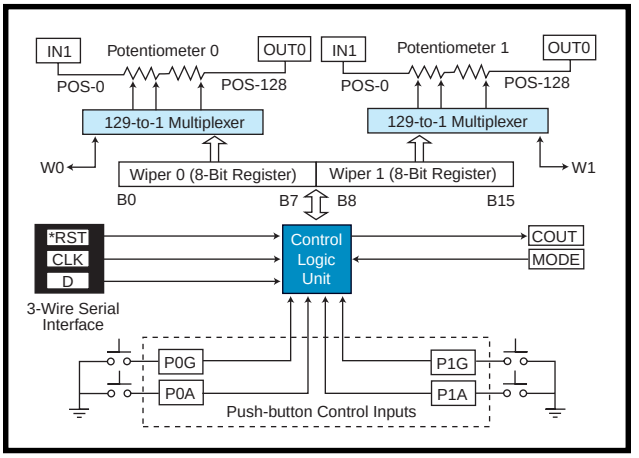


Figure 1—*This block diagram of the two-channel DS1800 shows its push-button as well as its three-wire serial interfaces.*

family members, the number of fast rise-time digital signals entering the device is minimal, and feedthrough of the digital signals into the analog signal path is virtually nonexistent. This is critical for audio applications.

The family consists of a full line of digital attenuators with up to six attenuators per package and resolutions of up to 256 steps. Different devices contain combinations of push-button, two- and three-wire serial, and three-wire parallel interfaces.

The push-button inputs are debounced and pulled up internally to ease circuit design. Although most of the units are volatile, a few models are nonvolatile, like the Xicor devices.

I mentioned the problem of zipper noise earlier, and you might wonder how a device with a maximum of 256 discrete steps doesn't suffer from this effect in a big way. Here's where the magic of the device comes into play.

Dallas incorporated a novel zero-crossing detection circuit into the device. Essentially, it defers any pending changes to the pot's setting until the analog signal entering the pot crosses the zero-voltage threshold.

With no signal present, you can alter the pot's setting without zipper noise. In my tests, I could barely hear any artifacts and they only occurred in quick large volume changes. So, I judged the device to be entirely suitable for critical musical purposes.

PUTTING IT TO WORK

A midi-fader is a device used in a recording studio to adjust the volume level of a specific sound source, such as a vocal or guitar track after it has been recorded to magnetic tape. Since this is often done to several tape tracks simultaneously and involves gradual changes or "fades," it's useful if a computer can control such actions.

Generally, the tape recorder is connected to a computer running a sequencer program that's generating some of the music being recorded. The recorder and computer are synchronized to each other.

Therefore, it's possible for the computer to send out commands to the midi-fader to adjust the volume levels, in much the same way as it sends out commands to the various electronic instruments to play musical notes at the proper time. This stream of digital commands is sent via MIDI (Musical Instrument Digital Interface).

MIDI is a serial, optically isolated, current-loop interface, with a fixed data rate of 32,500 bps and no handshaking signals. The MIDI protocol calls for eight-bit data transfers, with the most significant bit equal to 1 for commands and 0 for data and parameters.

From this, you can see the parameter range is limited to 128 discrete values. Dallas probably had MIDI applications in mind when it designed the DS1800 device with 128-position potentiometers.

The midi-fader listens to the serial MIDI datastream, responds only to the correct commands, and converts this information into a form that can be sent to the DS1800's three-wire serial interface. An LCD with a bar-graph readout of volume level tracks each of the eight channels implemented.

THE ANALOG SIGNAL PATH

Like all members of the family, the DS1800 I chose is a low-power device designed to work at the low voltages present in battery-operated equipment. It operates from 2.7 to 5.5 V, interfacing nicely with the 5-V logic supply used by common microcontrollers.

The downside is that the limit on the analog signal levels these devices can handle is ±2.5 V_{peak}, or about 1.7 V_{RMS}. But, this limit is no problem in nonprofessional music-studio equipment because the –10-dBv levels used for normal line-level signals is significantly less.

As you see in the lower section of Figure 3, the only other active component needed in the analog signal path of these digital attenuators is a single op-amp. Because the digital attenuators operate on a single +5-V supply, the op-amp should also operate on 5 V.

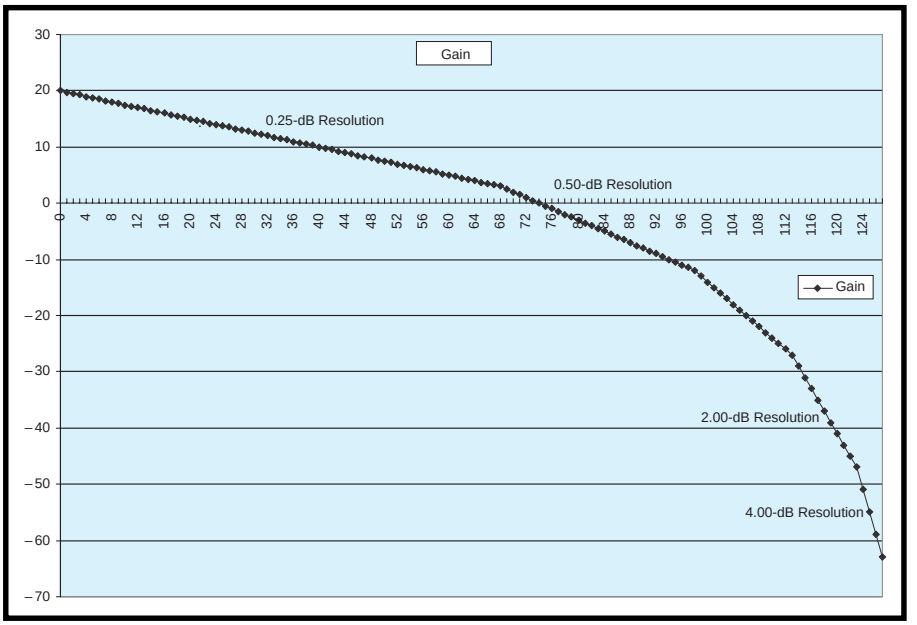


Figure 2—*The resistor chain in the DS1800 is broken up into five segments, with values chosen to effectively produce a logarithmic taper.*

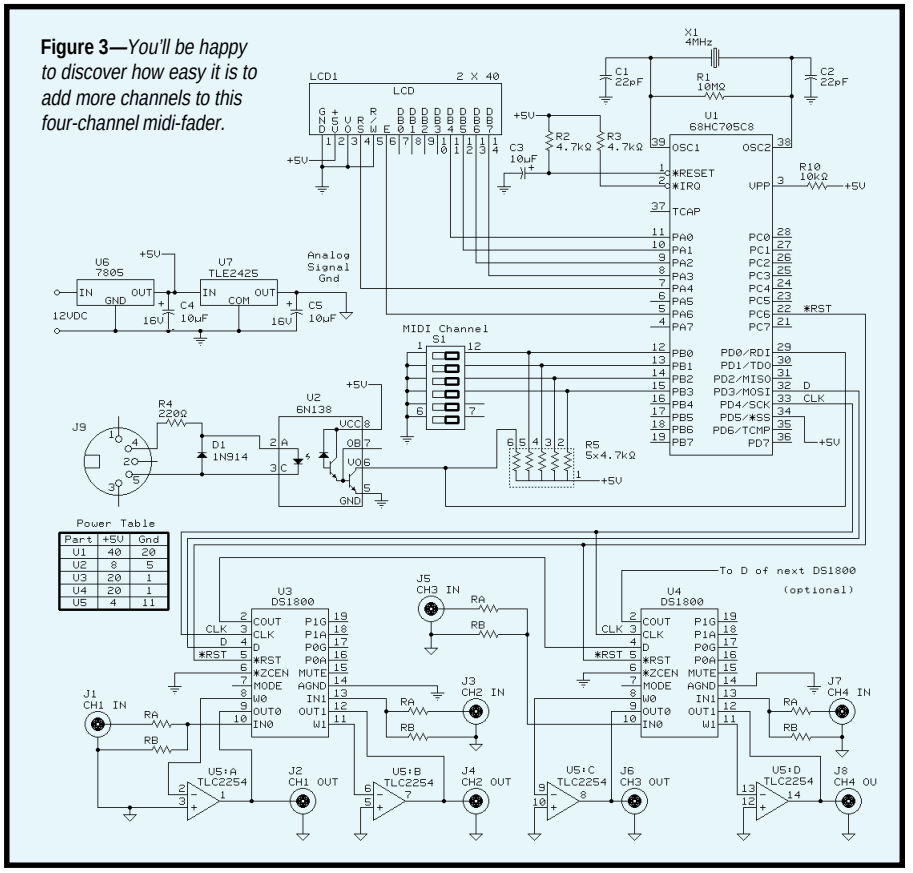


Figure 3—*You'll be happy to discover how easy it is to add more channels to this four-channel midi-fader.*

Of the numerous op-amps I tried, the Texas Instruments TLC2254 quad op-amp performed the best. Other devices suffered from various shortcomings like crossover distortion, low output levels, or low-level oscillations (squealing) at certain volume settings.

