CIRCUIT CELLAR INK®

THE COMPUTER APPLICATIONS JOURNAL

#66 JANUARY 1996

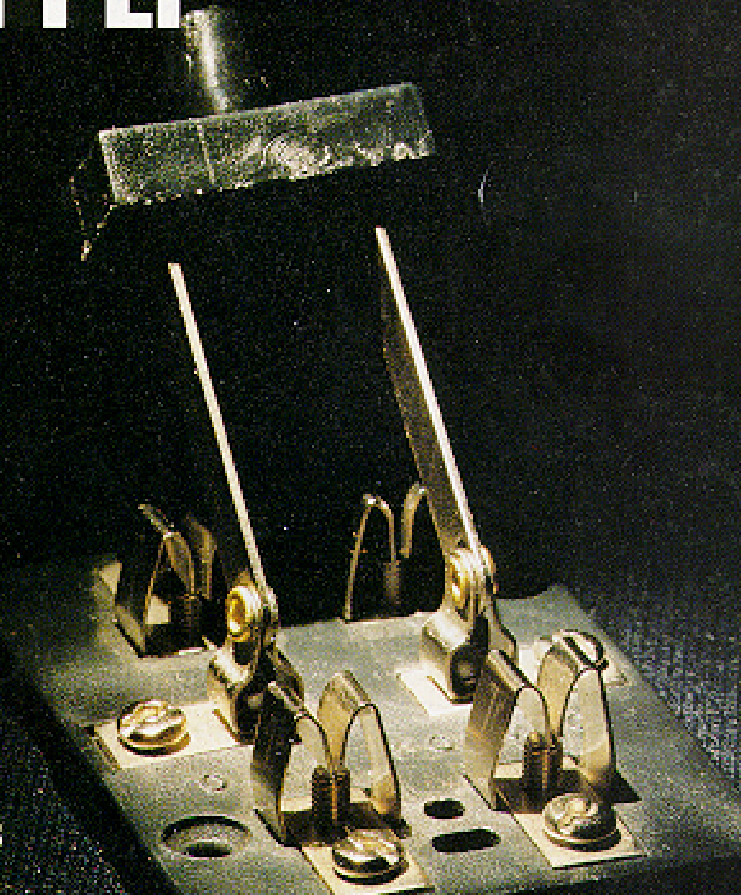
POWER SUPPLY DESIGN

Micro-controlled High-voltage Power Supply

Push-Pull Switching Regulator Design

Motor Energy Management

Lancaster Does Graphics Transforms



$3.95 U.S.
$4.95 Canada

TASK MANAGER

Bear Skins and Stone Knives

If you caught Steve's editorial in the November issue ("They Still Flip Hamburgers, Don't They?"), you'll recall that he wondered how far beyond the dark ages the food industry has come in their use of advanced technology. As a letter from a reader this month points out, if you look in the right places, you'll find that they've indeed come quite far.

However, sometimes I wonder the same question about power supply design. I mean, how many ways can you wire a rectifier bridge or wind a transformer? Take a look around, though, and you'll find smaller, lighter, and cheaper supplies available today capable of supplying voltages and currents well beyond their 10-year-old counterparts.

Keep in mind, too, the need for cleaner regulated power for today's more sensitive analog and digital electronics, plus stricter FCC regulations restricting emitted noise. Taken all together, power supply design has progressed quite nicely.

Another area of power supply design that didn't exist a few years ago was the use of a dedicated processor inside the supply to provide supervision, control, and feedback. Our first feature article this month details the design of one such supply.

Though engineers usually opt for off-the-shelf power supplies for final production, it's not as difficult or expensive as you might think to design the supply right onto the main board. Ed Schram's article looks at some of the issues involved.

While not falling strictly into the realm of power supply design, optimizing AC motor performance to minimize wasted energy still deals with the control of high voltages and currents. Our third feature article discusses a dedicated chip that helps make such power management easier and brings us from the prehistoric age, through the dark ages, right into the green age.

Next, we showcase one of our past Design Contest winners: the Micro bRISC PIC programmer. Follow Ken through the evolution of his design from primitive to sleek final result.

A few months ago, we had an article that introduced you to Digital's Alpha processor. Following up on that introduction, we next look at some software tools available for it.

In our final feature, Don Lancaster discusses some unconventional graphics transforms that can add spice to your graphics presentations.

In this month's columns, Ed takes a break from his journey to spend some time on other projects. His first: a video text display unit for the HCS II. Jeff explores a new kind of nonvolatile memory. Lastly, Tom looks at a new '486 chip from National Semiconductor.

Ken

editor@circellar.com

CIRCUIT CELLAR INK®

THE COMPUTER APPLICATIONS JOURNAL

FOUNDER/EDITORIAL DIRECTOR
Steve Ciarcia

EDITOR-IN-CHIEF
Ken Davidson

SENIOR TECHNICAL EDITOR
Janice Marinelli

TECHNICAL EDITOR
Carole Boster

ENGINEERING STAFF
Jeff Bachiochi & Ed Nisley

WEST COAST EDITOR
Tom Cantrell

CONTRIBUTING EDITORS
Rick Lehrbaum
Russ Reiss

NEW PRODUCTS EDITOR
Harv Weiner

ART DIRECTOR
Lisa Ferry

PRODUCTION STAFF
John Gorsky
James Soussounis

CONTRIBUTORS:
Jon Elson
Tim McDonough
Frank Kuechmann
Pellervo Kaskinen

PUBLISHER
Daniel Rodrigues

PUBLISHER'S ASSISTANT
Sue Hodge

CIRCULATION MANAGER
Rose Mansella

CIRCULATION ASSISTANT
Barbara Maleski

CIRCULATION CONSULTANT
Gregory Spitzfaden

BUSINESS MANAGER
Jeannette Walters

ADVERTISING COORDINATOR
Dan Gorsky

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

Cover photography by Barbara Swenson
PRINTED IN THE UNITED STATES

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

HAJAR ASSOCIATES **NATIONAL ADVERTISING REPRESENTATIVES**

NORTHEAST & MID-ATLANTIC
Barbara Best
(908) 741-7744
Fax: (908) 741-6823

SOUTHEAST
Christa Collins
(305) 966-3939
Fax: (305) 985-8457

MIDWEST
Nanette Traetow
(708) 357-0010
Fax: (708) 357-0452

WEST COAST
Barbara Jones & Shelley Rainey
(714) 540-3554
Fax: (714) 540-7103

Circuit Cellar BBS—24 Hrs. 300/1200/2400/9600/14.4k bps, 8 bits, no parity, 1 stop bit, (860) 871-1988; 2400/9600 bps Courier HST, (860) 871-0549. World Wide Web: http://www.circellar.com/

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

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

INSIDE ISSUE 66

READER I/O

THE EYE: A READER'S RETORT

I'm writing in response to Homer Tilton's "Perpetuating Color Myths" (*INK* 64). According to my references, there are three types of cones in the fovea, roughly equivalent to red, green, and blue. There is evidence that these signals are transformed into separate color and luminance channels early in the visual system, with some of the processing occurring in the retina. This process parallels what happens in an NTSC TV camera.

You might want to check the following references:

Schnapf, J.L., and D.A. Baylor. "How Photoreceptor Cells Respond to Light." Scientific ***American. 256: 40-47, 1987.***

Livingstone, M.S. and D.H. Hubel. "Anatomy and Physiology of a Color System in the Primate Visual Cortex." *Journal **of Neuroscience. 4: 309356, 1984.***

Hubel, David H. ***Eye, Bruin, and Vision.*** Scientific America Library, ***1988.***

Charles Rosenberg
Cambridge, MA

CONTROL THAT BURGER!

Since I have been an embedded systems engineer in the food service industry for about 15 years, Steve's November editorial was of considerable interest.

Contrary to your supposition, embedded controls are common in food service equipment. Perhaps you just missed our booth and the booths of our competitors because we all use embedded controls.

However, you're right that the acceptance of embedded controls has been slow, primarily because of cost. Fast food restaurants have extremely low profit margins. In simple cooking applications, it's impossible for an embedded controller to compete with a $10 electromechanical thermostat and $5 timer.

The performance of the two different systems is a world apart, but for some applications, the cheap system is more than adequate. Chains such as McDonald's, KFC, and Wendy's require more sophistication because they cook more than one item on a single piece of equipment.

Embedded controls have also not caught on because of the ambient conditions. High temperatures, high condensing humidity, pressure washing of equipment, lots of EM1 from load switching, contamination and corrosion from airborne shortening vapor and breading dust are a few of the problems.

As well, users are unsophisticated-mostly high school students-and the turnover is high. Operation must be dumbed down to the lowest common denominator. This labor base also results in both deliberate and accidental abuse of equipment and controls.

All these problems can be solved, but the solutions cost money. The restaurant manager is much more concerned with customer satisfaction and health and fire code inspectors. The owner wants equipment to be safe, reliable, and cheap-in that order.

We and our competitors have both solved many integration problems and embedded controls are common. In terms of technology, we're in the median of embedded applications. We use everything from PICs to HC 11 s and we code in assembler and C.

Finally, I've been a subscriber for years and really enjoy your magazine. It has a higher useful information density than any of the trade magazines that I get, and I get a lot. Keep up the good work.

Doug Burkett
Eaton, OH

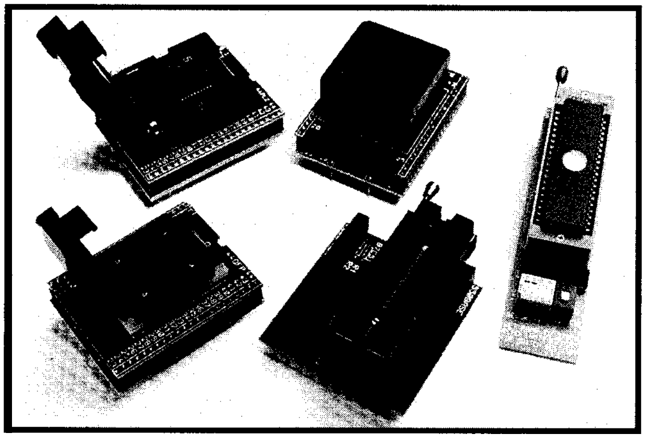
Contacting Circuit Cellar

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

Mail: Letters to the Editor may be sent to: Editor, Circuit Cellar INK, 4 Park St., Vernon, CT 06066.
Phone: Direct all subscription inquiries to (800) 269-6301.
Contact our editorial offices at (860) 875-2199.
Fax: All faxes may be sent to (860) 872-2204.
BBS: All of our editors and regular authors frequent the Circuit Cellar BBS and are available to answer questions. Call (860) 871-1988 with your modem (300–14.4k bps, 8N1).
Internet: Letters to the editor may be sent to **editor@circellar.**corn. Send new subscription orders, renewals, and address changes to **subscribe@circellar.com.** Be sure to include your complete mailing address and return E-mail address in all correspondence. Author E-mail addresses (when available) may be found at the end of each article. For more information, send E-mail to **info@circellar.com.**
WWW: Point your browser to **http://www.circellar.com/.**
FTP: Files are available at **ftp://ftp.circellar.com/pub/circellar/.**

NEW PRODUCT NEWS

Edited by Harv Weiner



51 XA-G3 ADAPTERS

Logical Systems introduces a family of adapters to aid the transition of Philips 51XA-G3 devices into 805 1 -FC products. The 5 1XA-G3 is an extended Architecture derivative CMOS 16-bit microcontroller that provides upward compatibility for 8-bit 80C-51 users.

51XA-G3 adapters are available for several purposes. Some program 5 1XA-G3 chips on an 8051-FC programmer. Others are package converters that connect PLCC or QFP chips to the 5 1 XA-G3 footprint, primarily for device programming. Development and prototyping adapters include ones that connect 5 1XA-G3 DIP chips to PLCC footprints and ones that connect 51XA-G3 PLCC chips to 8051-FC footprints for device transition. The adapters are available with auto-eject or lidded ZIF sockets.

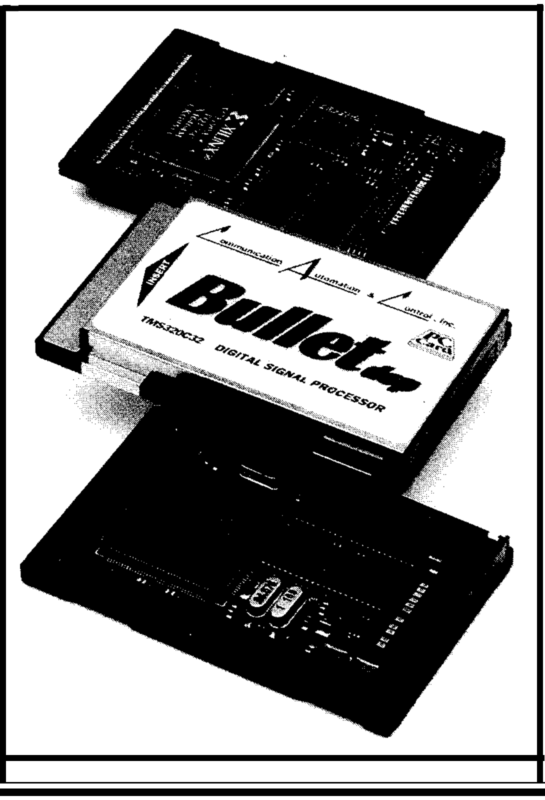
Pricing is in the range of $65-$179.

Logical Systems Corp.
P.O. Box 6184
Syracuse, NY 13217-6184
(315) 478-0722
Fax: (315) 479-6753

#500

50-MFLOPS DSP CARD

Communication Automation & Control introduces BULLETdsp, a portable, 50-MFLOPS, Type-3 PCMCIA interface card that targets high-performance audio-signal-processing applications. Based on Texas Instruments' TMS320C32 floating-point digital signal processor, BULLETdsp features two stereo CD-quality audio input and output channels and provides up to 1 MB of SRAM, 4 MB of DRAM, and 5 12 KB of flash memory. The chip also features a pair of 256 x 32-bit, single-cycle, dual-access RAM blocks; two memory-mapped 32-bit timers; an externally accessible serial port; a 64 x 32-bit program cache; and two DMA channels that are used by the onboard CODEC.



BULLETdsp's two audio I/O channels are based on Crystal Semiconductor's 16-bit CS4231 stereo multimedia CODEC which provides line-in, line-out (headphones], and phantom-powered mic-in channels. The CODEC's sigma-delta A/D input portion features 64x oversampling and linear phase, digital antialiasing filters. The D/A output portion features a sigma-delta modulator with 16-bit resolution and 8x interpolation filters that enable it to track the sampling rate automatically.

Both the host and DSP can control the CODEC's programmable sample rate (5.51–48 kHz), input gain (0–22.5 dB), and output attenuation (0 to -94.5 dB). The converter's linear (8- and 16-bit) Alaw, Flaw, and AD-PCM formats are also software selectable. A 15-pin receptacle accepts several dongles for audio connections and direct interfacing to the DSP's serial port. BULLETdsp consumes a maximum of 1.82 W at 5 V and 0.26 W in power-down mode.

Software development tools include Texas Instruments' TMS320C32 C Compiler Development System ($1500) and Assembler Development System ($500). BULLETdsp's base price of $895 includes 256 KB of SRAM.

Communication Automation &
Control, Inc.
1642 Union Blvd., Ste. 200
Allentown, PA 18103
(610) 776-6669
Fax: (610) 770-l 232

#501

NEW PRODUCT NEWS

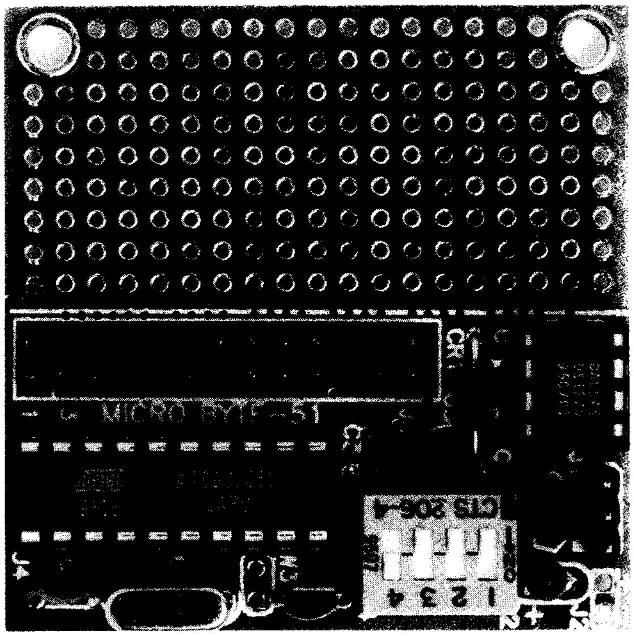
MINIATURE SINGLE-BOARD COMPUTER

Micro Byte-51 is a miniature single-board computer based on the popular 805 1 family of microcontrollers. The family device installed on the board is the 89C205 1, a fully static CPU with clock speeds up to 24 MHz. Micro Byte-51 is available with clock speeds of 12 or 20 MHz.

The 89C2051 features an on-chip, 2-KB, electrically erasable flash memory for program storage. New application software can be programmed in-system while the 89C205 1 remains installed, using the company's PB-5 1/11 programming board.

Micro Byte-51 measures only 2.00" x 2.00". It includes the DS1833 5-V EconoReset chip, the DS1275 RS-232 transceiver, and a prototyping area. The DS1833 generates a reset signal on power-up. The DS1275 is a line-powered transceiver which provides RS-232 levels for the MPU's serial port. The prototyping area takes up virtually half of the total board footprint. It can be separated from the board's main circuitry, reducing the board size to 1.00" x 2.00".