Note in Figure 3 that the plus input to the op-amp must connect to a virtual ground equal to half of V_{CC}. I used a Texas Instruments TLE2425CLP virtual ground generator because it provides a stiff ground with no additional circuitry.

All signal inputs and outputs are referenced to this pseudo-ground. Since the optoisolated MIDI input is the only other connection to external devices, floating the signal ground is fine.

The few additional passive components needed for the analog signal path are shown in Figure 3. At each channel input is a voltage divider consisting of RA and RB. You should choose this to limit the amplitude of the output signals to the ± 2.5-V range mentioned earlier, given your signal's input levels and the +20-dB gain present when the DS1800 is fully cranked up.

Note that the op-amp is configured in the inverting mode, so any signals going through the midi-fader are phase inverted. This setup has some ramifications in certain audio applications.

Two of the DS1800's features were left unused. The push-button inputs on the DS1800 enable manual volume control, which I didn't need.

Also, the device incorporates a mute feature. Its 128 pot positions are encoded into the seven least significant bits of its command byte, and setting the most significant bit equal to 1 puts the device into mute mode.

The advantage of this mode over just sending out the lowest-volume setting command is that it lets the user turn a channel on and off without affecting (or memorizing) the volume level in the nonmuted position.

Unfortunately, my sequencer software sends out the same command whenever the mute button is clicked, even though that button has a toggle-type action and indicator. This can result in situations where the midi-fader thinks muting is on and the sequencer thinks it's off.

I was unable to figure out a good solution to this problem. Since the mute push-button input on the DS1800 controls both channels jointly and I wanted each channel to be independent, I decided not to use this feature.

While not shown in Figure 3, in my personal unit, the analog signal section is expanded to support eight channels by replicating the circuitry involving the two DS1800s and the TLC2254. The firmware sends out data for all eight channels, and the cascade capability of the DS1800 makes it easy to add more devices.

THE DIGITAL CIRCUITRY

The DS1800 has a three-wire serial protocol that interfaces to the SPI port present in the Motorola 68HC705C8, as well as similar ports available on other common microcontrollers. Unlike the *Enable line found on most serial peripherals, the DS1800 needs an active-high signal on its *RST pin to initiate data transfer.

Since Dallas Semiconductor anticipated that multiple DS1800 devices would be needed in multichannel audio applications, it provided a simple way of controlling cascade-connected devices via the cascade-out pin, COUT.

For all cascaded devices, the D input is connected to the COUT pin of the previous device in the chain. All DS1800 CLK pins are connected in parallel. To load the cascade-connected DS1800s, first raise the *RST line high, send 16 clock and data bits per connected device, and then lower *RST.

The data effectively ripples its way through all of the connected devices. The data format required by the DS1800 is shown in Figure 4. I chose an SPI data transfer rate of 16 µs per bit, so four devices (implementing eight signal channels) can be quickly updated in about 1 ms.

The only fly in the ointment in this otherwise ideal data-transfer arrangement arises from the fact that the DS1800 expects its serial data to be sent least significant bit first, whereas Motorola SPI ports send data out most significant bit first. Both devices are stubbornly nonprogrammable in this respect, so I need a software routine that does a bit reversal.

In assembly language, this involves only eight Logical Shift Right and eight Rotate Left instructions, but try it in

a high-level language and see how involved it is. It isn't obvious to me why Dallas decided to do it this way, but the rest of the design is so good that I can live with it.

The SCI port on the 'C8 is programmed to the 32,500-bps MIDI rate. The optical isolation required on the MIDI receive line is provided by a high-speed 6N138 optocoupler. This device works well, as does the PC900 device originally specified in the MIDI standard. However, many other common optocouplers are not fast enough, so be careful if you try using something else.

The MIDI standard defines 16 unique channels (1–16) that allow for addressing up to 16 MIDI devices daisy-chained together. All MIDI commands have the MIDI channel number encoded into their four least significant bits. All MIDI devices incorporate some way to set the channel to which they will respond. In this design, it's done by reading a four-section DIP switch, S1, using PB0–PB3, at powerup.

For the front-panel display, I used a 40-character × 2-line LCD. By employing the user-definable characters available on most common LCD modules, a bar-graph readout can be displayed for each channel. This graph resembles the digital VU meters found on modern audio equipment.

Most common LCD modules use the Hitachi HD44780 LCD controller IC or its equivalent, which allows for either four- or eight-bit operation. I wrote a set of four-bit routines that minimize I/O pin usage by using timing loops (rather than polling the LCD's status register), which I've used with small microcontrollers sporting few I/O pins.

While the 'C8 has enough unused I/O port pins to work in a full eight-bit polled mode, I stuck with these routines because they don't hang the program if the LCD is not installed, making its use optional. This allows for a lower cost version of the project.

THE FIRMWARE

While the firmware is not particularly involved, there are some important considerations that must be addressed. The MIDI serial data transfer protocol doesn't include any provisions for handshaking. Therefore, the microcontroller must always be ready to accept MIDI data.

The firmware uses an interrupt-driven SCI input routine which accepts the MIDI data byte, interprets it, and implements it in less than one character time (300 µs).

To complicate matters, the MIDI protocol defines a data-compression feature called "running status." To be of any use for music, the MIDI datastream must be processed with little delay, so elaborate compression algorithms can't be used.

However, music involves a lot of repetitive actions, so whenever any command is the same as the one preceding it, the command itself is not sent—just its parameters. While quite crude, this technique provides some data compression and isn't too difficult to decode by the receiving MIDI device.

The DS1800 digital attenuators are not nonvolatile, and they power up in the position of maximum attenuation. Since I don't want to burden the se-

quencer software with the task of turning up the volume at startup, the midi-fader firmware looks after setting all channels to unity gain when it comes out of reset.

Along with the usual port-initialization tasks, the firmware also reads the MIDI channel DIP switch, assigns the midi-fader to that channel, and puts up a message on the LCD indicating this channel assignment.

Possibly the most difficult part of the firmware to code was that involved in the eight-channel VU meter display on the LCD. Eight custom characters were defined and loaded into the LCD controller. These were basically eight bars of differing heights.

A two-line display gives a resolution of 16 bar heights. The 128-step volume range provided in the MIDI specifications is mapped into these 16 heights using shift commands.

Then, the proper combination of two custom characters is sent to the LCD to make up the bar. I used several data pointers to place the bars in the proper place on the LCD for each of the eight channels. Since volume-change commands can come in quickly, this routine had to be tightly coded.

I won't get into the whole MIDI command set here, but I want to mention a couple things. The MIDI command used by the midi-fader is a control-change command, which has a hex value of B*x*, where *x* is the channel number of the device the command is being sent to.

Following this command byte is the controller number. Since many types of controllers are used in musical instruments, many controller numbers are preassigned to specific controllers.

To make the midi-fader compatible with other similar devices out there, I chose to use controller numbers 16–23 for the eight channels in the midi-fader. The last byte sent is the actual volume-level byte.

Like all data and parameter bytes, the last byte has a most significant bit equal to 0, so its range is limited to 0–127. Three MIDI bytes are sent for each volume change. When several volume change commands appear in a row, the running-status compression reduces this to two bytes per volume change.

Controlling the midi-fader from sequencing software depends on what package you're using. I use a package called Cakewalk, which includes a Fader View window (see Photo 1). After telling the program what channel my fader is set to and that it emulates an MOTU mixing device, I'm off to the races.