The onboard connectors make designing with the Micro Byte-51 relatively straightforward. One 26-pin header provides access to all CPU resource lines. It can be tied to user circuitry or to the PB-51/11 during programming. A three-pin header on the board brings out the RS-232 buffered serial lines. This header can be connected to the serial port on a host PC. An onboard regulator makes it possible to use Micro Byte-51 with an unregulated supply source of 9 VDC.



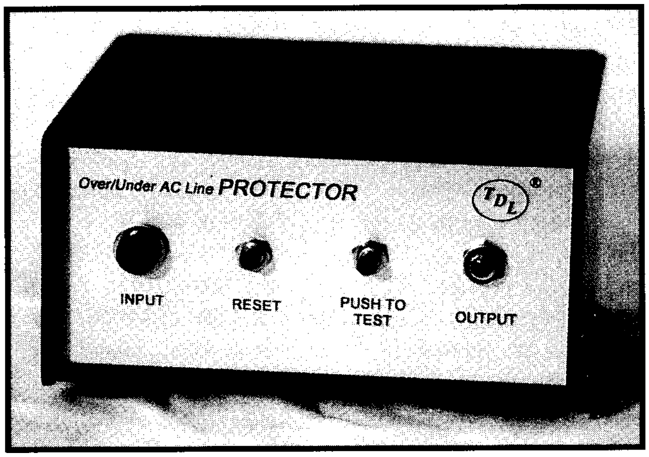
Micro Byte-5 1 sells for $39 for either the 12- or 20-MHz version. The PB-51/11 Programmer sells for $99.

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

#502

LINE-LOAD EMULATOR

TDL Electronics has introduced the **Protector,** a small, self-contained unit that disconnects a load such as a fax or PC if the line voltage goes below 100 V or above 130 V. A window comparator (using a Maxim ICL7665) and supporting TTL logic energize a solid-state AC relay, to keep the load connected as long as the line voltage stays within the window. MOVs provide transient protection.



When the push-to-test switch on the front panel is pressed, a built-in test circuit simulates an out-of-window line voltage. A reset switch restores normal operation. Front-panel lights indicate when input power is available and when power is applied to the load. The Protector is housed in an aluminum cabinet that measures 6" x 5" x 3". The unit is supplied with a 3-A fuse, but the solid-state relay is rated at 10 A.

The Protector is available as a printed circuit board (**$8.50**), a kit consisting of the circuit board and all board-mounted parts (**$39.95**), and assembled and tested ($69.95).

TDL Electronics Inc.
P.O. Box 2015
Las Cruces, NM 88012
(505) 382-8175
Fax: (505) 382-8810

#503

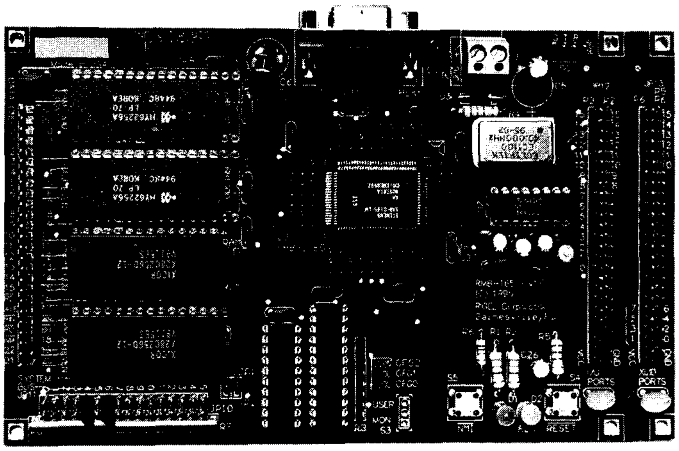
NEW PRODUCT NEWS

EMBEDDED CONTROLLER

Rigel introduces the **R-51 JX,** an 8-bit embedded controller board designed for the Intel 80C251, Dallas 80C320, or Siemens 50*X* series of processors. The board is optimized for the high-performance features found on these chips. It includes a can oscillator for various clock speeds, enhanced VCC and GND shielding to lower noise interference, and two serial ports for use with the DS80C320.

The R-5 **1** JX has **12** I/O bits available on terminal blocks and 64 KB of memory. Operating speed varies depending on the processor, but ranges from 11.0592 MHz (for the 80C25 1) to 40 MHz [for the SAB C501}. The system bus is available on a header in which the processor's address and data lines are demultiplexed. The bus interfaces the R-5 1 JX to external memory-mapped I/O devices. The user may decode the address along with the *XIOSEL, *RD, and *W R signals to select memory-mapped peripherals.

The R-5 1 JX comes with Rigel's integrated development environment READS (Rigel's Embedded Applications Development System). READS is a DOS-based integrated development system. Connected to an IBM PC or compatible host, READS allows one to write, assemble, download, debug, and run application software in the MCS-5 1 language. READS contains an editor, cross-assembler, and host-to-board communications in a user-friendly, menu-driven environment.



Tutorial source code quickly familiarizes the user with the R-5 1 JX. Example software enables the user to experiment with the board and READS. Examples illustrate the features of the 805 **1** and 80C25 1, specifically digital and serial I/Os, timers, counters and interrupt logic. The package also comes with a library of routines, including code for 7-segment displays, LCDs, keypads, stepper motors, and DC motor control.

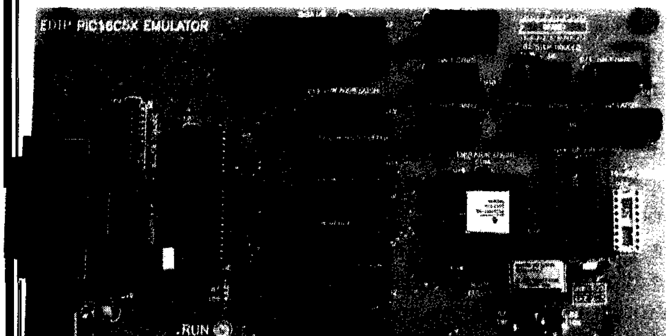
The R-5 1 JX includes 32-KB EPROM, 32-KB RAM, READS software, user's guides for board and software, BSO/Tasking's evaluation tools, example programs, software tutorial and circuit diagrams. The complete system price is $120-140.

Rigel Corp.
P.O. Box 90040
Gainesville, FL 32607
(904) 373-4629

#504

IN-CIRCUIT EMULATOR

E D Technical Publications' PICulator is a low-cost in-circuit PIC-16C5*x* emulation system in kit form. The Windows-based PICulator software offers unlimited breakpoints and on-the-fly modification of PIC program and data memory. Other emulation features include full display of stack, W, program-counter, and all other internal PIC registers. The user can modify each individual register location via the Windows driver program. The system also has single-step operation.



PICulator supports the PIC16C54, '55, '56, '57, and '58 via the standard PC parallel port. All PICulator components, except the emulation engine, are standard off-the-shelf devices, which means easy repair for users in harsh development environments.

PICulator sells as a kit for **$199** and comes with Windows supervisory software, power supply, and free technical support.

E D Technical Publications
P.O. Box 541222
Merritt Island, FL 32954
(800) **499-EDTP**
Fax: (407) 454-9905
edtp@ddi.digital.net

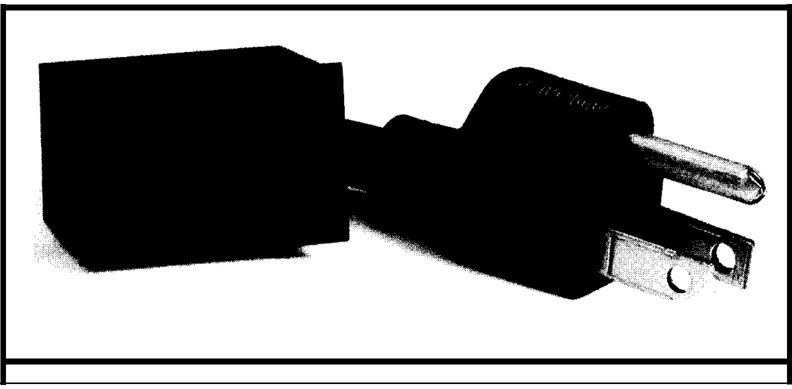
#505

NEW PRODUCT NEWS

SUBMINIATURE AC-LINE FREQUENCY MONITOR

A self-powered, self-contained AC-line-frequency meter is available from Datel. The **DMS-20PC-FM** connects to the line under test and is fully operational with no additional components or auxiliary power required. The operating input voltage ranges from 8.5 to 140 VAC. There are two input-frequency versions available: one for 50- or 60-Hz operation and the other for 400-Hz operation.

The DMS-20PC-FM is housed in a rugged, epoxy-encapsulated, moisture- and vibration-resistant plastic package. The entire unit measures only 1.38" x 0.88". Depth behind the panel, including all wiring, is 1.0". The three-digit, 0.4" high, bright red LED display is easy to read from as far away as 20'. The package is designed so that the meter can easily be mounted on circuit boards. An optional bezel assembly, featuring secure screw attachments, is available for panel-mount applications, including those in harsh industrial environments.



An ultrastable, quartz-crystal-controlled embedded microprocessor guarantees accuracy to ±0.1 Hz (50/60-Hz model) or ±1.0 Hz (400-Hz model) over the entire operating temperature range of -25 to +60°C.

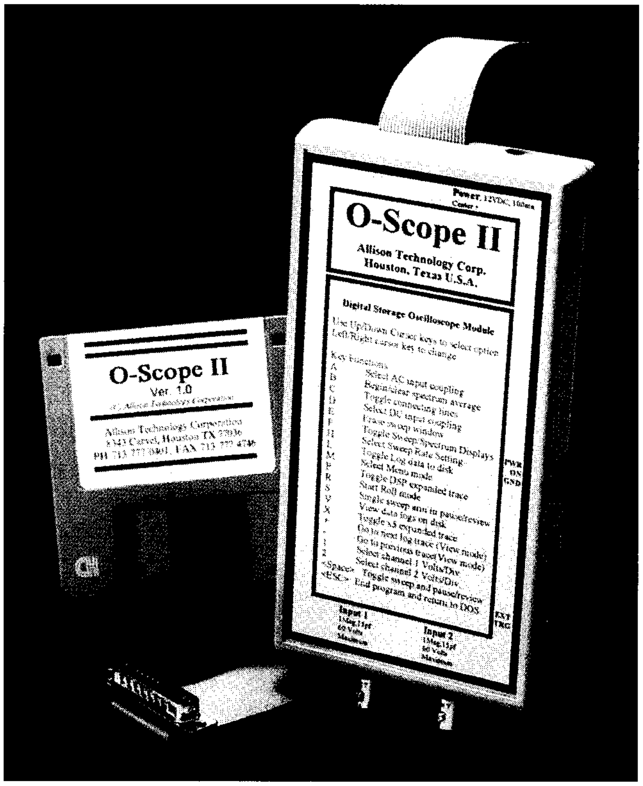
The DMS-20PC-FM sells for $65. Quantity discounts are available.

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

#506

#104

NEW PRODUCT NEWS



DIGITAL STORAGE SCOPE

The Allison O-Scope II is a compact module that plugs into the printer port of a PC and transforms it into a dual-trace digital storage oscilloscope. Uses for the scope include audio and stereo equipment, automobile diagnosis, data logging, field service, motor controls, noise analysis, power supplies, and vibration.

The O-Scope II can freeze displayed sweeps on the screen, print them out, or save them to disk. Operating parameters such as input range, sweep rate, and trigger level are displayed and easily adjusted through the computer keyboard. A versatile external trigger is included. In the frequency-spectrum mode of operation, the O-Scope II presents sweeps of a selected frequency range on a calibrated graph with the option of averaging.

The O-Scope II features simultaneous 2-channel data capture for accurate phase measurements. Bandwidth is 250 kHz with sampling rates up to 1,000,000 per second. The minimum PC required is a 12-MHz '286.

The O-Scope II costs $349. Other models start at $169.

Allison Technology Corp.
8343 Carvel • Houston, TX 77036
(713) 777-0401 • Fax: (713) 777-4746 **#507**

STEP-UP DC-DC CONTROLLER

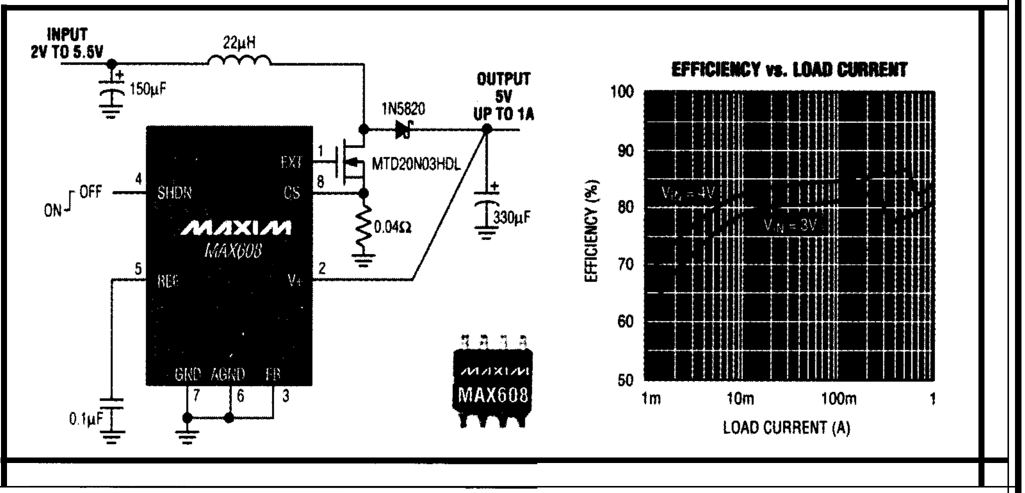
The **MAX608** is a low-voltage, step-up DC-DC controller that starts up from inputs as low as 1.8 V and delivers up to 3 W from a two-cell NiCd battery. Typical efficiency is 85%. The chip is ideal for two- and three-cell battery-powered systems.

The MAX608 allows portable systems to run longer because of its high efficiency and low supply currents. A unique current-limited pulse-frequency-modulation control provides high efficiency at heavy loads while using only 85 µA (typical] when operating with no load. In addition, a logic-controlled shutdown mode reduces supply current to 2 µA (typical).

The MAX608 switches at up to 300 kHz, reducing the size of external components. It operates in bootstrapped mode only (with the chip supply, OUT, connected to the DC-DC output). The chip is available in 8-pin SOIC and DIP packages and drives a low-cost external N-channel switch. The output voltage is factory set at 5 V and can be adjusted from 3 to 16.5 V with an external resistor divider. The input range is 1.6-16 V.

The MAX608 sells for $1.89 in quantities of 1000 up. An evaluation kit is also available.

Maxim Integrated Products
120 San Gabriel Dr.
Sunnyvale, CA 94086
(408) 737-7600 **#508**



FEATURES

FEATURE ARTICLE

Brian Millier

Microprocessor-Controlled High-Voltage Power Supply

Component integration is bringing some very odd feature mixes. Brian shows us his unusual project—constructing a power supply with a variable voltage of 500–1800 V and a maximum current of 100 mA.

You're browsing through a trade journal or magazine and you come across an advertisement for a new product. The manufacturer extols its features: 24-bit digital signal processing, 18-bit A/D and D/A converters, and vacuum tubes. The tubes are not the large transmitting type, either-they're 12AX7 receiving tubes.

Unless you're a musician producing contemporary music, you likely scoff at such an eclectic component mix. However, for music recording and processing equipment, it's fashionable to use vacuum-tube amplifiers along with the best digital-signal-processing ICs available. Manufacturers claim the venerable 12AX7 dual triode produces a warm sound or mellow distortion DSPs can't duplicate.

The HV power supply I describe in this article doesn't use vacuum tubes, so don't scramble for your old Greenlee chassis punches to prepare the chassis for the tube sockets! Considering the advancement in state-of-the-art switching-mode power supplies, however, some aspects of the design place it in the same unusual category as music-recording products.

An individual probably couldn't produce a switching-mode power supply of the type used in modern computers for anything close to the cost of buying one. However, if you need high voltages and moderate currents from your power supply and want a floating output, adjustable voltage, and current limiting, this article may give you some ideas to help you design just what you need.

DESIGN NEEDS

The power supply I designed and constructed had to produce a variable voltage from 500 to 1800 V at a maximum current of about 100 mA. It needed a floating output (not referenced to ground) and an adjustable current limiter.

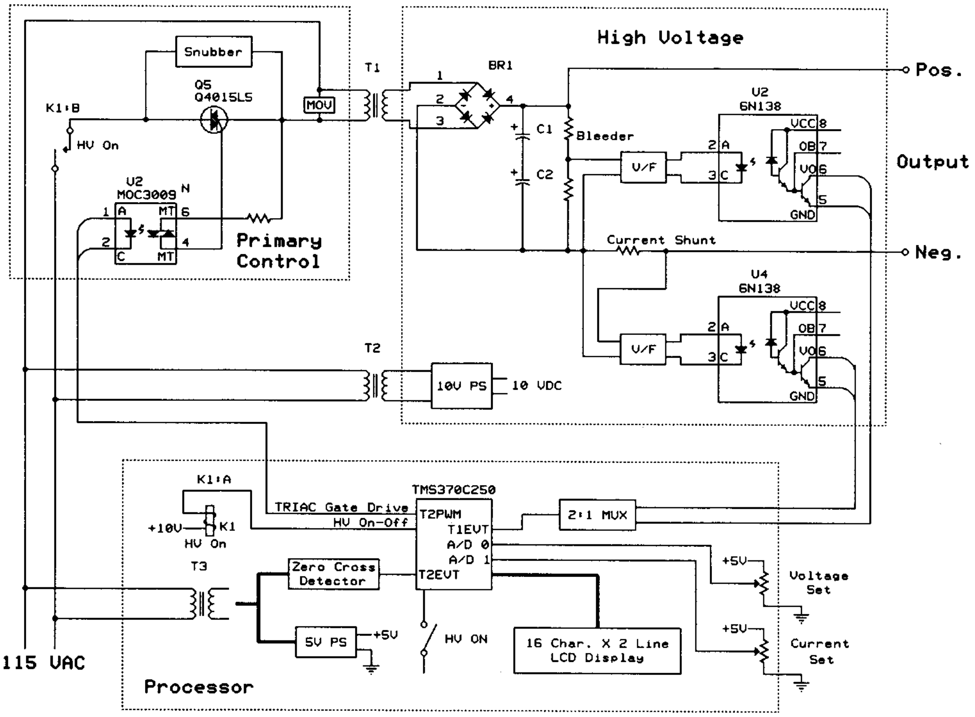


Figure 1—*The microprocessor-controlled high-voltage power supply is designed around three circuit boards, each performing a discrete function. This figure shows the overall block diagram of the circuit and details this functional breakdown by circuit board.*

Those of you who've worked in biology labs probably recognize this as an electrophoresis power source. Having built a number of such supplies using analog circuitry, I decided to design one with a microcontroller directing control and display functions.

Though I didn't use vacuum tubes, I did use a thyristor, the semiconductor invented by General Electric in the 1950s, for the actual power control. Combining this $1.50 semiconductor, a $12.50 microwave-oven transformer, a microcontroller, and an inexpensive LCD display yielded a cost-effective design.

I chose the TI TMS370 microcontroller and a custom-processor PCB, but other controllers could be used. The Motorola 68HC11 is well-suited to the task, and many single-board controllers feature it.

In designing any high-voltage regulated supply, you first need to decide what method of control achieves regulation. If you use a transformer, rectifier, and filter followed by a series-pass element, the latter requires a voltage breakdown rating greater than the raw DC-supply voltage less the lowest-rated output voltage (i.e., 2000 V – 500 V = 1500 V).

The pass element also dissipates large amounts of power if the rated

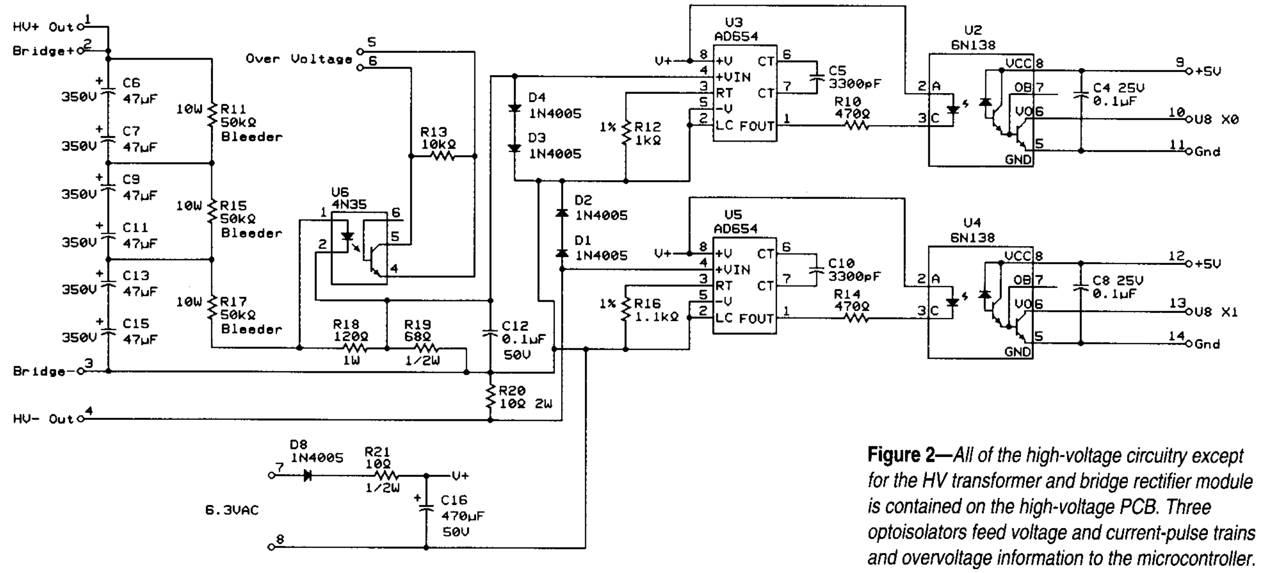


Figure 2—*All of the high-voltage circuitry except for the HV transformer and bridge rectifier module is contained on the high-voltage PCB. Three optoisolators feed voltage and current-pulse trains and overvoltage information to the microcontroller.*

current drawn is at the lower voltage limit. While specialized vacuum tubes work well under these conditions, they produce a lot of heat and require more work than semiconductors.

A high-voltage supply which floats with respect to ground presents additional challenges. Feedback signals representing both output voltage and load current are needed for regulation. Derived from a floating output, these signals must also float.

At some stage, the feedback signals are compared with setpoints entered by the user, usually from front-panel controls. For safety, the controls and the supply cabinet must be grounded. Clearly, comparing floating analog voltages with grounded ones is problematic. And, if the output voltage and current are displayed with conventional meters, the meters also need to withstand several thousand volts with respect to ground.

For these reasons, I used a microcontroller to phase control the 60-Hz AC waveform fed to the primary of a step-up transformer. I also used a conventional full-wave bridge rectifier and filter and converted the output-voltage and load-current feedback signals from analog to digital pulse trains so they could be transmitted through optoisolators. This conversion maintained the supply's output floating nature and provided digital signals which the microcontroller could easily measure and display.

CIRCUIT DESCRIPTION

As you can see in Figure 1, **115-V** AC power is supplied to the high-voltage transformer T1 through relay contacts Kl-B followed by a Q4015L5 triac. This device has a 400-V PRV and can control 15 A with only a small heatsink. The HV on relay K1 is jointly controlled by the microcontroller and the front panel HV on and off switches.

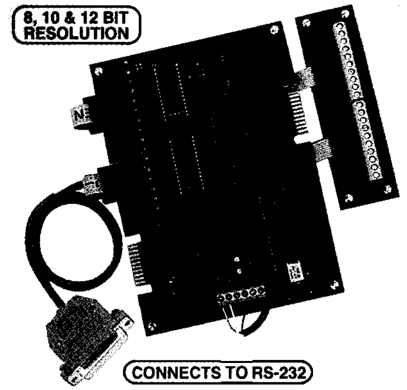
The phase-control drive to the triac gate is supplied via optoisolator MOC-3009N, a device specifically designed for thyristor gate drive applications. Rounding out the primary circuit is a snubber network placed across the triac and a MOV placed across the transformer primary. These standard design features protect the triac from the back-EMF produced by the inductive nature of the HV transformer.

The HV transformer's secondary, rated at 1.8 kV, feeds a chassis-mount high-voltage bridge-rectifier module. The rest of the high-voltage circuitry is mounted on a PCB, detailed in Figure 2. A high-voltage capacitor bank of six 47-µF, 350-V capacitors takes care of filtering. A bleeder resistor network, made up of three series-connected 50-kΩ, 10-W resistors, is connected across the capacitor bank. This network:

- safely discharges the capacitors quickly when the supply is shut off, even if no load is connected to the supply
- equalizes voltages across each of the three capacitor pairs to ensure the 350-V rating is not exceeded. This is important since the voltage across each capacitor in a series string is inversely proportional to the individual capacitance. (As electrolytic capacitors, their capacitance tolerance is quite wide.)
- forms one-half of the voltage divider needed to reduce voltage to the 1-V full-scale range of the voltage-to-frequency converter. The converter provides a voltage feedback to the microcontroller.

A 10-R resistor, R20, placed in series with the negative supply lead acts as the current shunt and produces a 1-V signal with the full-scale output current of 100 mA. Some Analog Devices AD654JN voltage/frequency (V/F) converters change both this and the voltage-divider signals into pulse trains that the microcontroller can interpret for feedback. Since the power-supply output is a floating high voltage, 6N138 optoisolators feed both the voltage- and current-sense pulse trains to the microcontroller.

A small half-wave supply consisting of D8, R21, and Cl6 provides unregulated 10 VDC to operate the V/F converters. A Hammond 166G6 filament transformer, with the better-than-2000-V isolation needed between primary and secondary, provides the 6.3 VAC needed for this supply.



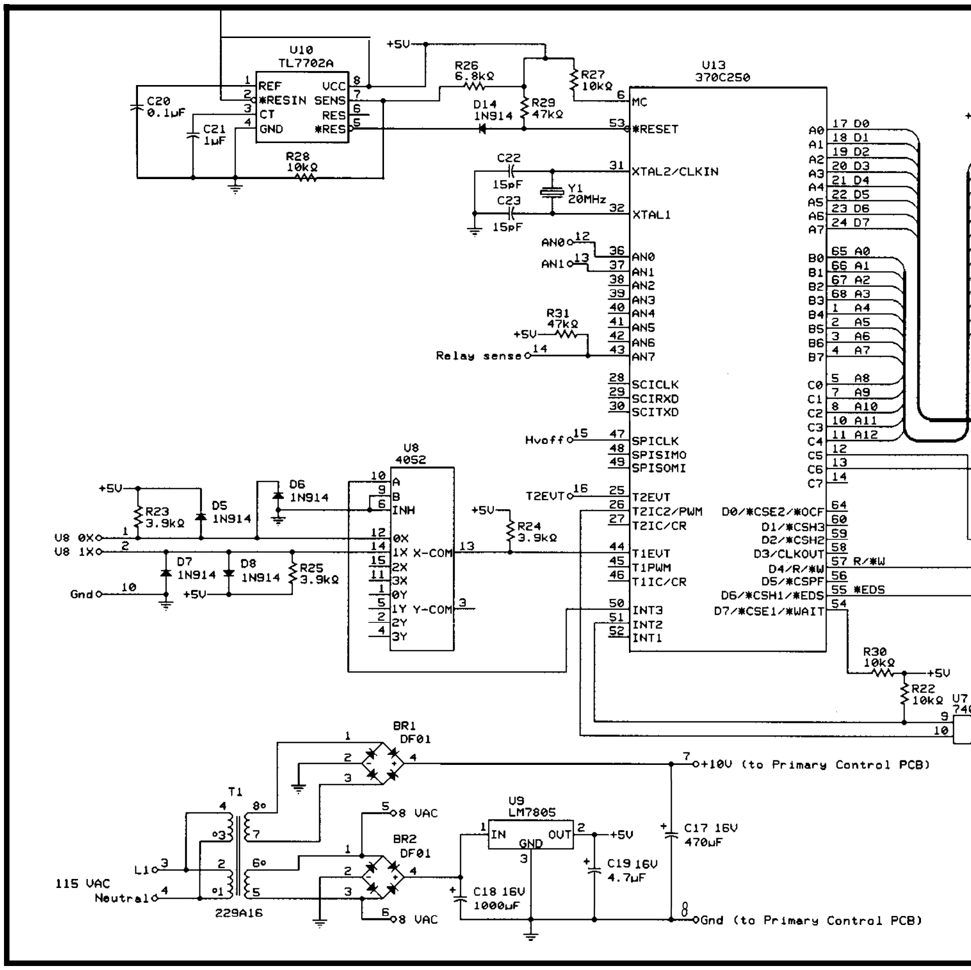


Figure 3-A Texas *Instruments TMS370C250 microprocessor performs all control and display functions. You may wish to modify the design to use the more commonly available 68HC11 or 8051 SBCs.*

Rounding out the HV PCB is a safety circuit made up of optoisolator U6 and resistors R18 and R13. This circuit shuts down the supply if an overvoltage condition occurs. The chosen value of R18 provides the 1.6 V needed to fire U6's LED when the HV rises beyond 2000 V. The output transistor in U6, connected across the HV off switch, shuts off the HV under these conditions.

The TMS370C250 (see Figure 3) is a custom microcontroller PCB that I designed for other projects but is flexible enough for this design. The core circuitry is completely conventional. U10, a TL7702A power-supervisor chip, ensures that the microcontroller stays in reset until the logic supply is within proper operating range. In any design where the microcontroller plays such a critical function, a power-supervisor circuit is an absolute necessity. Numerous signals external to the PCB itself, coming from and going to the TMS370, are marked by terminals numbered 1-19.

Figure 3 depicts the 5-V logic supply and 10-V supply used by the primary-control PCB. U7 provides a bit of glue logic for the LCD display. The important phase-control drive signal to the triac is labeled *MOC3009 drive.

Although the TMS370 has two general-purpose counter/timers and a watchdog timer, only one timer is free to monitor voltage and load-current pulse trains. CMOS multiplexer U8 selects one of the two feedback-pulse–train signals for counting by the TMS-370T1EVT input. Another TMS370 counter/timer connects to the output of a zero-crossing detector. This detector provides pulses synchronized with the AC-line waveform needed for the AC-phase control of the triac. As Figure 4 indicates, this circuit consists of Q4 and associated components.

Since users were accustomed to rotary potentiometers for both voltage- and current-limit settings, I used pots in this design. Both pots are connected across the 5-V logic supply. The microcontroller's internal 8-bit A/D converters then measure the wiper voltages ratiometrically. Figure 5 shows the wiring of these pots to the processor PCB. The cost and convenience of this implementation outweigh the temptation to use a numeric keypad.

An inexpensive 16 x 2 LCD module, driven by the microcontroller, provides readout of output voltage and current. Because it gets feedback on both values, the microcontroller can provide this information, as well as a latched indication of any fault conditions that may have shut down the supply. The cost of the LCD module is much less than that of two DPMs. In addition, DPMs monitor a floating high voltage and thereby complicate the physical design, since they'd have to stand off up to 2000 V to chassis ground.

TRIAC PHASE CONTROL

To clarify how a microcontroller can control power using phase control, a short description of how a triac works may be useful. For both the positive and negative half of the AC-

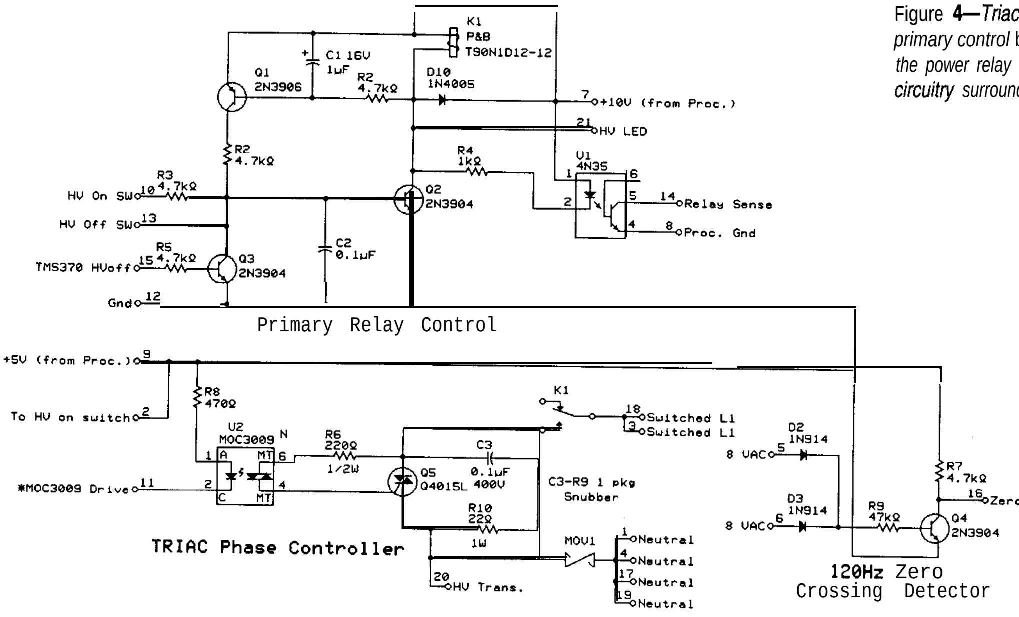


Figure 4—*Triac phase control is done on the primary control* board. A *simpler circuit to control the power relay could be* substituted *in place of the circuitry* surrounding *Q 1, Q2, and Q3.*

line waveform, a triac can conduct current through MT1 to MT2 for the remainder of that half-cycle by triggering its gate terminal with a short pulse. The polarity of the pulse must be the same as that of the AC waveform for that half-cycle.

At the end of each half-cycle, when the voltage across MT1 and MT2 drops to zero, the device unlatches and current ceases to flow, awaiting the next gate pulse. If the gate-trigger pulse occurs very early in the half-cycle, the triac provides power to the load at almost 100%. Conversely, a trigger pulse much later in the half-cycle (approaching the 180°-phase angle) provides almost no power. Figures 6a and 6b offer examples of these two triggering conditions.

The waveform is sinusoidal and power is proportional to the square of the voltage applied to the load. So, the relationship between the trigger-phase or firing angle and the power delivered to the load is complex.