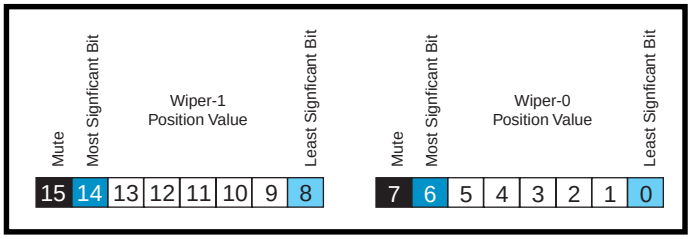


Figure 4—*The DS1800 uses this 16-bit command word structure.*

FADE OUT

I hope that you've gotten a good feel for Dallas's digital attenuator family. I didn't delve into a lot of details that musicians building a DIY midi-fader might want explained, but I hope this article piques your curiosity.

For more information, check Dallas's Web site. It offers excellent datasheets on all of Dallas digital attenuators, free for the downloading.

Brian Millier has worked as an instrumentation engineer for the last 15 years in the chemistry department of Dalhousie University, Halifax, NS, Canada. He also operates Computer Interface Consultants. You may reach him at brian.millier@dal.ca.

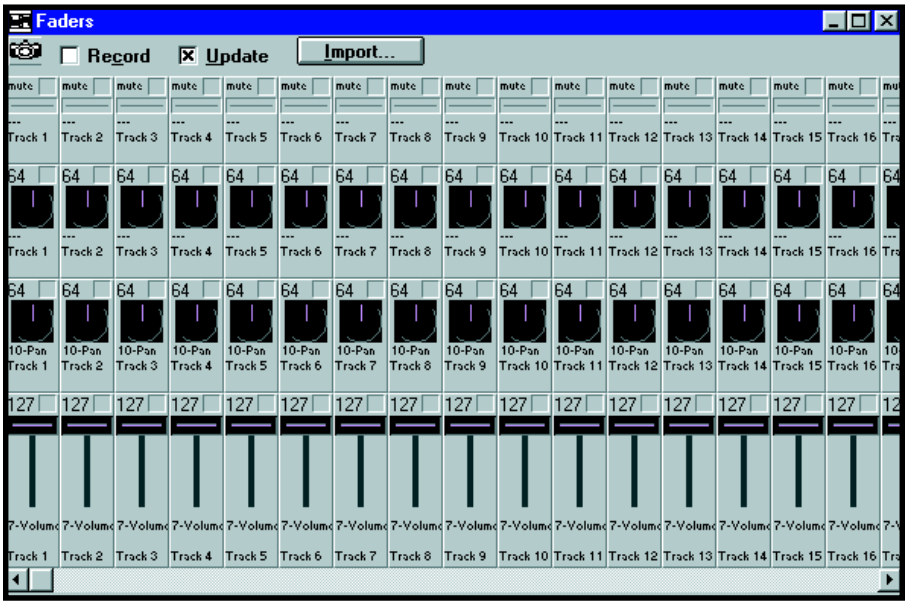


Photo 1—*Popular sequencer software (e.g., Cakewalk) includes a Fader View window designed to operate automated faders like my midi-fader.*

SOFTWARE

The source and object code for the 68HC705C8 microcontroller are available from the Circuit Cellar Web site as well as via <BMillier.chem.dal.ca>.

SOURCES

DS1800
Dallas Semiconductor
4401 S. Beltwood Pkwy.
Dallas, TX 75244-3292
(972) 371-4448
Fax: (972) 371-3715
www.dalsemi.com

TLC2254, TLE2425CLP
Texas Instruments, Inc.
34 Forest St., MS 14-01
Attleboro, MA 02703
(508) 699-5269
Fax: (508) 699-5200
www.ti.com

I R S

422 Very Useful
423 Moderately Useful
424 Not Useful

DEPARTMENTS

FreeDOS and the Embedded Developer

MICRO SERIES

Pat Villani

Kernel Design

Part 1 of 2

Fed up with forking over big bucks for your OS? Pat offers an alternative that's 99.9% MS-DOS compatible. And, it can be used for embedded development as long as you follow GNU's public license rules.

DOS-C started in 1988 as an experiment in writing device drivers using C for MS-DOS. As part of the experiment, I wrote block and character device drivers using a C data structure that matched the MS-DOS request packet.

I drew on previous experiences in applying C to embedded and real-time systems. And, I developed new C programming techniques to map assembly-language calls to C and vice versa.

I soon recognized I could write an operating system by using the same techniques. I decided the design should take advantage of C-language features.

Developing the OS would require fewer resources than a traditional assembly-language design. In fact, if I developed it myself, I could take advantage of commercially available tools like Borland C and Turbo Assembler. I could also use MS-DOS debugging tools to prototype and debug program modules under MS-DOS and later integrate them with special start-up code.

It looked like an excellent project. Using C to build an OS wasn't new—Unix proved this concept for large multiuser OSs. But, I could find no documented instances for a smaller PC OS. So, I proceeded to build a minimal OS using the device drivers I had written earlier.

In 1994, when Microsoft announced its intent to drop MS-DOS in favor of Windows 95 (then known as Chicago),

Jim Hall started the FreeDOS project. His plan was to develop an MS-DOS replacement so that when Microsoft stopped producing MS-DOS, FreeDOS could come to the rescue.

You can now find FreeDOS at various FTP and Web sites. There's also a book describing the kernel in greater detail for developers needing to modify the kernel [1].

FreeDOS may be used as a general-purpose DOS replacement that executes standard MS-DOS applications. Additionally, the source for the entire OS is available under the terms of the GNU public license (see sidebar "The Embedded Systems Developer and GPL") on many of these same sites.

What this means for embedded-system developers is that there's a free operating system available. All that's necessary is that the GPL terms are followed.

Although these rules seem stringent, the user needs only to provide a diskette with the source to the kernel on request. The license does not cover the application, so the embedded developer's code can remain proprietary and yet take advantage of a royalty-free environment, thus maximizing profit.

ARCHITECTURE

The DOS-C architecture is layered, similar to what you'll find in other OSs. The only difference between DOS-C and other monolithic OSs is that DOS-C has a decoupling between the upper and lower layers as shown in Figure 1.

To guarantee that DOS-C is MS-DOS compatible, there are two interface layers where full compliance is necessary. The first interface layer—the DOS API—is the set of entry points you expect from a DOS-compatible operating system. These include the traditional INT 20h, 21h, 22h, 23h, 24h, 27h, 28h and 2Fh.

The other interface layer—the device-driver interface—is also well documented and supported by many hardware and software vendors. This layer enables DOS-C to be compatible with MS-DOS to assure proper support of loadable device drivers.

The kernel is written in C wherever possible. However, certain functions are written in assembly language for either efficiency or necessity. These functions typically perform direct machine control, such as stack manipulation and flipping interrupt bits. You can find this code in a number of assembly files along with C-to-assembly interface functions.

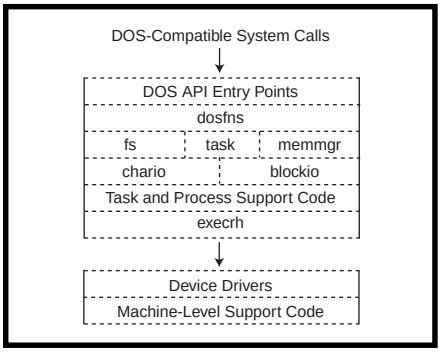


Figure 1—*The FreeDOS/DOS-C kernel architecture is layered, allowing interface behavior to be localized in the dosfns layer for API and the execrh layer for device drivers.*

DOS-C API

A DOS API entry-point handler intercepts any system call to DOS-C and translates it into a C call. I put this as close to the entry point as possible in order to use C as early as possible.

The C call then does some context switching and calls the appropriate internal service function. It's up to the service routine to dispatch the system call through the kernel to the appropriate handler. The dispatcher directs a system call into the kernel.

A large number of calls are directed into the dosfns (DOS functions) layer. This layer works with the lower layers to add the DOS appearance to the system calls.

It does this by combining calls into the various managers in the next lower layers with code to add features to the manager calls that are unique to a DOS interface. For example, this layer contains functions that encompass all rules for handling file I/O that deals with PSP, handles, or character and block I/O functionality.

FILE SYSTEM

The DOS-C design allows for a file manager that is independent from the design of the rest of the OS. Whenever an application requests an operation on a file resident on a block device, this `fs` module performs the operation by working with an internal table called the `fnode` (file node) table.

The `fnode` table is the foundation for the design of the FAT file-system manager. The `fnode` data structure controls all internal file operations and virtualizes the file.

When `fs` performs an operation like create or open, it allocates an entry from the `fnode` table and fills the `fnode` fields. The call returns a number that is the index into the `fnode` table.

From here on out, any function that performs an operation on the file receives this number. The `fnode` table entry that corresponds to this index controls all file parameters.

If you're familiar with DOS internals from the undocumented internals books [2], you know the need to map internal `fnode` to the SFT (System File Table) structure used by DOS. We handle this through the `dosfns` layer function.

The module `dosfns` maps the `fnode` number to an SFT handle, which is part of its "DOS personality" responsibility. At the `fs` level, it does its work in an OS-neutral mode to localize unique features to the personality module.

MEMORY MANAGEMENT

Memory management in DOS-C closely emulates MS-DOS. Because DOS-C is a real-mode operating system,

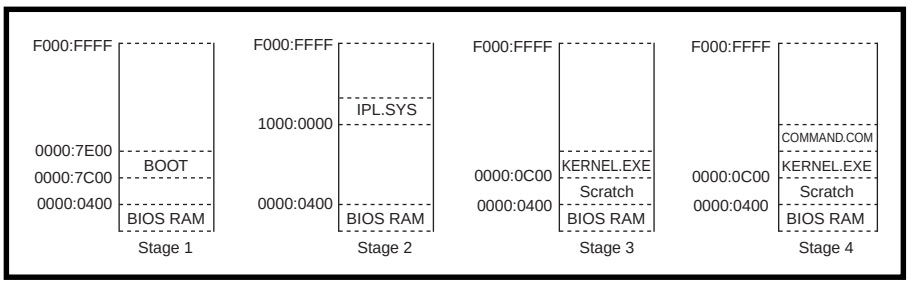


Figure 2—*There are four stages in booting FreeDOS and DOS-C. The first is execution of the boot program from the boot sector. The second is the disposable intermediate program loader, `IPL.SYS`. The third is the kernel itself, which includes device drivers. The final stage is bringing up the user interface—typically, `COMMAND.COM`.*

the DOS-C memory manager is much simpler than what you find in other protected-mode 'x86 OSs.

DOS-C manages memory via a memory-block linked list that uses the same arena header as the one MS-DOS uses. The arena header in DOS-C is known as the MCB (memory control block). As with MS-DOS, this data structure is in front of each allocated and free memory block. DOS-C uses the same `M` and `Z` signatures to identify the block type.

Also, the segment value of the PSP of the program it belongs to, or a 0008h if it belongs to DOS-C, identifies the owner of the block in the same way. DOS-C uses all other entries in an identical manner.

Many applications use this information, and DOS-C provides this same information for sake of compatibility. Unlike `fs`, where MS-DOS personality splits between it and `dosfns`, `memmgr` contains all DOS personality.

TASK MANAGEMENT

Because the DOS-C design emulates MS-DOS, it should be of little surprise that the operating system is not multitasking. As a result, the task manager, `task`, manages a single user task.

Its primary function is to act as a task loader, unlike other operating systems where task loading is incidental to managing the task. Also, there are only two relatively simple executable file types, `.COM` and `.EXE`.

Both executable file types start from a single load entry. It is up to DOS-C to identify the file type.

It does this by examining the first two bytes of the file. If it is an `MZ`, then it is an `.EXE` file and `task` invokes the `.EXE` loader. Otherwise, `task` assumes it to be a `.COM` binary image and invokes the `.COM` loader.

Both loaders initially allocate memory from `memmgr` to place the environment strings in. The differences begin when the actual file loading occurs. The `.COM` loader merely allocates memory and begins loading the file into memory. The `.EXE` loader, on the other hand, computes the size required, allocates memory, and then loads the image.

The Embedded System Programmer and GPL

Mention the GNU public license (GPL) to embedded developers, and reactions range from disapproval to hiding under desks. Developers mumble about having to give away source code. But, that's not necessarily the case.

Written by the Free Software Foundation (www.fsf.org), GPL lets developers retain ownership of source code while placing it and resulting binaries onto publicly accessible sites. Ownership comes through traditional copyright protection, and the license outlines acceptable methods of distribution that protect you from someone stealing your code by using it and pretending it's theirs.

Of course, you should consult your attorney to determine the impact of the license on your project. But, you should do that with any third-party products you use regardless of the source. Any license should be seriously reviewed. GPL is just another license. However, some terms within the license scare potential users.

The license's preamble states its intent to protect your rights to receive the product's source code. It seeks to protect your right to give away the source code if you wish. It also attempts to protect you from the pitfalls of software patents and warranties. In essence, it makes certain that software intended to be openly available stays that way. It then goes on to outline the terms and conditions of the license and finishes with a warranty.

In the terms and conditions, the license states that you can make exact, unmodified copies of software covered by the license, protecting you from any restrictions someone may attempt to place on the software.

It also states that you can modify the code but places some minor restrictions on your changes. Simply put, it lets you make changes but requires that they be clearly marked. This request protects you and the original author from making claims on the changes or the resulting code. This is a big boon for free software, and without this term, software like FreeDOS and Linux couldn't exist.

GPL makes certain a developer's work is protected even if it involves modifications to the free code. In fact, the license goes out of its way to say that changes you make do not necessarily fall under the terms of the license if they "can be reasonably considered independent and separate works in themselves." Clearly, this is important to the embedded developer because it lets you, for example, distribute a compiler under GPL with your special package.

With FreeDOS, your application doesn't fall under GPL unless you make it part of one of the FreeDOS utilities or kernel. You can distribute your application on a FreeDOS bootable diskette and not release source code. GPL emphasizes that "mere aggregation," such as placing your utility on some form of distribution, exempts your utility from using GPL. So, if you supply an installable device driver with your application, your driver code does not necessarily become subject to GPL even though it's part of the OS once it is in memory.

The question: how do you distribute FreeDOS with your application and meet the GPL terms? For commercial apps, give the customer the source code with your application or a written offer valid for three years. You are entitled to a "reasonable" charge for making a copy whether it be for the diskette and shipping or for transferring the source to diskette. The latter may be reasonable if your business is not software distribution and making a copy requires you to spend some staff effort creating the disk or tape.

At first, this might seem unprofessional and make your product seem amateur, but there are ways around it. When asked by commercial users, I usually advise them to either place an offer in their user manual or supply a separate offer along with their documentation. Usually, your customers aren't interested in the OS source code, but there's always a curious few.

When task completes loading an .EXE image, it seeks the relocation offset and does a segment fixup (necessary for the 'x86 family). From this point, both loaders proceed with creating the PSP, cloning the file table, initializing the task's registers, and executing the task if so requested.

DOS-C AND DEVICE DRIVERS

The device-driver interface layer, which is at the bottom of the DOS-C architecture but just above the device drivers, has two handlers—chario and blockio.

Although each handler uses different methods, both are the primary interface between the device drivers and the remainder of the kernel. Both handlers perform all the necessary buffer management for both types of I/O, including line-buffer management.

As with Unix, there are two types of I/O handlers. Their responsibilities are divided between character stream I/O (chario) and random access block I/O (blockio). These I/O handlers are loosely based on the Unix I/O model.

The character I/O type appears as a stream of bytes to the kernel. The kernel either sequentially reads a byte from or writes a byte to a device driver. There are also functions to read a buffer into memory and handle the familiar DOS line-editing functions.

The block I/O interface provides functions to read and write a block of data to a block device—usually a disk. Each block corresponds to a sector and is represented in memory as a data structure in a block cache. When the kernel reads data, blockio reads the sector into a buffer and places it into a least recently used (LRU) chain.

When blockio needs a new buffer, it writes the tail of the list to disk, if needed, and returns that buffer. When blockio completes the data transfer, the buffer goes to the head of the LRU chain, indicating that it's the newest buffer.

Management functions handle these operations for dirty buffer write back, as well as buffer fill. It also performs buffer-management functions like LRU management and flush.

As with the API, the device-driver interface is well documented. Because the interface is designed for an assembly-language system, a special assembly-language function interfaces all device drivers.

This function, execrh, accepts from an I/O handler a packet containing the function number requested of the device driver. It handles the correct sequence of calls to the strategy and interrupt entry points for the device driver and returns the packet to the I/O handler.

The DOS-C device drivers are the bottom layer of the kernel. Like the remainder of the kernel, C is used in these device drivers also. The device drivers perform the necessary device interface between the kernel and the device. DOS-C has the same device drivers as MS-DOS and they perform the same functions.

BOOTING

DOS-C differs from MS-DOS and other DOS clones in its design. Unlike MS-DOS where there are two files loaded into memory to form the operating system, DOS-C uses just one file, KERNEL.EXE.