Figure 7 depicts this relationship for full-wave phase control with a resistive load [1]. Clearly, the change in power delivered to the load doesn't change much at either extreme of firing angle. In this design, the DC-output-voltage range does not adjust down to zero nor up enough to require the full transformer-output-voltage rating. Therefore, the phase controller operates in the more linear center region (see Figure 7).

Although a closed-loop controller tolerates some nonlinearity in the transfer function of the feedback model, it works best when nonlinearity is minimized. Operating in this region also minimizes the complications that must be considered when driving a complex load with a triac.

This discussion assumes a strictly resistive load. However, a power trans-

BUG KILLIN' IS GREAT WORK WHEN YOU CAM: GET IT BUT WHEN THE COIN' BOGS DOWN, I GET OUT PARADIGM DEBUG. STAND-ALONE OR TEAMED UP WITH AN IN-CIRCUIT EMULATOR, IT'S A LICENSE TO KILL.
TO BE CONTINUED...

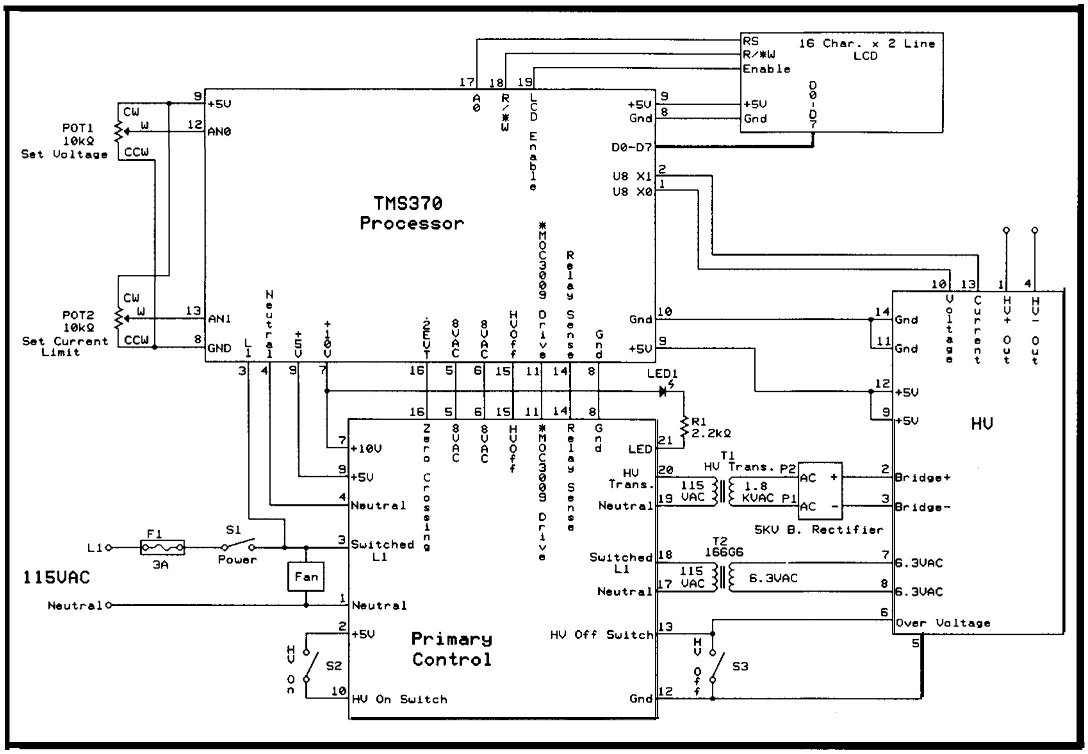


Figure 5—*The* power supply *functions are divided among three interconnected PCBs.*

former presents a complex load in which the current lags the voltage to some extent, complicating things considerably. If the firing angle never approaches either 0” or 180°, the sticky design details that arise due to phase shift between the voltage and current through the triac are largely avoided.

It is beyond the scope of this article to delve into the complexities of phase control of highly inductive loads needing high power. Suffice it to say, this design works well here.

THE POWER CONTROL LOOP

To provide power control, synchronize the microcontroller with the AC-line zero crossings (at a rate of 120 Hz) and have it send trigger pulses delayed by a variable amount of time from the zero-crossing point. The time delay provides the necessary firing angle. It is the microcontroller’s job to:

- monitor both power-supply output voltage and current
- compare these values to the user’s setpoint parameters
- derive the necessary delay time to produce the correct firing angle.

Since the controller is operating in closed-loop mode, this process is iterative. The controller’s firmware algorithm constantly makes small adjustments to the firing angle to produce as small an error as possible between actual and setpoint voltage. This process is subject also to the criteria that the current must not exceed the current-limit setpoint.

This controller varies the AC power to the primary of a transformer to control the DC output coming from a rectifier/filter connected to the secondary. Therefore, it must take into consideration the time constant of the filter network. For a 60-Hz system using reasonably large filter capacitors, the time constant is significant.

In an analog controller, the feedback response must be reasonably highly damped-typically in the fractions-of-a-second range-or the system oscillates or hunts. Some oscillation around a setpoint is inevitable in any closed-loop system, but how much is tolerable depends on the application’s need for stability.

In this design, the microcontroller takes alternating voltage and current readings six times per second. This somewhat slow rate is dictated by the desire to get a high-resolution reading using voltage and frequency conversion. Using these three readings per second, it makes corrections to the firing angle until the proper output conditions exist. The microcontroller’s timer circuit has a resolution of 200 ns, so the firing angle can be adjusted by this small amount.

The slow feedback response and the high resolution of the correction minimize hunting problems caused by the

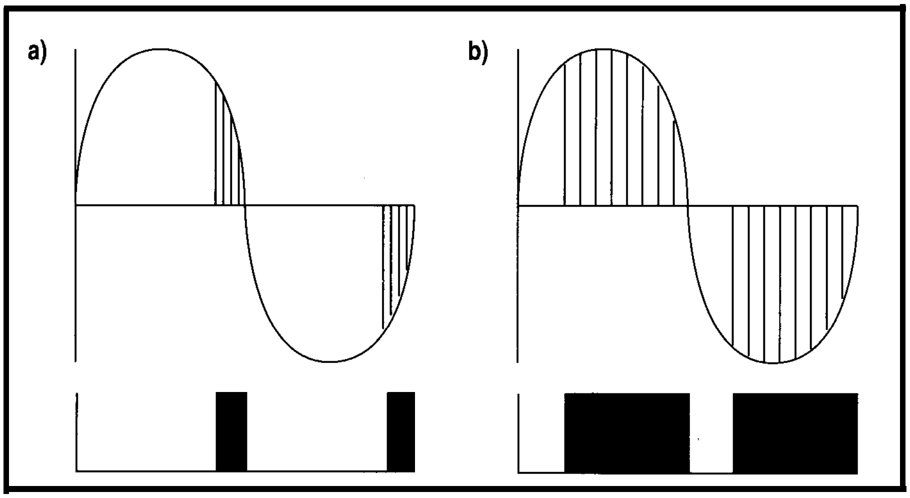


Figure 6-a) *Large firing angles deliver very little power to the load. b) As the firing angle decreases toward zero, the delivered* **power** *approaches 100%.*

filter time constant considerations. They also mean poor transient response to rapid changes in either line or load. However, this characteristic isn't a problem in this application because only slow changes occur in the load. The design would not be suitable if the load required a very tight voltage tolerance and the line voltage was susceptible to rapid fluctuations.

Figure 4 pictures the primary control PCB. The ● MOC3009 drive signal from the microcontroller turns on the MOC3009 optoisolator, specially designed for triac-gate firing. C3 and R10 form a snubber across triac Q5, and the 14K241U MOV is placed across the HV transformer primary to absorb back-EMF spikes. Only a small heatsink (common TO-220-style, 2" sq.) is required.

The 120-Hz zero-crossing detector is located on this PCB as well. D2, D3, and two diodes in BR2 (on the processor PCB) provide a full-wave rectified but unfiltered waveform which forces Q4 into conduction for all but a short interval around the AC line's zero crossing. These short positive 120-Hz pulses are fed to the TMS370's T2EVT pin. The firmware generates the necessary triac firing angle as a time delay from these pulses.

The unusual circuitry associated with Q1, Q2, Q3, and relay K1 turns the high voltage on and off mechanically, independent of the triac control. In most commercial electrophoresis power supplies, separate momentary-contact switches control HV on and off. I followed this convention, including Q1 and Q2 to form a latch circuit.

Momentarily activating the HV-On switch sends 5 V through R3, which turns on Q2 and pulls in K1. The resulting voltage drop across Kl's coil biases Q1 through R2. Current then flows through Q1 to maintain Q2's conduction even after the HV-On switch is released. Power to the HV transformer's primary flows through K1:B contacts until Q2 stops conducting, which causes the relay to drop out. This effect can result from:

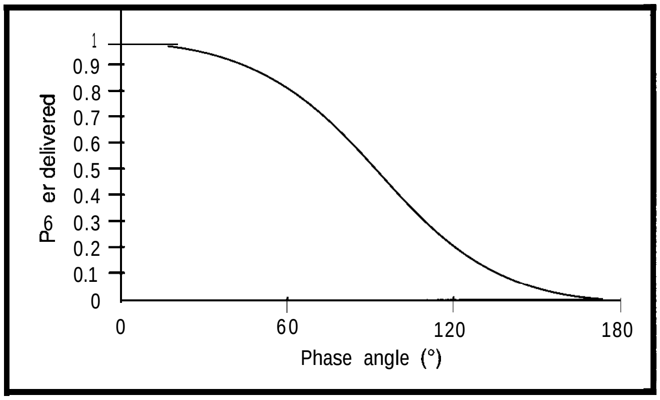


Figure 7—*The relationship between the phase angle (or firing angle) and delivered power is not a linear one.*

- the user pressing the HV-Off switch momentarily, grounding the base of Q1
- an overvoltage condition sensed by circuitry on the HV PCB (This condition causes conduction of a 4N-35N optoisolator which, since it is in parallel with the HV-Off switch, has the same effect.)
- the TMS370 also receiving an output signal (TMS370 HV off) which turns Q3 on when it is asserted. This signal is issued in response to fault conditions the microcontroller senses.

The state of relay K1, and therefore the presence of HV, depends on several conditions (all of which are latched by K1 and associated circuitry). The microcontroller senses the state of this relay through U1, another 4N35N optoisolator. This isolation keeps relay-coil spikes from reaching the processor.

For most purposes, one could replace all of the circuitry with a non-momentary contact switch labeled HV On/Off, a relay-drive transistor, and a relay. Then, any error conditions sensed by the overvoltage sensor or the microcontroller could achieve shutdown just by depriving the relay drive transistor of base drive. Next time!

MICROCONTROLLER NEEDS

Commented assembly-language source is available on Circuit Cellar

FOR EMBEDDED C/C++ APPLICATION5 USING THE AWESOME BORLAND & MICROSOFT COMPILERS, I BET MY LIFE ON PARADIGM LOCATE & PARADIGM DEBUG. CAN YOU TRUST ANYTHING ELSE?
TO BE CONTINUED...

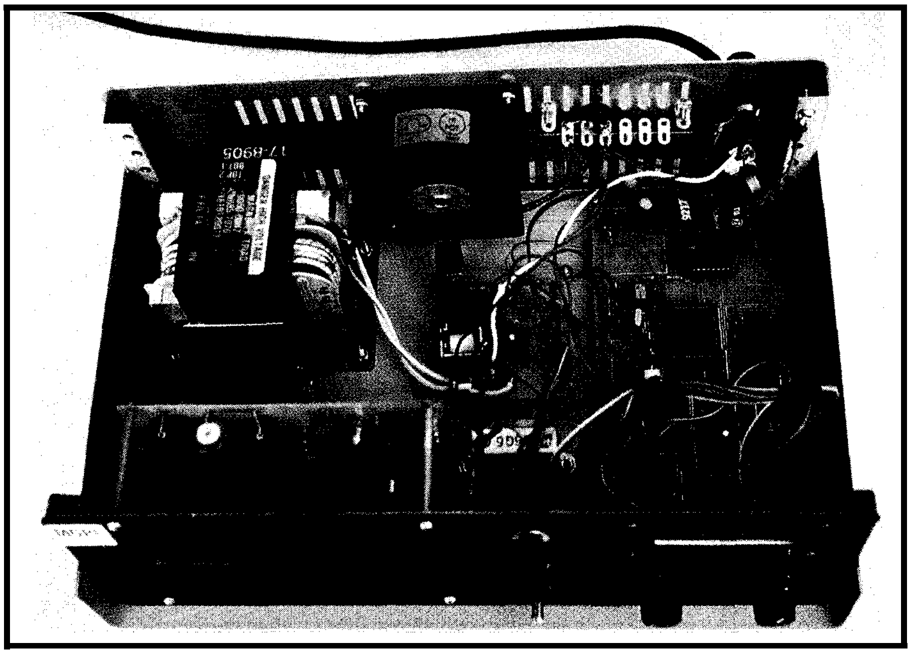


Photo 1—*In* an *overhead view of the power supply, the microcontroller PC6 is to the extreme* right, *the* primary *control PCB is in the middle near the fan, and the HV section is fo the left.*

BBS. I'd like to go over a few of the basic details.

Regardless of the microcontroller used, some minimal functional blocks must be present either in the microcontroller itself or in supporting peripherals. These blocks include:

- an input-capture function which accurately represents the exact time of each 120-Hz zero crossing. This design uses the Timer 2 event input.
- a pulse-width modulation block. In this case, the period of the PWM is fixed at $\frac{1}{120}$ s (8.33 ms). The firing angle is the variable parameter defined as a time delay after the occurrence of the zero crossing. This design uses the Timer 2 PWM output, with the 200-ns system clock used as the clock source. This setup results in a very high firing-angle resolution of:

$$\frac{180''}{\frac{8.33\text{ ms}}{200\text{ ns}}} = 0.0043''$$

- a 16-bit event counter to total the pulses from the V/F converter that measures voltage and load current. These readings are used for both feedback-loop and display purposes. This design uses the watchdog counter for this function. The 68HC11 doesn't have a 16-bit counter, but it does have an 8-bit counter with an overflow interrupt which could serve the same purpose.
- a two-channel, 8-bit A/D converter to monitor the voltage and current-limit setpoint potentiometers. Most modern microcontrollers with any form of built-in A/D converters usually provide at least 4 channels. You can configure the extra channels to measure such things as the temperature of the device(s) the supply is powering.

  This configuration enables one to design in a safety shutdown or foldback current limiting. Modern linearized thermistors are reasonably inexpensive and provide large resistance changes over temperature. They are thus well suited for measurement by the limited resolution of the 8-bit A/D converters available in the microcontroller.
- a few available I/O pins to drive the primary power relay and sense some switches.

First, the microcontroller program initializes some of the functional blocks. It then enters the setting loop, which allows the user to enter the desired voltage- and current-limit conditions prior to turning on the high

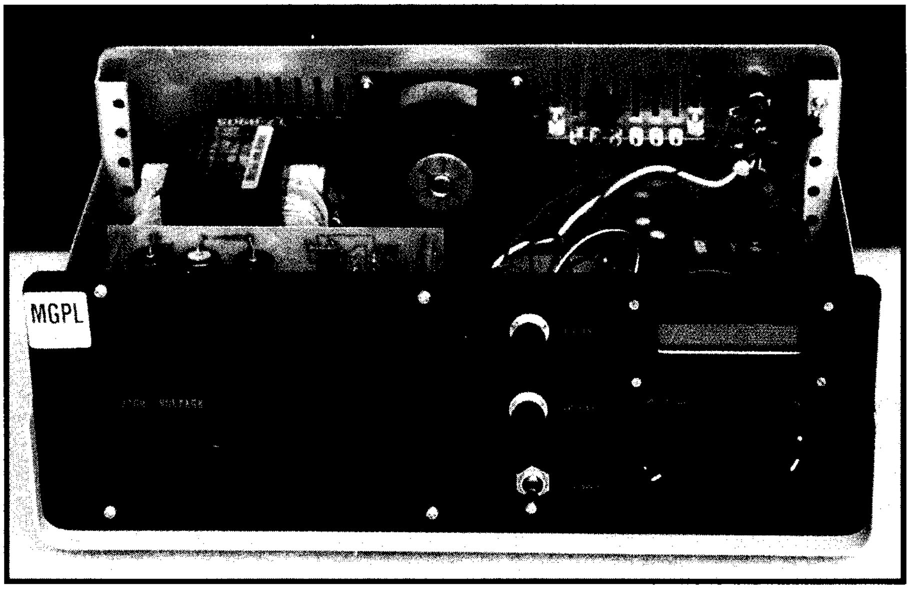


Photo 2-A view *of the unit's* front panel *shows the adjustment* pots and LCD display.

voltage. The program remains in this loop until it senses that the HV-On relay has been activated as a result of the user pressing the HV-On switch.

At this stage, the program enters the HV voltage-control feedback loop and remains there until the user presses the HV-Off switch or some fault condition occurs. The flowchart for this loop is shown in Figure 8.

In operation, the supply takes a few seconds to ramp up from zero to the setpoint voltage, at which point it regulates around that setpoint. As mentioned earlier, this design doesn't react very quickly to rapid line or load changes. When the high voltage is shut down, intentionally or because of a fault condition, the LCD displays, for safety reasons, the following message: WAIT 5 seconds-Discharging.

The program code, since it requires less than 1 KB of memory, fits into the onboard EPROM of many microcontrollers. In this design, I used a lower-cost TMS370C250 ROMless version and a 2764 EPROM.

CONSTRUCTION PRACTICES

My intent was to describe phase control using a microcontroller. However, this particular design features a floating high-voltage output. Therefore, a discussion of construction practices and safety are in order.

Any power supply capable of producing the voltages and currents this one does can also produce lethal shocks and must be treated accordingly. If you're unaccustomed to working with high voltages, I advise you to obtain some applicable literature on the subject before attempting to construct such a device.

The case should be metal, with a three-wire power cord providing the case ground. All circuitry on the HV PCB is floating with respect to ground at up to several thousand volts. Mount this board using 2" standoffs to keep it well away from the grounded case. Isolation of this voltage is also maintained by the three optocouplers on the board. Ensure that they are correctly wired and that the wiring on the HV side is well separated from that on the ground-referenced output side. Photos 1 and 2 present the layout in my power supply.

Use 5000-V test-lead wire for all high-voltage wiring. Make sure the bleeder-resistor chain is connected across the HV-capacitor bank and is functional. Before you troubleshoot, use a meter to check that the capacitors are being discharged properly.

The microwave-oven transformer I used has one of its secondary leads cut short and connected to the core ground. I removed this lead from ground and extended it to the AC in-

TO BE CONTINUED.

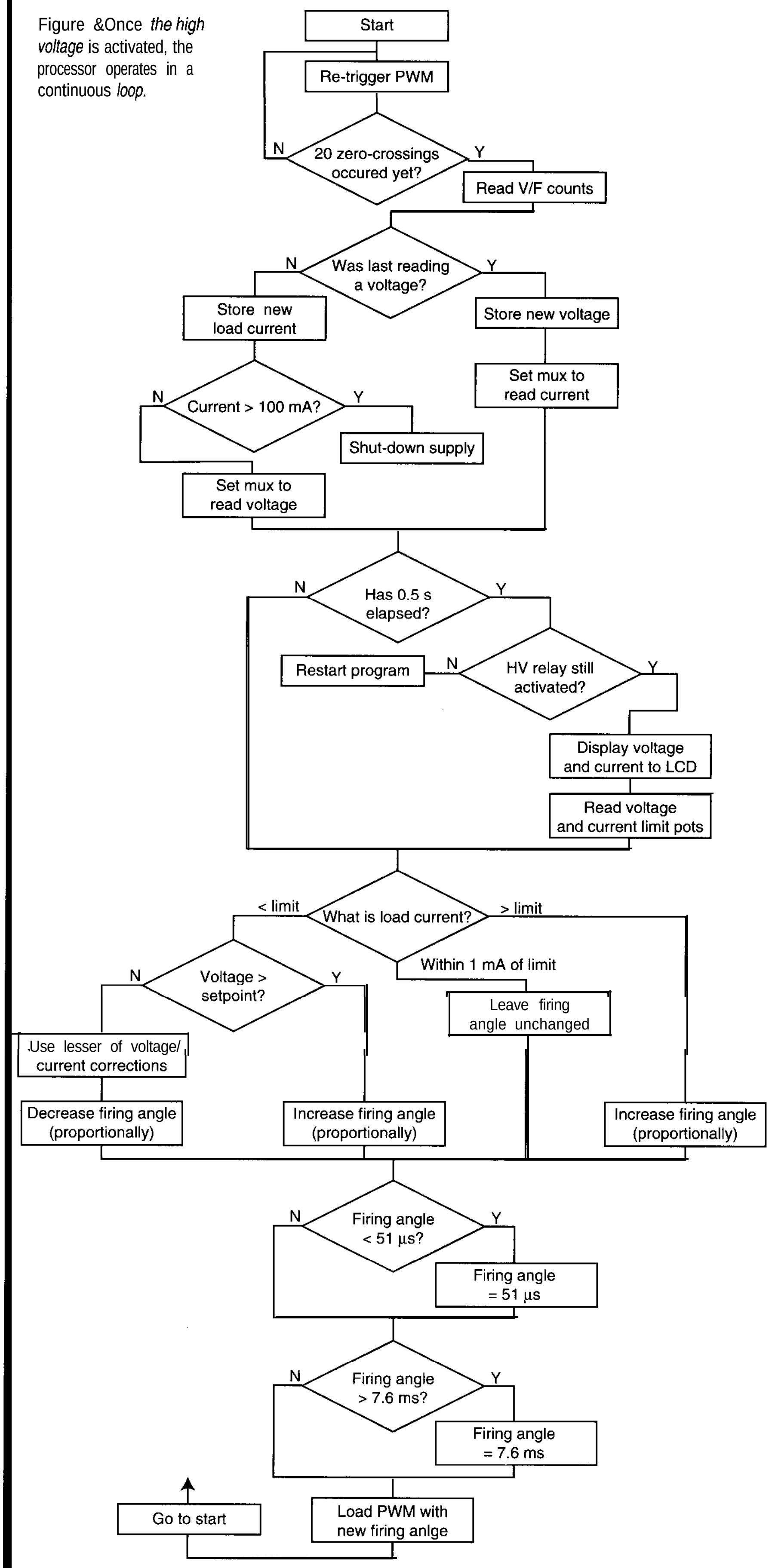


Figure &Once *the high voltage* is activated, the processor operates in a continuous *loop.*

put of the bridge rectifier using test lead wire and heatshrink tubing. The filament winding was unused and its leads trimmed off. The transformer was designed to supply much larger amounts of power on an intermittent basis, so a small Sprite fan was installed to cool the assembly.

Of course, the code wasn't 100% correct the first time I tried it. I anticipated this and didn't want to be probing around with a scope when such high voltages were present. So, I created a simulation.

In place of the high-voltage transformer I substituted a large low-voltage one. I lashed up a rectifier/filter and used a different voltage-divider ratio on the output to provide the required 1 -V fullscale needed for the V/F converters. While the feedback response dynamics were somewhat different, this simulation was good enough to test the code out.

You should follow the same route if you're unfamiliar with high voltages. Get the control circuitry working in a simulation. Then test the HV transformer, rectifier, filter, and bleeder assembly. I'd suggest using a variac so you can slowly increase the primary voltage to the HV transformer while monitoring the HV output voltage. Using a scope and a pull-up resistor, monitor the pulse train from the voltage opto's phototransistor output. This procedure should produce about 15 kHz at 1250-V output. The current-limit circuitry is less critical and can be tested when everything else is up and running.

When the power supply is functional, check its output to make sure it floats with respect to ground. You can also connect a digital meter from its negative terminal to earth ground to ensure that the leakage current is in the low-microamp range with the power supply running at full output. For commercial applications, high-pot testing and many other measures are also required to meet government testing standards.

SHUTDOWN

Application for this power supply is specialized. Outside of biology labs, where they are common, you'd hardly ever encounter them. However, I hope you can apply the ideas presented here when designing your own microcontroller-based power supplies. ❑

I'd like to thank Doug Cooke of the Marine Gene Probe Laboratory for his help designing and testing this project.

Brian Millier has worked as an instrumentation engineer for the last 12 years in the chemistry department of Dalhousie University, Halifax, NS, Canada. In his leisure time, he operates Computer Interface Consultants and has a full electronic music studio in his basement. He may be reached at brian.millier@dal.ca.

REFERENCE

[1] D.R. Graham, Ed., *General Electric SCR Manual,* Fifth Edition, General Electric, Syracuse, NY, 235, 1972.

SOURCES

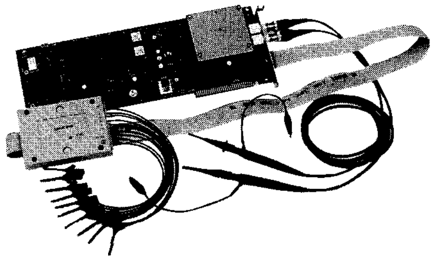
HV transformer TM91XFMR1858 and bridge rectifier TM93RCR2339
Herbach and Rademan
18 Canal St.
P.O. Box 122
Bristol, PA 19007-0122
(215) 788-5583
Fax: (215) 788-9577

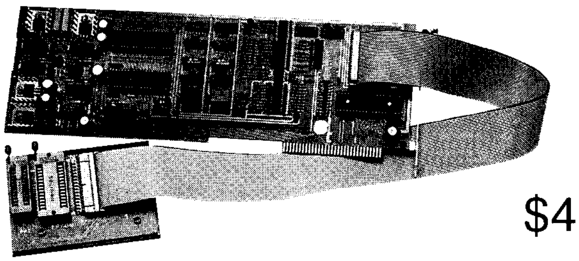
TMS370 microcontroller family
Texas Instruments, Inc.
9301 Southwest Fwy.
Commerce Park, Ste. 360
Houston, TX 77074
(713) 778-6592
Technical Hotline: (713) 274-2370
BBS: (713) 274-3700

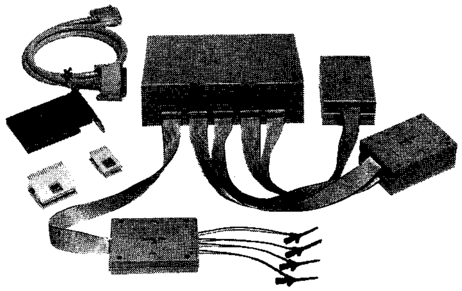
V/F converters AD654JN
Analog Devices
One Technology Way
P.O. Box 9106
Norwood, MA 02062-9 106
(617) 329-4700
Fax: (617) 326-8703

I R S

401 Very Useful
402 Moderately Useful
403 Not Useful







FEATURE ARTICLE

Edward Schram

Push-Pull Switching Regulator Design and Application

Too often power-supply changes force a design into a corner. Ed spends some time looking at how and when to choose a power supply. Surprisingly, it's not necessarily best to decide beforehand.

Why, after all the work is done and the sheet metal has been designed, does the maker of that perfect little power supply decide to change it?

And why doesn't someone make a +5-, ±12-, +24-V power supply in a small package? Oh yes, the +5 V needs tolerances for a microprocessor, and the 12-V lines are only driving RS-232 lines so they can be sloppy. But, the +24 V is used for programming logic so it needs a tight tolerance.

Sorry, did I forget to mention the ±8-V supply? It's for analog circuitry, and needs to be filtered pretty well. Input power is 115 VAC or 12 VDC. I want it to be mobile and work in the car. Can this be ready in two weeks!

How many times have you asked your vendors these questions?

After a couple of products, a designer either picks a power supply and designs around it or comes up with a way to generate the required power after the design is done.

Although the first solution means a quick design and use of an off-the-shelf power supply, I prefer to design the logic and associated circuitry, then work out the supply. Better yet, doing the supply along with the design offers tradeoffs in both when necessary (or convenient).

So, how do you get those odd little voltages and meet other requirements? With a little practice and good fortune, designing a switching power supply that meets the requirements of a design is not difficult. Most designs can be prototyped, tested, and ready for manufacturing in a few days.

In this article, I'll provide you with the basics I've learned over time. A working system that allows you to prototype most any design should give you some experience. To make it easier to wind the transformers and get the supply working, the design is based on the push-pull topology. This type of converter isn't best for every application. However, it can be created in less than a day and does the job well.

For my discussion, I'll design a 50-W power supply that works from a 115-VAC line power or 12 VDC. The outputs are 5 V at 5 A, ±12 V at 0.6 A, and ±8 V at 0.5 A. Figure 1 shows the complete design.

The difference in the total output power and rated power accommodates the losses in the design. Besides, who makes a 47.4-W power supply? Since its overall efficiency is approximately 75 %, the supply is actually 133% of 47.4 W or 63 W. In other words, about 0.5 A is drawn from the source-a handy number if you should decide to do something silly like fuse the input.

A push-pull design includes:

- input supply voltage specification
- input supply rectification
- input supply load sharing
- input filtering to produce input DC voltage
- input DC voltage to working supply
- input DC voltage to pulsating DC
- pulsating DC transformed into output pulsating DC voltages
- output rectification and filtering
- addition of linear regulators as required to meet voltage needs

INPUT SECTION

The input section preconditions the system. It handles any input rectification, filtering, and load sharing from multiple sources. Filtering not only provides a clean source of power, but also keeps switching noise from escaping back to the source.

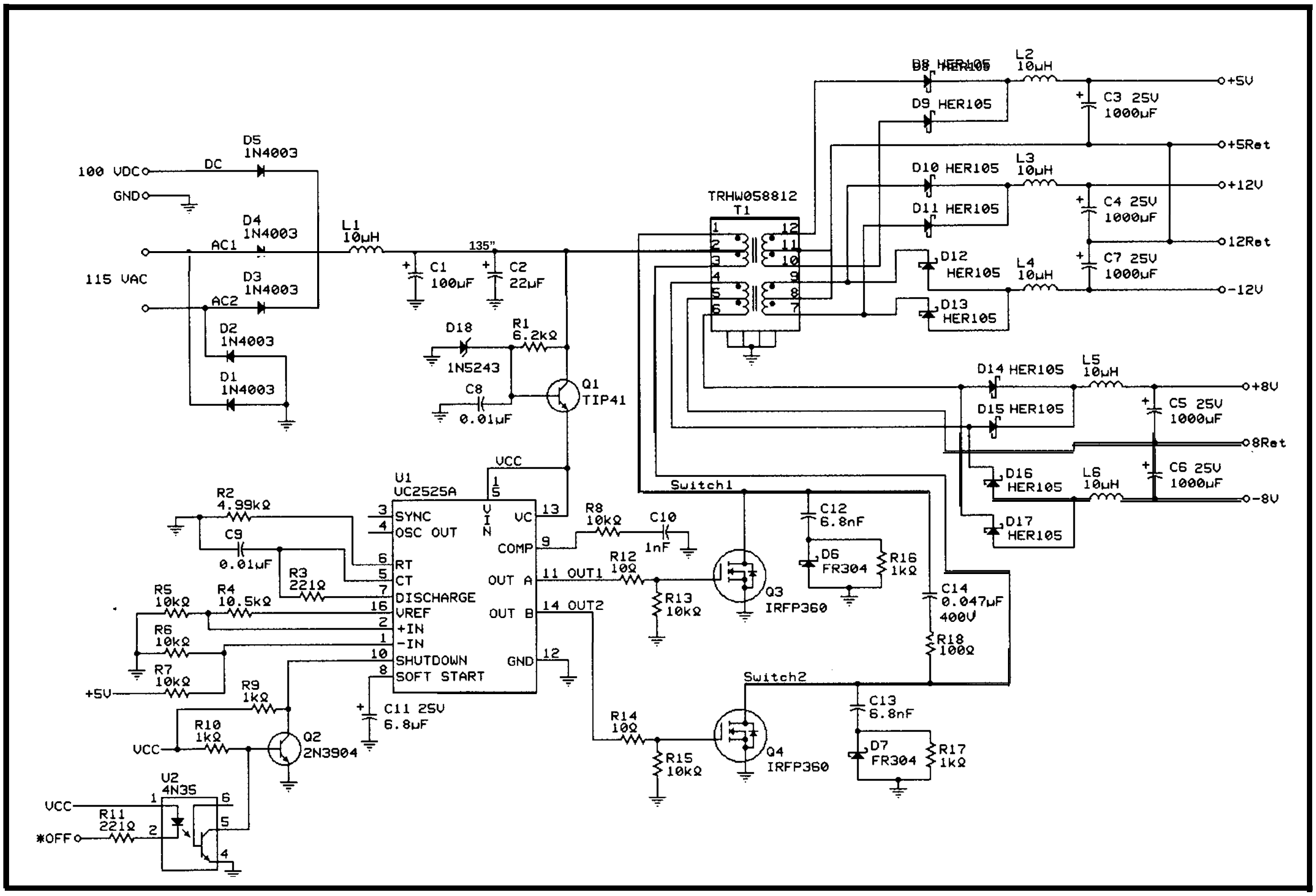


Figure 1 -A wide range of input and output voltages can be handled with a low parts count and price when you design the power supply yourself.

Load-sharing diodes should match the rectifiers in input rectification. The diodes must be able to withstand the maximum reverse voltage of the highest input voltage in the system and the current from the source. I use 1N4003s, which are rated at 1 A and 200-v PIV.

The rectifiers in input rectification should allow for the current used in the supply and for the reverse voltage present on the supply. The 1N400*x* series are most common and are picked based on current requirements. For this design, I use the 1N4003, rated at 1 A and 200-V max reverse voltage.

This design relies on an inductor followed by a capacitor to provide filtering and to add some measure of noise immunity to the system. The inductor is most commonly 10–20 mH and is chosen based on current through it more than anything else.

CAPACITOR

The value of the capacitor is chosen according to the input voltage peak, voltage minimum, power-line frequency, output power, and charging current requirements. Input voltage and input power requirements are given as:

$$P_{in} = \frac{P_{out}}{\text{Efficiency}}$$

$$V_{in} = \frac{P_{in}}{f}$$

where variable *f* represents the line frequency.

We can calculate the value of the capacitor as:

$$C_{in} = \frac{V_{in}}{V_{pk}^2 - V_{min}^2}$$

where C_{in} represents input capacitance in farads and V_{pk} the peak line voltage–rectifier losses.