Additionally, the kernel uses the standard C memory layout of text + data + bss + stack and the 'x86 small model. This setup gives you an interesting degree of flexibility that MS-DOS and other DOS clones don't—you can locate the DOS-C kernel anywhere in memory, including ROM.

BOOTING FROM DISK

Figure 2 depicts the four distinct stages involved in the boot. This approach simplifies the design through modularization and minimizes the effort expended in software maintenance.

For example, a bug that may exist in KERNEL.EXE doesn't require changes in IPL.SYS. As well, no change in size of either IPL.SYS or KERNEL.EXE affects the other.

The booting of DOS-C begins from within the computer's ROM BIOS. After power-up self-test, the BIOS initiates the boot procedure and is at the first stage.

This stage is identical to the way almost any PC-compatible OS starts out. The BIOS sets aside a specific location in memory and the initial

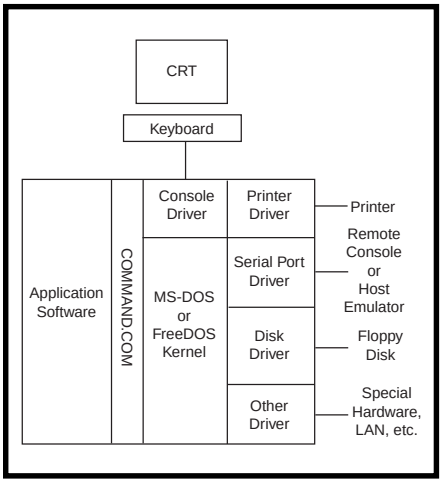


Figure 3—*When using MS-DOS, the typical development environment is relatively straightforward. Both COMMAND.COM and the kernel act as intermediaries between the application and device drivers, virtualizing the devices while managing the device overhead for the application program.*

CS:IP for the boot block. For MS-DOS, IBM placed the boot location at 0000:7C00 to provide 30 KB for loading IO.SYS and MSDOS.SYS.

DOS-C doesn't need boot to load in this location because the following stages are either position independent or relocatable. And, we can judiciously juggle the following stages for any size kernel.

As a neutral choice, I can load stage two, IPL.SYS, into memory at 1000:0000h. There is no magic to this address, and it can be almost anywhere.

I can dynamically locate IPL.SYS anywhere in memory because it is built from the kernel source code using the 'x86 small model. This limits the total size of IPL.SYS to only 64 KB. But by limiting the scope of the IPL, we can easily fit its functionality into this amount of memory.

IPL.SYS loads KERNEL.EXE and is aware of the MS-DOS file systems. However, its lifetime is short and its purpose is simple: load the kernel. To reduce development time, I derive the IPL from the kernel source by reducing the kernel functionality to only what is needed to load the kernel.

This task is completed via #ifdef switches throughout the kernel. These switches cut out unneeded functions such as the API, write, or delete.

We only need functions like open, read, and close. We further reduce IPL.SYS by including only one loader

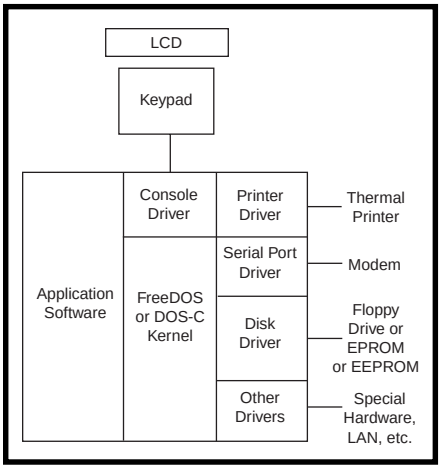


Figure 4—*Similar to the development environment, the FreeDOS/DOS-C kernel performs the device and overhead management. Using device virtualization, the kernel allows the devices to be those typically found in embedded controllers and yet still use a standard API. In the embedded application, the use of* COMMAND.COM *is optional.*

and the device drivers for disk and console.

Stage three starts once the kernel is loaded into memory. The kernel starts at main, like any C program, and begins by initializing itself. This task includes locating all of memory and building arenas and loading the proper addresses into the interrupt vectors. After initialization, the system begins to process CONFIG.SYS.

Finally, the kernel searches for and loads COMMAND.COM into memory and makes this the granddaddy program. If COMMAND.COM returns, the kernel shuts down and you need to reboot. This situation is no different from the case where MS-DOS cannot find COMMAND.COM and issues the error message "Bad or missing command interpreter," only this approach is cleaner.

When COMMAND.COM successfully loads, stage four commences. COMMAND.COM does its own initialization, including creating the initial environment and executing AUTOEXEC.BAT. When this completes, COMMAND.COM issues a prompt and goes into a loop awaiting user input. DOS-C is now loaded and ready to run.

BOOTING FROM ROM

When built with a different start-up module, DOS-C can also run from ROM. The booting method varies slightly because of the nature of the medium. When booting from ROM, the processes of loading the boot block and IPL.SYS naturally disappear, but the memory layout of the kernel may change.

Typically, ROM space is allocated in different areas than RAM and may be noncontiguous. To accommodate this, a commercial locator fixes the memory locations so the ROM image is correct and does not need fixing up.

It also means that when DOS-C runs, its strings and tables may be located in ROM. The ROM start-up code handles this situation by initializing RAM and moving fixed data to RAM.

The ROM startup effectively bypasses stages one and two. When it completes, it invokes main and DOS-C continues with stages three and four. Then, DOS-C is ready and running from ROM.

TYPICAL APPLICATION

When you examine a typical DOS application, such as the one illustrated in Figure 3, you'll notice that the application runs more or less alongside MS-DOS. This action is because MS-DOS actually runs in the same memory space as the application.

Although you might initially consider this situation a hindrance, for the embedded developer, it's a blessing in disguise. For example, the application can handle real-time events external to the operating system.

However, you need to be careful in system design to properly partition functionality. For example, you don't want to replace any device-driver functionality with code unique to your application. Your design is much more robust if you can replace only the device driver when changing platforms.

As an example, let's look at an intelligent printer. In design, it's similar to the typical DOS application. In Figure 4, you immediately see the similarity between them where the application coresides with the kernel.

You also see a significant number of differences in the peripheral devices. One difference is the console. The normal keyboard and display is replaced with an LCD and keypad.

In this example, the application and kernel remain constant, but the device driver changes to handle the different console. The same holds true of other device drivers. It's possible to create the application on a typical DOS platform and move it into the target device by changing only the device driver.

Only certain key functionalities need to be implemented, such as line feed to move characters to the LCD. The same is true of other devices.

TAKING OFF

In Part 2, I'll take a closer look at an intelligent printer similar to the ones used in airports for printing tickets and boarding passes.

These ATB printers are generally quite complex and need many communication modules and interface emulation modules with different types of interface drivers. Also, they sometimes need different print-mechanism drivers for different types of stock that may be used by the printer.

Typically, applications such as these, where the core software remains untouched but many I/O options exist, make good candidates for embedded OSs. Next month, I'll show you how DOS-C simplifies such a design.

Pat Villani has 22 years of industry experience in both hardware and software, most recently developing firmware and real-time kernels. He currently works for a major computer company writing portions of OS kernel and system management tools. You may reach Pat at patv@iop.com.

SOFTWARE

FreeDOS is available at ftp.simtel.net and www.freedos.org. The latest kernel is available at www.iop.com/~patv.

REFERENCES

[1] P. Villani, *FreeDOS Kernel*, R&D Books, Miller-Freeman, New York, NY, 1995.

[2] A. Schulmann et al., *Undocumented DOS, Second Edition*, Addison-Wesley, Reading, MA, 1994.

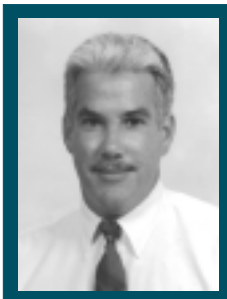
I R S

425 Very Useful
426 Moderately Useful
427 Not Useful

SILICON UPDATE

Tom Cantrell

Fabulous '51s



Even though it's been around for a long time, the 8-bit micro keeps on going and going and going. In fact, new variations keep popping up. Tom lets us in on the smarts some of these new ones possess.

No doubt, you've heard dire predictions like "The 8-bit micro is dead." Of course, the source is usually some expert who happens to make a living marketing 16- and 32-bit chips. Or, it might be an analyst who sells rosy studies to those marketeers so they have some ammo for the next staff meeting.