In addition to the farad value of the capacitor, the switching supply charges and discharges the capacitors with high-frequency current pulses. The time constant to recharge C_{in} from the rectified AC voltage is given as:

$$t_c = \sin\left(\frac{\cos^{-1}\left(\frac{V_{min}}{V_{pk}}\right)}{2\pi f}\right)$$

with V_{pk} defined above a V_{min} of:

$$V_{min} = V_{pk}\cos(2\pi f t_c)$$

As a result, heat is generated in the capacitor. Therefore, the filter caps should have a higher RMS current rating than the RMS AC current as defined by:

$$i_{chg} = \frac{C\,(V_{pk} - V_{min})}{t_c}$$

$$I_{chg} = i_{chg}\sqrt{t_c 2f - (t_c 2f)^2}$$

$$i_{dis} = \frac{P_{in}}{V_{min} D_{max}}$$

$$= \frac{i_{dis}}{2}$$

$$I_{CAP} = \sqrt{I_{chg}^2 + I_{dis}^2}$$

Peak discharge current occurs when the duty cycle is at the maximum level and V_{IN} is at the minimum. To further alleviate the problem, use capacitors with extremely low ESR values.

For a 63-W offline application, V_{min} is set to 99 V and V_{pk} is set at 135 V. These values account for low line levels, the effects of rectification, and losses in the power supply. Line frequency for the U.S. is set to 60 Hz. For an international design, the power input section can be designed for 45–65-Hz operation.

My initial design results in a calculated C_{in} of 122 µF (100 µF + 22 µF}, t_c of 1.94 ms, I_{chg} Of 0.9428 A, I_{dis} of 0.6383 A, and I_{CAP} of 1.138 A. For use with a DC input, a capacitor value of 100–470µF is best.

It's important that the input diodes be able to handle the peak current from the supply for t_c. The ones I've chosen meet this requirement.

WORKING VOLTAGE GENERATION

To operate the control circuitry, I need to generate a working DC voltage in the 9-18-V range. The power required for the working supply is fairly small, so a basic series regulator works quite well. To make sure the required voltage is available at the emitter of the transistor, simply bias the base of an NPN power transistor with a zener diode and resistor pair. Remember, the voltage at the emitter is V_{be} less than the zener voltage and is usually 0.6 V.

One of the drawbacks to a linear regulator is the power requirement in the current-limiting resistor. To lower the current, it must minimize the zener diode current in conjunction with the base current to the transistor. Since the base current is negligible, the zener becomes the most important part of the equation.

A 1N5243 13-V zener meets the needs of my design nicely. The device has a current requirement of only 9 mA.

The bias resistor must supply enough current to turn on the zener diode and also allow the base current I_b to supply load current. For this application, you need approximately 100400 mA on the load. You can determine this by:

$$I_{load} = \beta I_b$$

$$\beta \approx 100$$

$$I_b = \frac{0.4}{100}$$

$$= 4\ \text{mA}$$

Therefore, if I,,,, is 9.5 mA, the resistor must supply 13.5 mA to the base of the transistor.

$$I_r = \frac{V_{in} - V_{zener}}{R}$$

$$R = \frac{V_{in} - V_{zener}}{I_b - I_{zener}}$$

$$= \frac{99 - 13}{0.0135}$$

$$= 6.2\ \text{k}\Omega$$

The resistor should be a 3-W device since the power through the resistor is:

$$P_R = \frac{(V_{in\,max} - V_{zener})^2}{R}$$

$$= \frac{(135 - 13)^2}{6200}$$

$$= 2.4\ \text{W}$$

The bypass cap on the zener prevents some of the noise produced by the zener from transferring to the regulator's output.

Since this noise drains the input power and adds to the total requirements of a 50-W device, power consumption is negligible. It accounts for 4% of the total power used in the system. So, our 47.4-W supply has now become a 49.8-W supply.

The rectifiers and snubber circuits on the output eliminate the missing watts and probably a little more. Normal designs allow for 75–80% efficiency, giving us a total power budget of 63 W.

CONTROL SECTION

The control section is responsible for converting the now-available input DC voltage into a pulsating DC voltage, monitoring the output voltage regulation and any other such jobs as may be assigned. The supply voltage (generated as described previously) supplies this control section as V_{in} and V_c for the switches.

The controller I'm using is the UC-2525 from Unitrode. While not the newest or the fastest device on the market today, this controller is reliable and offers lots of experimentation. Its 1% voltage regulation and wide range of operating frequencies make it ideal for general-purpose design.

To use the provided source, the voltage from pin 16 {V_{ref}} is fed to pin 2. The on-chip reference generator is 5.1 V, which is sufficient for most applications. A supply to drive logic devices and microprocessors can be from 4.75 to 5.25 V. The onboard comparator compares this voltage to the voltage supplied on pin 1 and increases or decreases switching based on the results.

Use a resistor divider if you·prefer to lock the supply at 5 V. Using IO.5 kΩ from pin 16 to pin 2 and 10 kΩ from pin 2 to ground provides a reference voltage of 2.49 V.

If the generated voltage (assuming 5 V out) is fed to pin 1 via a resistor divider using two 10-kΩ resistors, the voltage is extremely close to 5 V. One caveat is that the resistors' tolerance adds to the controller's tolerance, reducing the output voltage regulation characteristic. For most applications, this should not be a problem.

The UC2525 can operate from 100 Hz to 400 kHz. For experimentation purposes, lower-frequency operation is better because most transformers are hand-wound and lossy. Timing capacitor C_t charges through R_t and discharges through R_d.

For this application, I use 5 kΩ for R_t, 0.01 µF for C,, and 220 Ω for R_d. These choices set the frequency for the supply at about 30 kHz. After some practice, you may decide to change the timing and look at the difference in the circuit's operation. The faster the clock rate, the more power the unit should deliver.

Pin 4 provides a place to check the frequency of operation. Pin 3 provides a synchronization input not used in this design.

The compensation network attached to pin 9 provides loop compensation. A 10-kΩ resistor and a l-nF capacitor suffice. The softstart capability of the IC provides a means of slowly starting the supply. A 5–10-µF capacitor to ground works here.

The shutdown pin requires a high level to shut off the device. This can be on a pulse-by-pulse basis-if current mode of control is desired-or it may be a digitally controlled signal. The normal mode of operation is to leave this signal low.

The optoisolator and associated circuitry form a means of controlling this signal from an external source. Supplying a low-level signal to the cathode of the LED turns on the optoisolator, turns off the transistor switch, and supplies a high level to the shutdown pin.

The outputs from the IC are fed to the FET switches through a current-limiting 10-Ω resistor. This signal is then shunted to ground with a 10-kΩ resistor to ensure that the FET is turned off.

PULSATING DC GENERATION

The DC input is now ready to be converted to pulsating DC with the FET switches and transformer. The circuitry includes snubber networks.

The FETs used for switches need to withstand both the switching voltage and the voltage spikes caused by the inductance in the transformer. The basic voltage in a push-pull design is $2V_{source}$. Along with this voltage is another V_{source} in spikes and switching noise.

Here's a good rule of thumb-while prototyping a design, always pick FETs or transistors that handle at least $3\,V_{source}$ across the gate drain or collector emitter.

For our application, a close fit is the International Rectifier IRFP360, which has a V_{DS} of 400 and a maximum on resistance of 0.2 Ω. Resistance is an important factor in keeping the supply cool and efficient.

After proving the basic design, I verify the peak voltage at the drain of the FET and reduce V_{DS} and R_{DS} to the lowest values possible. This reduction usually translates into a less commonly available FET. However, a better FET is worth a longer lead time.

Usually, the FETs have to be mounted on a heat sink. A good practice is to solder a 0. 1- or 0.01-µF cap from the heat sink to ground on the PCB, to assist in reducing EMI.

The snubber networks eliminate noise and ringing from the lines, compensate for transformer leakage inductance, and aid in prolonging the life of the switches. Two types of snubbers are on the primary side of the transformer. One reduces ringing from the switching of the FETs, and the other reduces the effects of leakage inductance from the transformer. To reduce the ringing:

$$C = \frac{I_{Supply} \times \frac{1}{F_{opcrating}}}{2V_{in\,max}}$$

$$= \frac{0.5\ (0.000033)}{270}$$

$$= 68\ \text{nF}$$

To reduce the power in the resistor, I use a 6.8-nF capacitor. Remember, at this point the voltage is empirically set to 405 V, so you need a capacitor rated to at least 400 V. The resistor should discharge the capacitor in one-half the controller's minimum on time:

$$T_{on\ min} = \frac{D_{max}V_{in\ min}}{fV_{in\ max}}$$

$$= 12\ \mu\text{s}$$

Our minimum time is 12 us. This time requires a resistor value of:

$$R = \frac{T_{on\ min}}{2C}$$

$$\frac{12\ \mu\text{s}}{13.6\ \text{nF}}$$

$$= 1\ \text{k}\Omega$$

The power through the resistor is:

$$P_R = \frac{CV_{in\ max}^2 f}{2}$$

$$= 1.8\ \text{W}$$

Use a 2-W device in this application. As the frequency of operation goes up, so does the power dissipated in the snubber network; keep an eye on this increase. A tradeoff can be made in the snubber, as you can see by the capacitor change. As you decrease the capacitor, the resistor's power rating and value change.

The diode in this snubber reduces spikes. Use a fast-recovery diode with a PIV of 400 V. The snubber across the transformer, RC between FET drains, is to reduce harmonics caused by the leakage inductance in the transformer. Unless you know the value for this inductance, a 0.047-µF, 400-VDC capacitor and a 68-100-R, 3-W resistor should be used.

TRANSFORMER DESIGN

Now that the control circuitry is designed and in place, I need to wrap the transformer. The transformer is straightforward-it's based on the ratio of input voltage to output voltage:

$$n = \frac{N_p}{N_s}$$

$$= \frac{0.9D_{max}(V_{in\ min} - V_{loss})}{V_{out} + V_{loss}}$$

$$= 7.27$$

$$N_s = \text{Integer}\left(\frac{N_p}{n}\right)$$

$$\frac{40}{7.27}$$

$$= 6$$

Recalculate N_p based on $N_s = 44$.

To get all the needed outputs, the transformer windings should be as follows: 44 for V_{in}, 6 for +5, 20 for f12, and 15 for ±8. To allow for DC isolation, place three layers of 1-mil mylar tape between the primary and secondary windings. First, wrap the first half of the primary, tape and wrap the first half of the 5-V secondary as well as the first half of the 8-V secondary. Complete 12-V secondary before completing the second half of the 8 V and the second half of the 5 V. Finally, tape the rest of the primary.

This procedure ensures maximum conduction into the 5-V, 8-V, and 12-V line. Many texts and design examples show optimum winding performance and how different winding methods affect the output. Feel free to experiment and try other arrangements.

OUTPUTS

You should rectify and filter the output of the transformer to provide a smooth DC voltage out. The rectifiers used in the output should be fast recovery diodes. I used the HER105 here.

The output inductor is based on the maximum off-time of the switch:

$$D_{min} = D_{max}\frac{V_{in\ min}}{V_{in\ max}}$$

$$= 0.5\frac{99}{135}$$

$$= 0.367$$

$$T_{off\,max} = \frac{1 - D_{min}}{f}$$

$$\frac{1 - 0.367}{30000}$$

$$= 21.1\ \mu\text{s}$$

The inductance required to prevent discontinuous mode of operation is based on the minimum load current:

$$\Lambda I\,L_{max} = 2\,I_{o\ min}$$
$$= 2\,(5)$$
$$= 10\,A$$
$$L = \frac{(V_o + V_f)\,T_{off\,max}}{\Lambda I\,L_{max}}$$
$$\approx 10\,\mu H$$

The capacitance you need to maintain a specified ripple voltage is:

$$C_{out} = \frac{\Lambda I\,L_{max}}{8fV_{ripple}}$$
$$= 416\,\mu F$$

The closest standard value is therefore 470 µF.

In addition to the farad value, the maximum ESR is defined as:

$$C_{ESR} = \frac{V_{ripple}}{\Lambda I\,L_{max}}$$
$$= \frac{0.1}{10} = 0.01$$

To get this ESR, you need a capacitor of incredible value. In this design, I use four 1000-µF capacitors wired in parallel.

The rest of the outputs are filtered in much the same manner. Since the ±8-V line is to be used in analog circuitry, I use post regulators (78xx, 79xx) to provide the necessary regulation.

CONCLUSION

While this article is far from exhaustive, I've tried to shed some light on power supply design and to provide guidelines to help you design your own. I hope I've given you the knowledge and courage you need to experiment with switching regulators.

The benefits-picking and choosing outputs, adding the power supply to the main circuit board, and reducing overall costs-far outweigh the time you spend learning to do it yourself. The average cost of the custom supply described here is about $30 in small quantities.

After you've played with the circuit for a while and demystified the operation, small switching supplies will be a standard part of your design notes, not something left for the purchasing department to find. ❑

Edward Schram is an engineering consultant specializing in microprocessor-based hardware design and coding of CCITT/ITU protocols for the commercial telecommunications industry. He may be reached at ecs@ecsd.win.net.

CONTACTS

Unitrode Corp.
7 Continental Blvd.
Merrimack, NH 03054-0399
(603) 424-2410

International Rectifier Corp.
233 Kansas St.
El Segundo, CA 90245
(310) 322-3331

IRS

404 Very Useful
405 Moderately Useful
406 Not Useful

Energy Management in Motor Control

FEATURE ARTICLE

Michael Rosenfield

The MTE1122 controller staves off the energy-consuming monster—namely, the electric motor. By matching the capability of the motor to workload, the processor reduces electricity waste up to 58%.

electricity is the lifeblood of our entire industry and the heart of every electronic product. The source of that energy is limited, and while we are developing thousands of new ways to consume more energy every day, our efforts to conserve it need dramatic enhancement.

The electric motor is the energy monster. According to the Department of Energy, more than 50% of all energy produced in the United States is gobbled up by fractional-horsepower electric motors. More important is the estimate that up to 20% is consumed when no useful work is being accomplished.

One of the most popular types of electric motors is the AC single-phase induction motor. It runs most efficiently at full load. As the applied load lessens, a greater portion of the energy consumed by the motor is wasted since the power it consumes does not decrease proportionately.

From garage-door openers and washing machines to the industrial machinery used to produce them, there are millions of AC induction motors in the world today. Think about the potential energy savings that can be realized and the number of power plants, both fossil-fuel and nuclear, that don't have to be built if energy is used more efficiently.

This article discusses a solution to the energy-management problem. The MTE-1122 Energy Management Controller IC [see Figure 1 for a system block diagram and Table 1 for its features) is at the heart of an energy-management system which provides benefits in two vital ways.

By efficiently and continuously matching the capability of the motor to the variations in load demand generated by the host machine, the MTE-1122 reduces the motor's energy consumption by up to 58%. It achieves this feat by monitoring the phase difference between the voltage and current waveforms, calculating the load on the motor, and adjusting the voltage supplied to the motor. The energy-management controller makes these calculations thousands of times a second, enabling it to vary the voltage supplied to the motor, thus controlling its power demands.

In addition to the direct cost savings achieved, another significant benefit created by this controller is a reduction in the motor's operating temperature by up to 9%. Generating less heat results in both increased motor life and system reliability.

This article describes an electronic control circuit capable of improving the electrical efficiency of induction motors, especially at low load levels.

Why are motors specified that are more powerful than necessary? There are perhaps three major reasons:

- overspecification-sometimes it is easier or costs no more to specify a larger motor than to determine actual loads

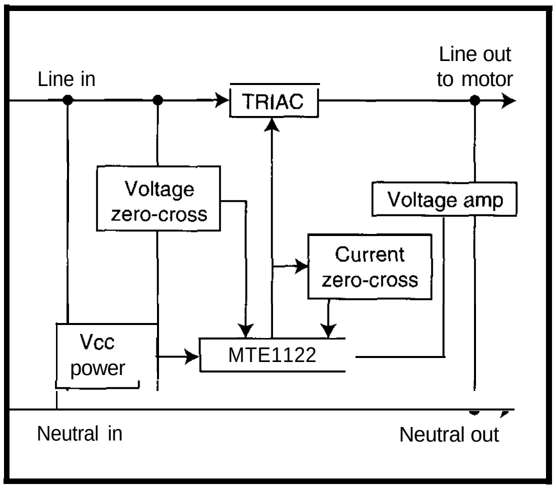


Figure 1-A complete electric *motor* energy management system consists *of the MTE1 122 at its core with some support circuitry around it.*

- worst-case design-some systems are specified at a particular load, but operate much of the time with lower loads
- idle time-many times, systems can't be conveniently shut down when not in use.