Well, it's nonsense! Statisticians may quibble over the exact number, but there's no doubt the lowly 8-bit MCU plays a huge—and growing—role in our lives. Yes, 16- and 32-bit chips are doing wonderful things, but their strong growth derives from the law of small numbers, not the death of 8-bitters.

Indeed, today's billions of units per year are just the tip of the iceberg. It's not hard to imagine a cyberfuture where little chips permeate our existence.

MCU CLASSIC

I've worked with various companies hoping to enter the micro business. It's critical, but sometimes difficult, to convey to a commodity IC outfit that micros are a different beast altogether.

For example, a manager recently asked me how designers decide which micro to use. To him, if a micro offered a few key features for a good price and delivery, that was all there was to it.

Of course, that's not at all what happens. The decision is driven by human factors. Designers sometimes choose a new chip for the fun of learning it. Usually, though, they rely on their own stable of chips, tools, and accumulated know-how. Only truly compelling technical issues justify the nontrivial challenge and risk of switching to a completely new architecture.

That's why 20-year-old designs like the '51, 68*xx*, PIC, and Z8 still play a lead role. The issue isn't whether a new architecture is technically superior (not hard with 20 years of hindsight) but whether the old chips are good enough.

MULTISOURCE CODE

Old-timers will remember the heyday of second sourcing, when chip

Compatible with MCS-51 products
8 KB of in-system reprogrammable downloadable flash memory
- SPI serial interface for program downloading
- Endurance: 1000 write/erase cycles

2-KB EEPROM
- Endurance: 100,000 write/erase cycles

4.0–6-V operating range
Fully static operation: DC to 24 MHz
Three-level program memory lock
256 × 8-bit internal RAM
32 programmable I/O lines
Three 16-bit timer/counters
9 interrupt sources
Programmable UART serial channel
SPI serial interface
Low-power idle and power-down modes
Interrupt recovery from powerdown
Programmable watchdog timer
Dual data pointer
Power-off flag

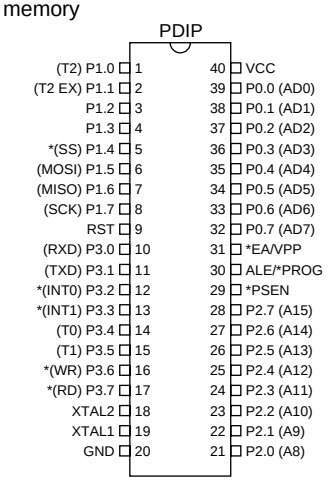


Figure 1—*Atmel, a flash '51 pioneer, offers a new '8252 which incorporates 8-KB flash memory for code and 2-KB EEPROM for data. The chip can be programmed in parallel or serial mode, the latter via a four-pin (select, clock, in, out) SPI interface.*

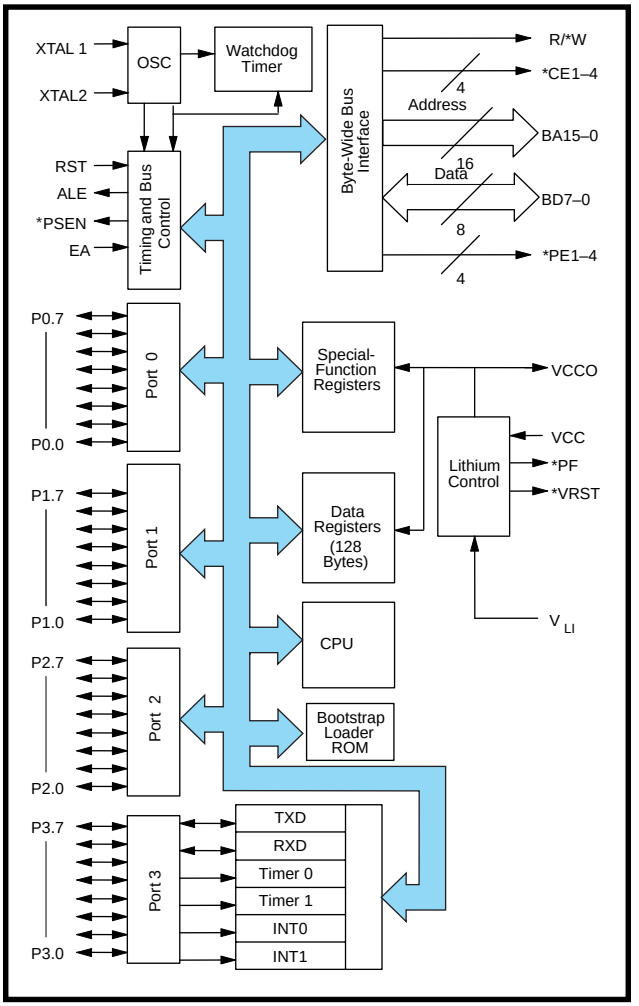


Figure 2—*The Dallas '5001FP supplements a standard '51 core with a dedicated byte-wide code and data bus, freeing port pins to handle peripheral I/O functions.*

suppliers battled for designers' share of mind by encouraging, at least on the surface, wide sourcing. I remember a Motorola press conference where they touted the 68k's big tent with a bunch of licensees that included the likes of Hitachi, Philips, Rockwell, and others.

From a purchasing agent's point of view, it was the golden age—much as it remains for commodity ICs today. You could call multiple distributors and get them bidding against each other by telling each that the other guy was lower until only one was left standing.

That might work for DRAMs, but the micro business needs more fixed investment (for R&D, tools, etc.), and thus profits, than such destructive competition allows. Inexorably, the multisourced micro disappeared and is now practically extinct.

Except for the '51, that is. Of the old 8-bit war-horses, it's unique by virtue of profligate sourcing. It's a veritable people's micro, if you will.

This doesn't mean the laws of economics are repealed. Yes, you can shop around for a cheap, plug-compatible 8051. But don't expect suppliers to cut their own throats. They're about as excited about a lowball '51 deal as the folks at Rockwell or Hitachi are about a 68k inquiry. In other words, don't call me, I'll call you.

What it means is that, across the raft of suppliers, '51 designers have access to a broad spectrum of products—a lineup more extensive than possible from any single supplier. Furthermore, it means the '51 is continually freshened and upgraded, ensuring that the technology gap with newer architectures doesn't get too wide.

Finally, although true plug compatibility is limited to the old baseline chips, plug similarity goes a long way towards ensuring healthy competition. It's a lot easier to switch from one '51 to another with a few different features than to a completely new architecture.

Let's take a look at the latest developments on the '51 front, and you'll see what I mean.

FLASH IN YOUR FUTURE?

Flash micros are starting to take off, and there's a lot of reason to believe they'll become even more popular. Atmel, the undisputed leader in flash-based '51s, is well-positioned for such an eventuality. However, Philips (the largest '51 supplier overall) has recently announced plans to join the fray.

Before checking out the parts, realize that all flash micros aren't created equal. For example, first-generation flash micros are little more than an EPROM/OTP replacement because they use the same parallel programming scheme, 12.5 V_{pp}, and so on.

This is OK, but it really doesn't support the popular idea of streamlined assembly line programming. Sure, it's possible to kludge together some kind of hack for a pseudo-EPROM flash chip (e.g., muxes for all the pins, switchable 12 V_{pp}, etc.), but it isn't clean.

By contrast, Atmel's latest flash '51s, such as the 89S8252 shown in Figure 1, offer the best of both worlds. They still support parallel, high V_{pp} programming for those happy to stick with gang programming.

At the same time, they include a serial programmer that only needs a few pins (SPI interface) and a single power supply (5 V). Holding

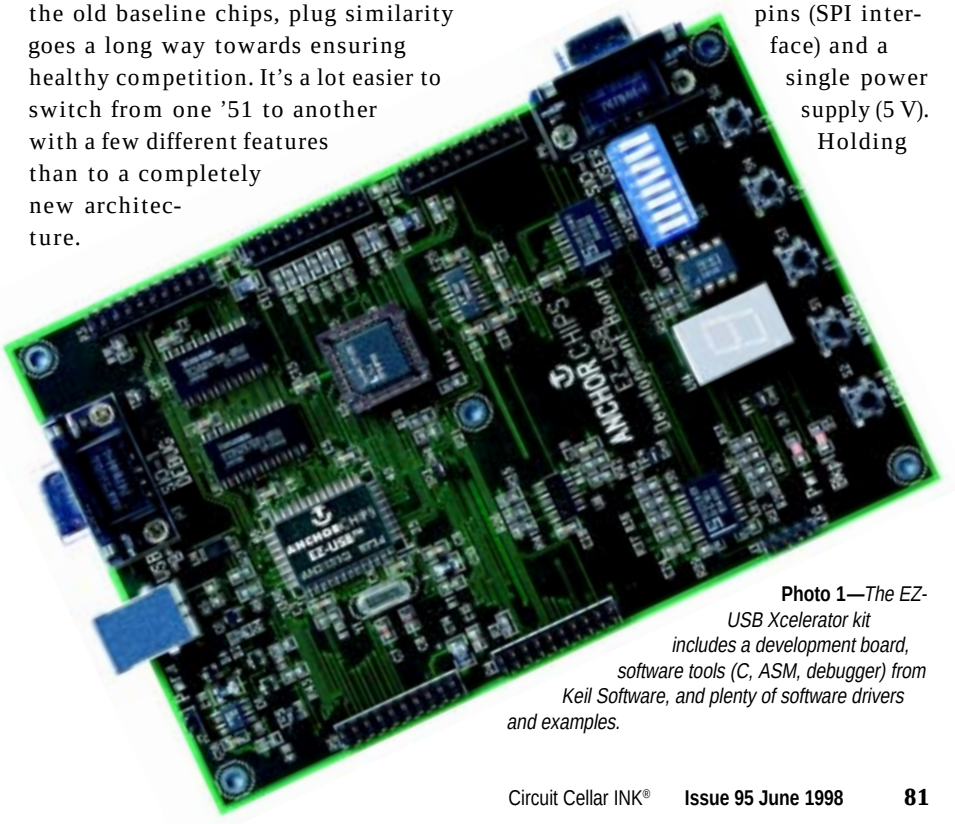


Photo 1—*The EZ-USB Xcelerator kit includes a development board, software tools (C, ASM, debugger) from Keil Software, and plenty of software drivers and examples.*

the part in RESET invokes the serial programming mode, which lets the flash (and extra 2 KB of on-chip data EEPROM) change a single byte at a time (i.e., no need to erase the entire chip).

The new Philips 89C51RX+ dishes up another flavor of flash memory. Like Atmel, Philips supports traditional parallel and in-system programming (ISP) modes.

However, the Philips ISP philosophy is subtly, but profoundly, different from Atmel's. Both parts are ISP in the sense of easy post-PCB assembly (re)programming. Philips goes a step further with self-programming that enables the chip to dynamically change its own code.

Here's how it works. A Boot ROM overlays the top 1 KB of code space, which contains routines to erase a block, program byte, verify byte, and so on. There's also a Boot Vector that defines what happens after RESET depending on the state of a flash-status byte. Execution at the Boot Vector can also be forced by external pin setting.

The Boot Vector in turn points to a user-written Loader in one block of the flash memory that's responsible for erasing and programming the other blocks. The Loader, taking advantage of calls to the Boot ROM, can self-program the chip using any technique. For example, using the 89C51RX+, you can remotely (re)program it via modem a built-in feature of your design.

Note the default shipping configuration includes a prewritten Loader to accept commands and data via the on-chip UART. That means a factory-fresh part can be inserted in a board and programmed serially at a later time.

One key difference: in serial programming mode, the Atmel part only needs 5 V, whereas the Philips uses 12 V. For compatibility with existing programmers, both require high voltage for parallel programming.

Otherwise, Atmel and Philips parts have a lot of other neat features and upgrades. They have dual data pointers, fancier peripherals, watchdog timers, faster clocks, and more. Designers are sure to benefit from the battle for their flash '51 favor.

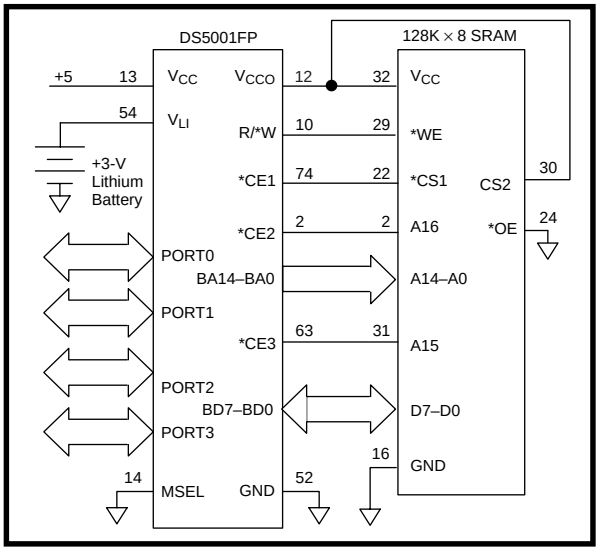


Figure 3—*With built-in voltage supervisor, battery switching, and bootstrap ROM, the '5001FP is especially well-suited for SRAM connection. The chip completely manages the SRAM power and control signal generation, maintaining data integrity without needing any glue logic.*

'51 RAM CRAM

Dallas Semi is known for its fast (4 clock/instruction at 33 MHz) 80C320, but it has other tricks up its sleeve, too.

The DS5001FP depicted in Figure 2 is an interesting alternative that combines a vanilla '51 with a dedicated byte-wide memory bus, so adding external memory doesn't consume I/O port pins. Four chip enables (*CE1–4) support various combinations of 32-KB blocks for a total of up to 128-KB memory. Likewise, four peripheral enables (*PE1–4) handle data accesses.

While any type of memory can be used, the DS5001FP is especially well-suited for SRAMs (see Figure 3). An on-chip voltage supervisor not only generates power-fail detect (*PF) and reset (*VRST) but also controls the previously mentioned byte-wide bus enables to prevent spurious writes.

Like Philips' flash parts, the '5001 includes a bootstrap loading feature that downloads the SRAM via serial port after first-time powerup. Once initialized and verified, the bootstrap loader leaves the stage, and power-management logic ensures that the SRAM remains valid (up to 10 years, depending on the battery). The SRAM need never again be touched (at least until the battery runs dry), but it can be rebootstrapped as often as desired by driving the *PROG input.

Besides the '5001 chip, Dallas also offers the DS2251T module, which combines the MCU with a 32–128 KB of SRAM, real-time clock, and battery.

"OH SAY CAN USB?"

That was the title of my article about USB in *INK* 74. There, I pointed out that the USB concept was grand, especially compared to the ludicrous rat's nest of parallel, serial, mouse, game pad, and so forth lurking behind PCs.

However, the downside of vaunted PC compatibility is inertia. I correctly reckoned USB wouldn't take off until the arrival of a critical mass of built-in driver software with Win98.

Despite the fact there's a zillion USB-capable PCs out there, the vast majority of USB ports are gathering dust.

The issue is complicated by the fact that Win98 seems to be slipping (no real surprise), with the rather interesting kicker that an awful lot of lawyers are involved. As well, the quality and quantity of USB support in Win98 remains to be seen. I overheard one Microsoft engineer saying they've "got more important networks to deal with."

The transition to Win98 and/or USB may not be as quick and clean as anticipated. At this point, USB is scratching for each peripheral design and inch of shelf space one torturous step at a time.

Ultimately, however, I believe the combination of automatic-installed base (i.e., USB on every motherboard) and

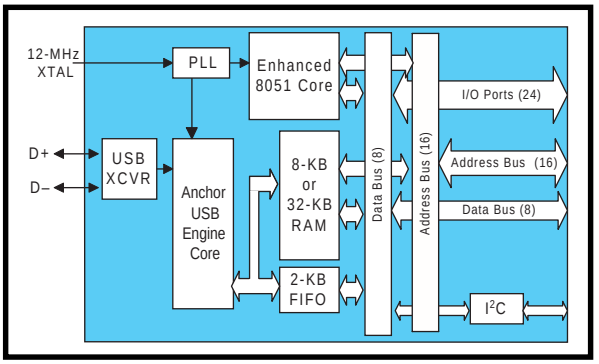


Figure 4—*When USB finally starts rolling, look for '51-based derivatives like the Anchor Chips AN21xx to get onboard.*

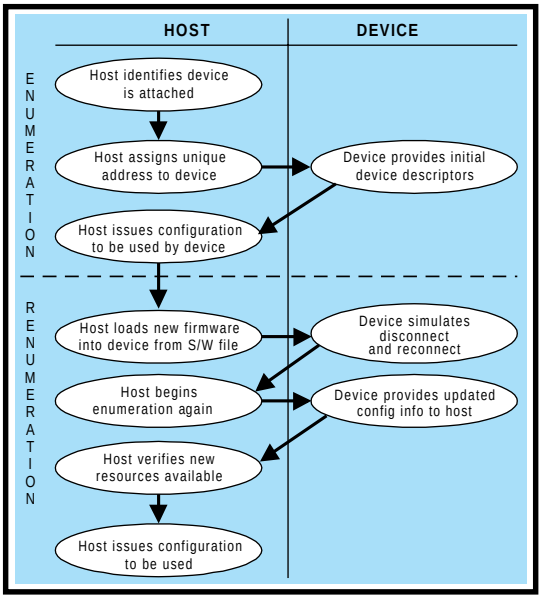


Figure 5—*Booting up the AN21xx is a two-step process. First, there's a normal USB enumeration cycle, and application-specific '51 code is downloaded into the on-chip RAM. Next, renumeration reintroduces the PC to the now-personalized chip.*

simple sanity will prevail. And when it does, the market for USB chips—no shortage of them '51 based—will explode.