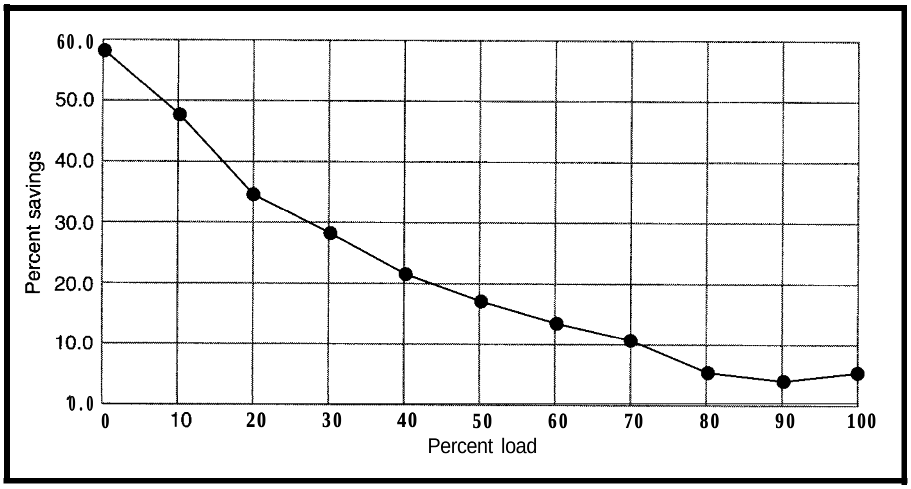
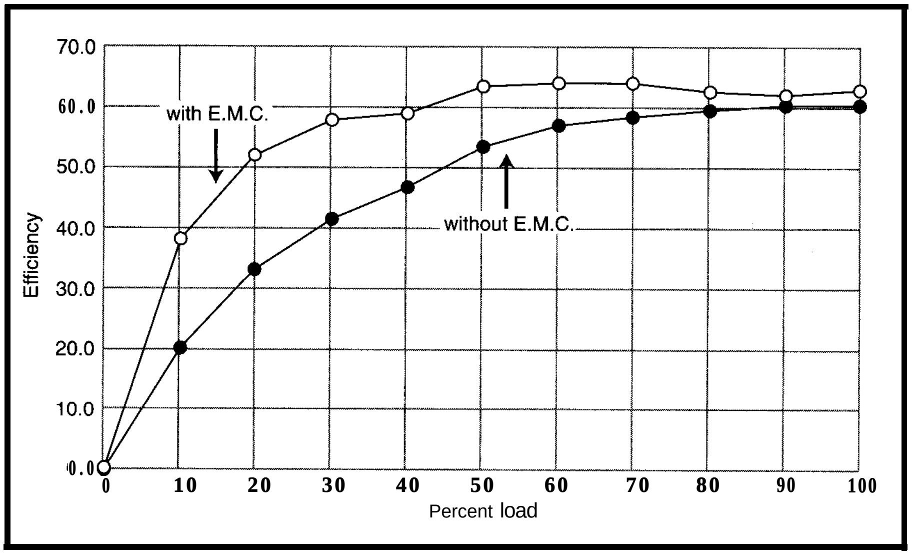


Figure 2—*Energy savings is a function of motor loading using the MTE1122. Energy savings is greatest at low loads.*



gure 3—*Motor efficiency plotted as a function of motor load with and without the MTE1122 again shows the savings possible.*

Overspecification can be corrected by proper design. For example, the compressor systems in modern residential refrigerators have been optimized quite effectively. These systems have fairly constant and well-characterized loads.

In contrast, most industrial motorized systems fall into the worst-case design category. Such systems include air movers, commercial refrigeration compressors, water pumps and circulators, laundry equipment, conveyor systems, and machine tools.

For instance, consider the drill press. A typical press has a motor capable of drilling a 1" hole in steel. But if the user primarily drills ½" holes in wood, the motor won't be working very hard.

One way to improve the efficiency of induction motors used in such applications is the MTEl122, developed by Microchip Technology.

The MTEl122 is an energy-management controller IC for single-phase induction motors. This CMOS device is based on Microchip's RISC processor core and proprietary algorithms. When combined with external analog components, it provides an electronic system that economically reduces the operating costs of small induction motors by as much as 58%. It also enables motors to run cooler and with less vibration. The system operates on single-phase 110 or 220 VAC.

The MTEl122 calculates the amount of load on a motor connected to it and adjusts the motor's supply voltage to match the load. For example, if the load is lower than the motor's rated load, the voltage to the motor can be reduced, thus decreasing the energy used by the motor.

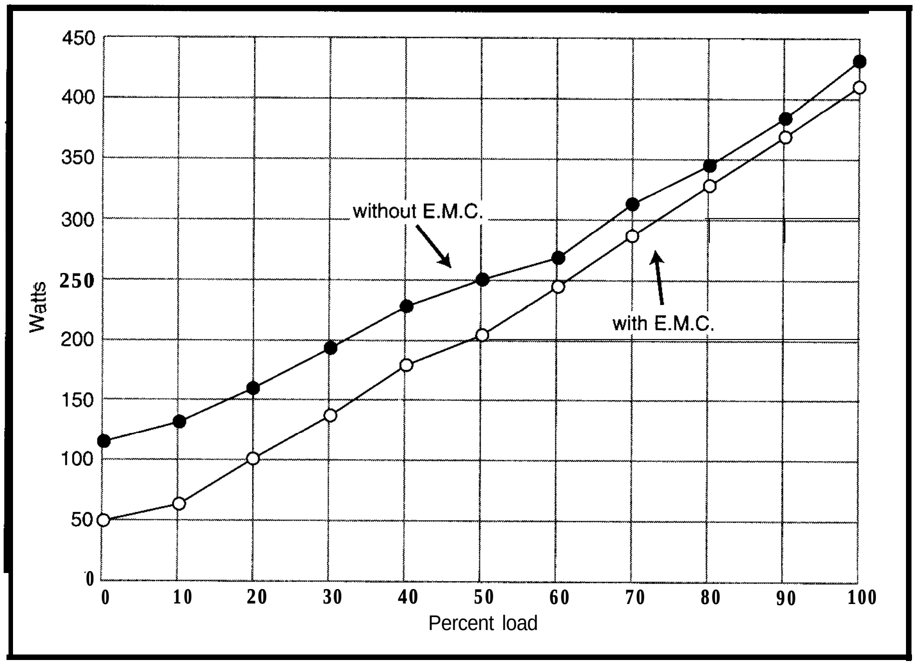


Figure 4—*Motor power draw is as a function of motor load, and the MTE1122 helps save some power.*

ENERGY SAVINGS

Reducing the voltage to the motor cuts its power draw. A ⅓-hp motor typically sees 85 VAC at no load when powered through the **MTEl122,** for an energy savings of as much as 58%.

The test motor was attached to a dynamometer (see Energy Measurement sidebar). The load on the motor was adjusted in 10% increments, from 0 to 100% of rated load. RPM, current, and power measurements were taken at each step. Motor efficiency and energy savings were calculated from these data. While the motor speed decreased about 1% more rapidly with the MTEl122 controller in place, the power produced by the motor was constant. The results are plotted in Figures 2–4.

In addition, a M-hp 110-/220-V motor was tested. Its energy savings are shown in Figure 5. It's interesting to note that this particular motor is even more efficient at 220 V than at 110 V.

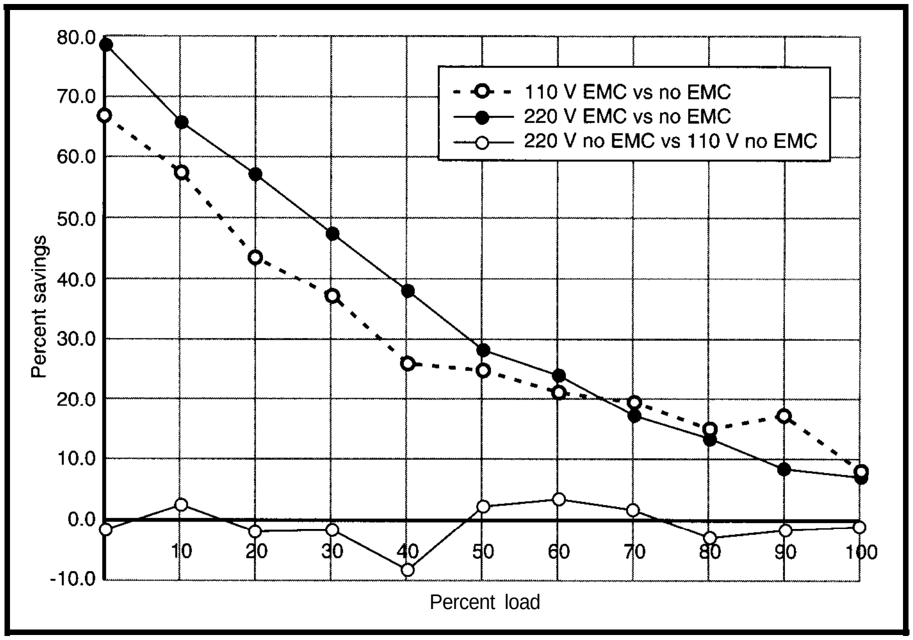


Figure 5—*Energy* savings *is a function of motor loading using the **MTE1** 122 (both **110- and 220-V results** are shown).*

At no load, a ⅓-hp test motor dissipated 120 W, primarily as heat. With the MTEl122 managing the power load, dissipation drops to 50 W, a savings of 58%. At full load, the figures are 428 W and 406 W, respectively, for a savings of 5%.

Several copies of this circuit were built up and used to control the power to a ⅓-hp motor attached to a ⅓-hp dynamometer. Figure 2 shows the energy savings obtained. Motor efficiency with and without the controller is shown in Figure 3.

Actual savings may vary based on motor size, load, and construction.

CIRCUIT DISCUSSION

The circuit uses low-cost, readily available components. The MTEl122 is available from any Microchip distributor. Note that V_{cc} is supplied directly from the AC line without the need for a transformer. Component values have been calculated to work on 1 10- or 220-VAC lines, with motor-current draws of up to 15-A RMS continuous. This translates to l-l.5 hp at 110 V and 3 hp at 220 V.

This system only works with rotating inductive loads (i.e., motors) that are not otherwise power-factor corrected. Capacitor-run motors don't function with this system, nor does fluorescent lighting. Universal motors (brush-type) do not benefit from the system either. While this system usually saves energy, lightly loaded motors obtain the greatest savings.

For best results, systems with electrical devices in addition to motors (such as appliances) should power those devices directly from the line, not through the energy management control system.

Also, note that each motor must have its own control circuit (unless the motors are never activated at the same time).

The circuit can be laid out on a single- or double-sided board, observing the standard layout techniques used with monolithic microcontrollers. The triac requires heatsinking, the size depending on the motor current draws and ambient temperatures. The distance between the motor and the MTEl122 circuit isn't critical.

Be sure to follow standard electrical construction practices.

CIRCUIT DETAILS

Figure 6 shows a suggested circuit implementation. The MTEl122 (U3) consists of a high-performance 8-bit microcontroller with embedded proprietary algorithms (see Table 2 for pinout descriptions). It monitors the voltage across the motor (Ul), the voltage zero-crossing (Q2), and the current zero-crossing (by monitoring the signal on gate of Q3). By measuring the time between voltage and current zero-crossings, the MTE1122 calculates the amount of load on the motor.

U1 and R10–13 form a differential amp with a gain of 1/48. C4 limits noise

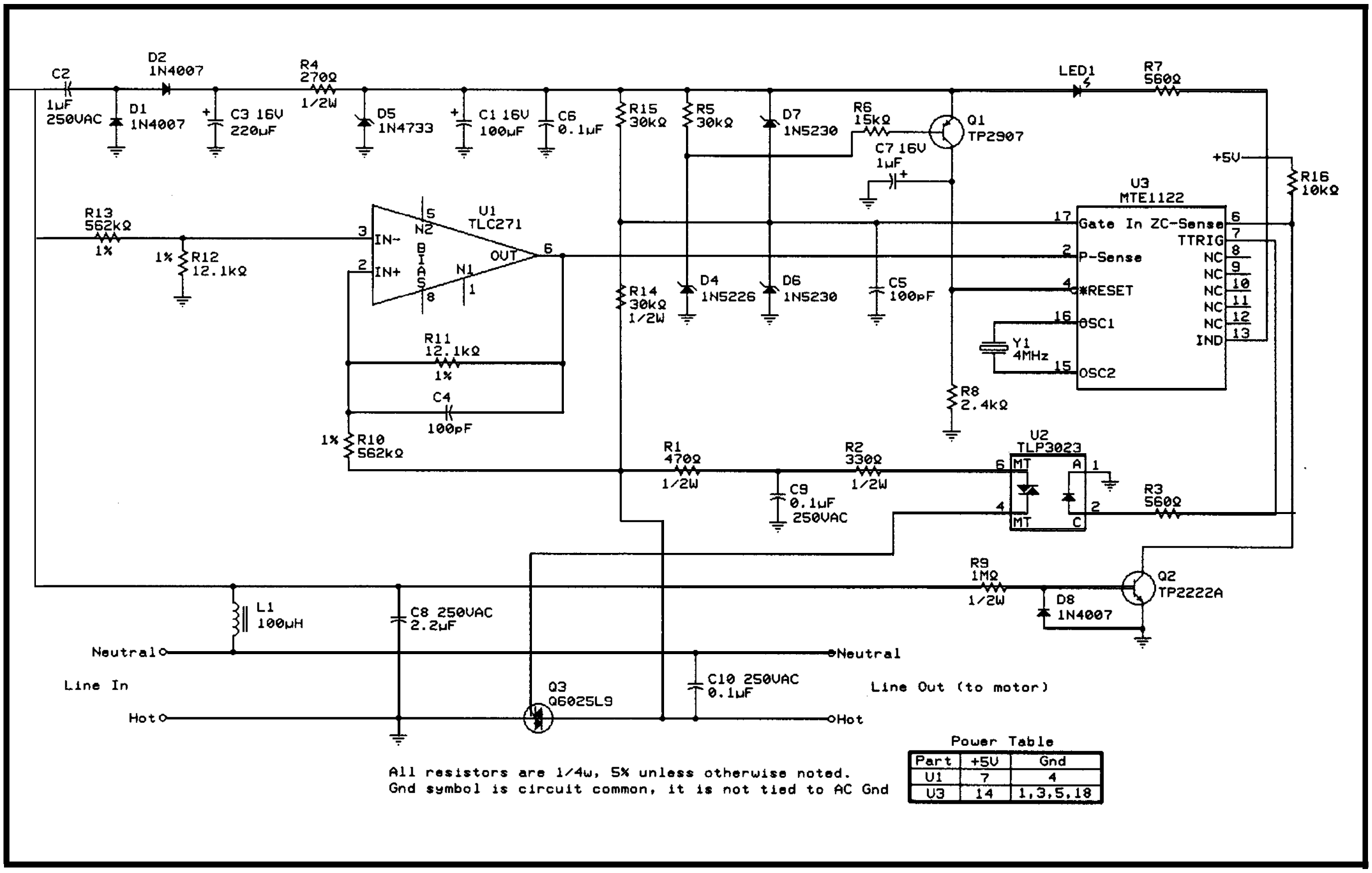
But What about Three-Phase?

Many of the industrial motors used by industry are three-phase induction motors. How do they compare with their single-phase cousins?

A three-phase motor is more efficient because it has a rotating 360-Hz field instead of a pulsing 60-Hz field. For this reason, it also requires no starting windings. However, it is still inefficient at low loads.

As of this writing, a number of 10-, 20-, and 50-hp motors are operating successfully (and efficiently!) using three of the single-phase circuits described in this article, one wired across each phase leg. Microchip is running tests on a sawmill, several refrigeration compressors, and several punch presses, to name a few.

The results are encouraging. Hard performance numbers should be available by the time this article reaches publication.

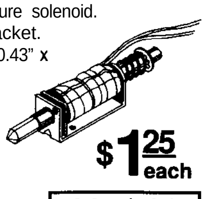


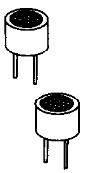
Power Table

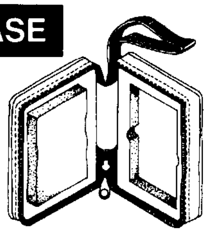
Part	+5V	Gnd
U1	7	4
U3	14	1,3,5,18

All resistors are 1/4w, 5% unless otherwise noted.
Gnd symbol is circuit common, it is not tied to AC Gnd

Figure 6—*By using the MTE1122 at the core of the energy management circuit, complexity, size, and cost are kept to a minimum.*







sensitivity. C2–C6 and components in between rectify and filter line voltage to provide V,,. L1 and C8 also filter the line voltage. Q1 and C7 provide powerup reset for the MTE1122.

U2, in an optotriac, triggers the power triac. This triac should be sized to the current draw of the motor. For smaller motors, a smaller triac can be used. For larger motors, use a larger triac or back-to-back SCRs. Any triac whose triggering characteristics match those produced by U2 is fine. Keep in mind that the triac performs better if it is not used at its current limit. Of course, heat sinking must be provided, based on load and temperature.

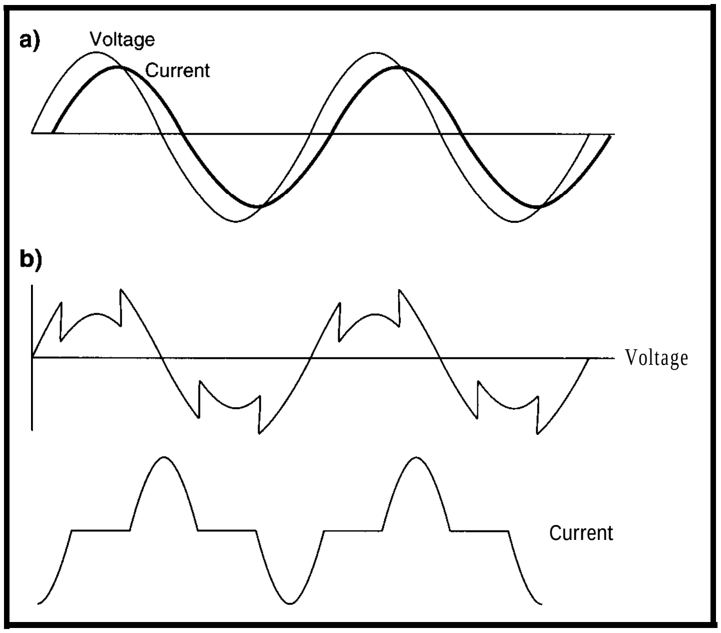


Figure 7-(a) *Without the MTE1122, current lags the voltage. (b) With the MTE1122 in the circuit, voltage and current are reduced as the load decreases.*

D3 and R7 indicate normal operation of the circuit. Both are optional.