The AN21*xx* from Anchor Chips is a good example of the way new blood keeps the '51 young at heart. It combines the best aspects of other derivatives in a unique combination (see Figure 4).

Anchor Chips starts with a hot-rodded four-clock/instruction core. A PLL fed by an external 12-MHz crystal generates the 24-MHz CPU clock and 12-/1.5-Mbps (high/low data rate) USB clock.

The USB engine is powerful, with a default device descriptor that lets it boot as a generic device. Logic handles the low-level transfer details, accomplishing an entire multipacket transaction in hardware. Thus, according to Anchor, managing the network consumes only 10% of CPU bandwidth, leaving the rest for application-specific processing.

Although a ROM version is available, Anchor's claim to fame is exploiting on-chip RAM—and there's a lot of it (up to 32 KB). There's also an extra 2 KB of USB FIFO which, using a ping-pong approach, supports full 1024-byte isochronous (i.e., time-sensitive stuff like audio, video, and high-speed data acquisition) packet transfers in less than half of a single 1-ms USB frame.

The chip comes in low-cost 44- and 80-pin PQFP packages (Photo 1). Larger versions support external expansion by bringing the address and data bus (non-muxed, don't need a latch) off chip.

Another variant brings out only the data bus for connection to an external FIFO, providing dedicated read and write pins (*FRD, *FWR) to drive it. Otherwise, every version of the chip includes an I^2C port and 24 I/O lines configurable to function as PIO or the usual alternates (e.g., UARTs, timers, interrupts, etc.).

Of course, a RAM-based design raises the obvious question of how to install code after powerup? One approach, like an FPGA, relies on accessing an external boot memory (either I^2C or parallel) to load the on-chip RAM after powerup.

Anchor calls its more novel approach "renumeration." The USB spec requires bus enumeration when a peripheral disconnects or connects (i.e., hot plug).

At powerup, the Anchor chip automatically enumerates as a default USB device. Then, as shown in Figure 5, a host PC downloads the '51 code via the USB connection into the on-chip RAM.

Once loaded, the chip simulates a disconnect/reconnect, causing the host PC to renumerate. This time, Anchor's chip responds to interrogation with the just downloaded device description.

This feature provides a neat chameleon-like capability. For instance, take a piece of USB-based data-acquisition gear with a variety of functions and modes. Depending on the task, an Anchor-based solution can switch hats by redefining the number of endpoints (up to 16), type of connection (e.g., isochronous, bulk, interrupt, control), and FIFO allocation appropriately.

AND THE CHIP PLAYED ON

The 8-bit micro was initially pronounced dead 20 years ago during the height of the original showdown between the 8086 and 68k.

The marketing guy giving the premature eulogy moved on to a small startup. The company did very well over the years, making him quite wealthy along the way.

If you haven't already guessed, that company's success was based in no small part on a popular line of 8-bit chips! You can bet I always ask him if the 8-bit market is dead yet every time I see him.

MCUs may come and go, but '51s are here forever.

Tom Cantrell has been working on chip, board, and systems design and marketing in Silicon Valley for more than ten years. You may reach him by E-mail at tom.cantrell@circuitcellar.com, by telephone at (510) 657-0264, or by fax at (510) 657-5441.

SOURCES

89S8252
Atmel Corp.
2125 O'Nel Dr.
San Jose, CA 95131
(408) 441-0311
Fax: (408) 436-4300
www.atmel.com

89C51RX+
Philips Semiconductors
811 E. Arques Ave.
Sunnyvale, CA 94088-3409
(408) 991-5207
Fax: (408) 991-3773
www-us2.semiconductors.philips.com/microcontrol

DS5001FP, DS2251T
Dallas Semiconductor
4401 S. Beltwood Pkwy.
Dallas, TX 75244-3292
(972) 371-4448
Fax: (972) 371-3715
www.dalsemi.com

AN21*xx*
Anchor Chips, Inc.
12396 World Trade Dr., MS 212
San Diego, CA 92128
(619) 613-7900
Fax: (619) 676-6896
www.anchorchips.com

I R S

428 Very Useful
429 Moderately Useful
430 Not Useful

PRIORITY INTERRUPT

Darwin Couldn't Have Done Better

Evolution is a good thing. I hate to think how boring everything would become if there were no improvements in ideas or products. Just imagine—if nobody wanted to go faster and climb higher, a *mouse* might still be just a rodent.

Today, we are in the information age. I say that rather than calling it the computer age because the wealth and speed of information gathering is the greatest evolution. A computer, like the automobile, may be the physical component that triggers cultural growth and evolutionary change, but it takes an application with universal acceptance to create a true revolution.

One of the critical areas being affected by the rapid evolution of the Internet is print publishing. If you listen to marketing people from MIS and electronic-media organizations, they'll tell you that there's no reason to buy a printed book or magazine ever again. Of course, a similar claim was made shortly after the invention of the television. And, we all remember how the VCR was supposed to spell the demise of movie theaters. In short, be careful of predictions.

This information revolution is a phenomenal bandwagon that can't be ignored. Because the degree of commitment is so critical, however, it's even more important to carefully examine all the facts. The first reality is that, much to the chagrin of electronic publishers and electronic communication companies, print media isn't going away. While it's theoretically possible to publish everything in an electronic-only format, the people who actually buy subscriptions seem to have a different opinion about its necessity—basically, "don't rush me!"

In my experience, the pressure isn't to simply replace print magazines with their electronic online equivalent. Instead, there seems to be a marriage of accommodation. This marriage of technologies doesn't necessarily dictate an equal distribution of either resources or subject content, however. Of course, certain tasks should be logically allocated to the more efficient partner. Until bandwidth isn't the limiting issue, printed media is still the better place for high-resolution photographs and complicated schematics. Long program listings, articles too short to print, archived articles, and product datasheets are more suited to the Web.

A realization of this new division of resources should be apparent this month. Magazines traditionally offer Reader Service. That's the official name for the bingo card next to the advertiser index page that you fill out and mail in for more product information. The bad news is that I've eliminated bingo cards in *INK*! The good news: we now have the advertisers and their URLs set up on our Web site.

Traditionally, the number of bingo-card responses was a measure advertisers used to gauge the value of their ad placements. But, using such an inadequate and inefficient medium to judge the quality of a highly motivated audience like *INK*'s certainly begs the question. Like most of our readers, when I see something interesting in an ad these days, I don't go check off the bingo card and sit around three weeks waiting for something to arrive in my mailbox. I immediately go to the Web site and download the datasheet—instant action and instant gratification.

I don't have a crystal ball and I can't predict the eventual shakeout between print and electronic media. What I can predict is purely personal. *Circuit Cellar INK* is in the process of greatly expanding its Web site to better serve both readers and advertisers. Like all good things, the process will take some time, but our Web site is going to contain more of the projects and relevant application materials that have always distinguished Circuit Cellar from the crowd.

It may no longer be enough to simply say that we publish a reputable technical reference magazine. People with hands-on responsibility, like *INK* readers, always have a thirst for more application materials. The Internet offers a convenient medium to expand technical product support. Establishing and maintaining a company Web site shouldn't be viewed merely as a competitive accommodation to rolling evolution. Properly utilized, a Web site can be the vehicle that is that evolution.

steve.ciarcia@circuitcellar.com