As stated, you can lower the energy consumption of a motor running only partly loaded by decreasing the current flowing into the motor windings. This decrease in current can be achieved by lowering the voltage across the motor windings.

If the voltage isn't increased when the motor load increases, the internal reactance of the motor decreases, the windings draw too much current, and they may overheat and be damaged. Because the MTEl122 is an intelligent controller, it can monitor motor voltage and load. It makes corrections within 8 ms—well before there is any potential for motor damage.

THEORY OF OPERATION

In an induction motor, the current draw at no load is quite high because the stator windings supply all the magnetic field energy for the rotor. As a result, the motor draws a major portion of its full-load current even when idling. The energy not converted into work becomes waste heat and vibration, which shortens the life of lubricants, bearings, and other nearby components.

The torque produced by an induction motor is proportional to the square of the applied voltage. Thus, a motor producing part of its rated load only needs part of its rated voltage.

As Figure 7 illustrates, the current in the windings lags the voltage in an induction motor, due to the inductive reactance in the windings. The cosine of the amount of lag in degrees is the power factor. Power factors are 1.0 for resistive loads such as heaters. They can vary from close to 1 for a fully loaded motor to as low as 0.1 for an idling motor. The actual power being consumed by the motor is:

Voltage x Current x Power Factor

- based on 8-bit RISC technology
- proprietary power management algorithm
- reduces power consumption of induction motor systems
- 5-V operation
- 18-pin PDIP and SOIC packages
- 8-bit A/D converter
- automatic power-on reset
- powerup timer
- commercial and industrial temperature range operation

Table I--The *principal design features of the energy management controller make it idea/ for embedded applications.*

A lightly loaded induction motor has low power factor. As the motor reaches its rated load, its power factor gets closer to 1. How close it gets to 1 depends on the motor's internal design. Values of 0.65-0.75 are typical of loaded single-phase motors.

The MTEl122 calculates motor loading by measuring the time between current and voltage zero-crossings-in effect, power factor. When the load on the motor is low, lowering the voltage on the motor decreases power consumption. You can lower the voltage by turning a triac on at the proper time during the voltage cycle.

The proprietary algorithms in the MTEl122 monitor the voltage across the motor and the zero-crossing times and adjust them on a cycle-by-cycle basis. At no load, the voltage to the motor can be as low as 85 VAC instead of the usual 120 VAC. Also, note that the current doesn't need to flow continuously. Power consumption can be cut as much as 58% (depending on load) and operating temperature lowered by as much as 45°F.

A motor powered by the MTEl122 and this energy-management control circuit draws less average current. Its power factor is also improved. However, the power factor seen by the line is *not* improved.

RESULTS

Well, so much for theory. How does this device work?

Since energy savings vary with motor load, and motor load depends on machine design, this is a complicated question. Let's look at some real-world examples.

I performed a series of tests on a fairly old washing machine. Energy sav-

Energy Measurement

To measure the true power of an induction motor, you need a true-RMS power meter, one that measures non-sinusoidal waveforms. Models of this type of instrument are available from Fluke, Amprobe, and Tektronix, among others.

I used the meter and current probe to measure the voltage, current, and true-RMS power supplied to the Energy Management Control System driving a ⅓-hp motor. A phototachometer measured the motor RPM.

The torque supplied by the motor was measured with a dynamometer, which also supplied the adjustable load on the motor. The dynamometer consists of a DC generator whose armature is coupled to the motor under test (see Figures i and ii). The generator output goes to a load bank, and its field winding [i.e., case) is attached to a force-measuring scale.

By increasing the electrical load on the generator, you increase the coupling between the armature and field. This increase causes a drag on the motor that is opposed by the generator's case (which contains the field windings). The case is free to swivel, and the force on the case is measured by the scale.

Motor Power Out (in W) is calculated by:

$$P = \frac{\tau n}{5.18}$$

Power Out (in hp) is calculated by:

$$P = \frac{\tau n}{7142.72}$$

where τ is in Newton-meters and n is RPM.

Efficiency (in %) is calculated by:

$$\frac{(\text{PowerOut})}{(\text{PowerIn})} \times 100$$

The equipment I used to make the measurements for this article is listed in the reference section.

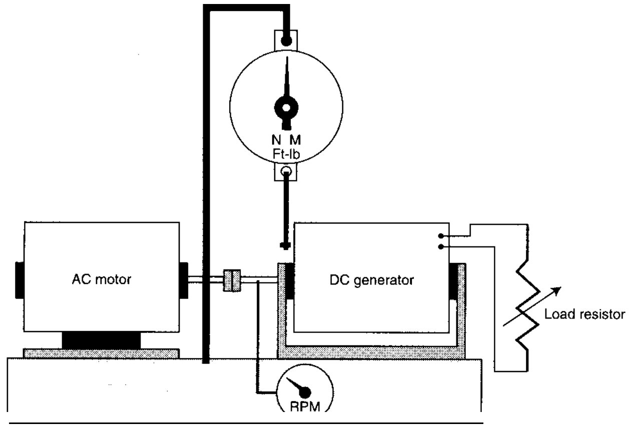


Figure i-A side-view sketch *of a dynamometer shows the motor under test.*

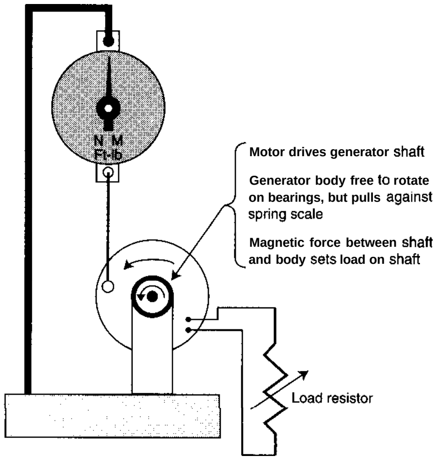

N M
Ft-lb
Motor drives generator shaft
Generator body free to rotate on bearings, but pulls against spring scale
Magnetic force between shaft and body sets load on shaft
Load resistor

Figure ii-An *end-view sketch illustrates how the dynamometer works.*

ings varied from 9% during the wash cycle to 45% during the spin cycle. For a family of four, figuring a typical amount of laundry (say 8 hours per week) at $0.105 per kWh yields an estimated $4.50 in savings per year.

While these savings are significant, they are certainly not earth-shattering, considering a cost of about $35 for the energy-management circuitry. For a larger family or a commercial washer running 40 hours per week, the device pays for itself in a year. That's a good return on any investment.

I also ran tests on an air-handler (house air-conditioner fan). On high speed, 4% savings were realized and on low speed, 10%. This type of device has a much higher percentage of on-time than a washing machine. The fan used a ⅓-hp motor, and saved 20 and 30 W, respectively. With 50% run-time, you save $13 over the course of a year on low speed.

Though they were not tested, heater fans and circulator pumps are also likely to benefit from this technology. Pool and hot tub pumps are another likely beneficiary. And, of course, many machine tools operate long hours each day with only partial loads. Again, the more hours per day a machine operates, the quicker the MTE-1122 Energy Management Control pays for itself.

ALTERNATIVE APPROACH

You can also increase motor efficiency by adding another winding to the motor and phase-shifting it with capacitance. This step produces a *capacitor-run motor* with a power factor of 0.9 or better (regardless of its load) and considerably lower idle power consumption. It is a more efficient motor and produces less vibration.

This approach, however, isn't cost-effective in motors less than 1 hp nor

P-Sense	analog input used by the device to measure load voltage
Gate Enable	analog input that monitors voltage across the triac. It is used as a current feedback mechanism.
IND	TTL-compatible output that indicates normal system operation. It is intended to control an LED or another indicator device.
ZC-Sense	TTL-compatible input used to determine the zero crossing point of the AC voltage waveform
TTRIG	TTL-compatible output used to drive the triac
RESET	TTL-compatible input used to reset the device by holding this pin low
OSC1, 0SC2	Oscillator crystal or resonator connections

Table 2—The pin functions of the MTE1 122 include a mix of analog and digital signals.

in motors for residential use. Thus, for lowest-cost approaches, the MTEl122 and its associated circuitry are probably the best method for improving motor efficiency.

GREEN MOTOR FUTURE?

The MTE1122 is the heart of a digital energy-management controller that provides substantial power-consumption savings for AC-induction motors. Because of its low component count and small board design, it can be integrated into industrial equipment and consumer appliances that use AC-induction motors. Applications requiring energy management are numerous.

Clearly, the world will require more efficient use of its electrical energy as we approach the next millennium. Energy management will become a necessity. ❑

*Michael Rosenfield is a principal engineer for product development at Microchip Technology. He has more than **15** years of design engineering and production experience, working previously for Systematic Computer Services. He may be reached at (602) 786-7494.*

SOURCES

MTE1122
Microchip Technology
2355 W. Chandler Blvd.
Chandler, AZ 85224-6199
(602) 786-7200
Fax: (602) 917-4149

CSM-100 ⅓-hp motor, DYN-100A Dynamometer, Rl-100A Load bank, HPT-100 Digital Photo Tachometer
Hampden Engineering Corp.
P.O. Box 563
East Cone Meadow, MA 01028
(413) 525-3981

THM560 Scopemeter, A622 Current probe
Tektronix Corp.
P.O. Box 1000
Wilsonville, OR 97070-1000
(563) 627-7111

I R S

407 Very Useful
408 Moderately Useful
409 Not Useful

#119

DESIGN CONTEST WINNER

FEATURE ARTICLE

Ken Pergola

The Micro-bRISC Device Programmer

Tackling Microchip's Midrange Arsenal of PICs

PIC development tools abound—but their price is often prohibitive. Here, Ken introduces us to the hardware and software aspects of a low-cost, high-performance PIC programmer.

There's no denying that Microchip's PICs have spawned a renaïssance in microcontrollers. Because of their wide availability, low cost, and high performance, many engineers have been introduced to embedded programming and design with Microchip's RISC-based PIC microcontrollers. Although there are many PIC development tools on the market, their high cost prohibits many developers from plunging in.

Unlike microprocessor projects, simple microcontroller projects can be developed quickly and without overwhelming beginners with address decoding, memory and I/O interfacing, and bus concepts. It's also important to learn how to use microcontrollers in cost-sensitive designs.

I developed Micro-bRISC for low cost, high performance, and entry-level users. In this article, I'd like to trace through the development of Micro-bRISC, examining what enables it to successfully program midrange PICs.

THE GENESIS OF Micro-bRISC

Initially, I experimented with Microchip's PICSTART- 16B PIC device programmer. Although the PICSTART-16B uses a parallel programming algorithm for PIC16C5*x* devices, it uses a serial algorithm for PIC16C*xx* devices.

By using the signals as defined in the Microchip serial programming specification, I successfully programmed the PIC16C71 and PIC16C84 with just five wires tapped from the PICSTART-16B ZIF socket. I was elated, believing that it would therefore support future devices such as the PIC16C64 and '74.

However, a call to Microchip proved me wrong. While it was theoretically possible to support the new microcontrollers with the PICSTART-16B, a new board, the PICSTART- 16C, was in the works to support the new devices. However, it was then a long time from release.

Armed with Microchip's PIC16C6*x*, '7x, and '84 programming specifications, I developed my own PIC16C64 programmer. My goal quickly changed to support any PIC, present and future, that conformed to these programming specs. After much coding, hardware refinements, and name changes, Micro-bRISC was christened.

IT'S ALL IN THE NAME

Micro-bRISC is a parallel port-driven programming board that programs the PIC16C*xx* midrange PIC families. Currently, it supports the PIC16C64, '71, '74, and '84 microcontrollers. Its firmware and software are open-ended, so supporting future '6x, 7x, and '8x controllers requires only a software-not firmware-upgrade.

Micro-bRISC is primarily intended for Microchip's powerful MPASM universal assembler (available on Microchip's BBS). Micro-bRISC's HEX and EXE files can be downloaded from the Circuit Cellar BBS.

Those who already own a PICSTART- 16B programmer can use Micro-bRISC to augment its features. The PICSTART-16B and the Micro-bRISC programmers, both classified as development or prototype programmers [versus production quality), coexist peacefully on the same system since they use the serial and parallel port, respectively.

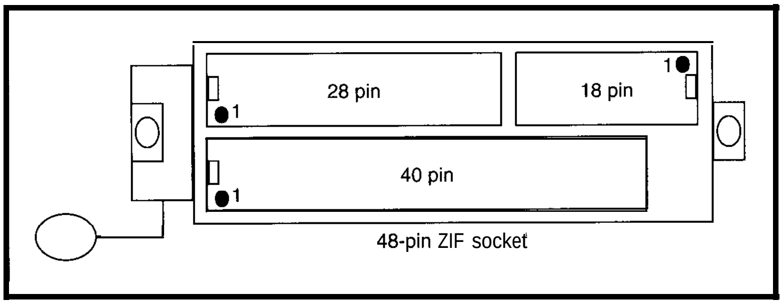


Figure I--The *Micro-bRISC 48-pin ZIF socket accommodates three* different *PIC package sizes. Currently, this arrangement supports 1.5 PIC* devices! *The* drawing *is* exaggerated *to show clarity. Of course, all parts straddle the middle of the* socket.

The Micro-bRISC's strengths include its:

- low-cost
- extremely fast programming times
- ease of construction and single-chip design
- support of PIC16C64, '71, '74, and '84 and future support of '61, '62, '620, '621, '622, '63, '65, '70, '72, '73, and '83
- easy-to-use mouse- or keyboard-supported GUI software
- PC-hosted parallel port interface via standard Centronics printer cable
- support of LPT1, LPT2, and LPT3 via automatic detection
- support of bidirectional ports for higher programming throughput
- use of AC or DC power supply
- compatibility with Enhanced Parallel Ports (EPP)

PARALLEL VERSUS SERIAL

Although there are tradeoffs in using either interface method, I chose the parallel port over the serial port for high-speed data transfer and low parts count.

An asynchronous serial interface is easier to implement but much slower. At high data rates such as 115,200 bps, providing proper communication would be challenging. In addition, not all PCs can sustain a serial transfer rate that high. I would be restrained to a serial transfer rate that all PCs could accommodate. Faster PCs would be shortchanged.

Although it required more I/O pins, the parallel port's high throughput made it a more enticing interface medium. It also enabled me to use just one chip in the design, whereas a serial interface would have required an extra chip, such as the MAX232, for the TTL and RS-232 voltage translations. To fully take advantage of the parallel port's speed, I had to choose my target processor carefully.

Since the size of program memory in Microchip's microcontrollers is continually growing, the need to shuttle data around quickly was obviously fundamental. For example, while the '71 and '84 have 1K words (14-bit words) of program memory, the '64 and '74 use 2K and 4K words of program memory, respectively.

Since a PIC word is 14 bits wide, an RS-232 programming link can transfer two bytes for each word. To program the '74, 8192 bytes would be sent to the target board. For the device to verify, the target would send 8,192 bytes to the PC. At 9600 bps, the program and verify functions would need 16,384 bytes and 17 s to transfer. (This estimate doesn't include the overhead of the programming algorithm's program delays, overprogram delays, or target board's firmware execution time.)

Since Micro-bRISC can accomplish all this with overhead in 4 s, I feel I was correct to choose the parallel port. Due to its 13-bit wide program counter, the program memory space ceiling for the PIC 16Cxx midrange devices is 8K words.

BREAKING THE BOTTLENECK

Having chosen the parallel port as the I/O link, I decided to add S-bit wide bidirectional transfer capability. By using I/O cards on the market, I implemented a true bidirectional parallel port. Completely transparent to the user, standard and bidirectional ports are automatically detected and used.

If a standard (nonbidirectional) parallel port is detected, Micro-bRISC receives data from the PC in byte-sized chunks and transmits to the PC in 5-bit chunks (3 bits for data and 2 bits for handshaking signals). Micro-bRISC sends five separate transfers to transmit a whole PIC word (14 bits wide) to the PC.

However, if a bidirectional parallel port is detected, Micro-bRISC both sends and receives byte-wide data, thus using the parallel port to its fullest potential. Therefore, transmitting and receiving PIC words to the PC only takes two separate data transactions.

The following list illustrates the extremely quick programming cycle times (device blank check, program, and verify) obtained on a '486DX 33-MHz computer with a bidirectional parallel port:

- '71 [1K word device]: 1 s
- '64 [2K word device]: 2 s
- '74 [4K word device]: 4 s
- '84 [1K word device]: 12 s

Faster computers should see higher programming yields than slower computers.

Obviously, an EEPROM-based microcontroller such as the '84 takes longer to program because its present programming specification calls for a longer programming period per EEPROM location. In fact, it requires 10 ms for each of its 1024 EEPROM program memory locations, 64 EEPROM data locations, and 7 configuration memory words, thus requiring 10.95 s for just the programming delays. The actual programming time is also greater due to the firmware programming algorithm overhead and the parallel port I/O data transfers.

MAXIMIZING MINIMALISM

One of the earliest Micro-bRISC prototypes used three TTL buffers to provide the high-current driving capability for interfacing to the parallel port. I didn't want users to be restricted to an extremely short cable due to lack of proper signal buffering.

Then I remembered one of the strong suits of PICs—their high-current sink and source capability. After careful consideration and testing, I removed the two 74HCT541 buffers from my original design and let the target PIC16C57 flex its muscles instead.

The final TTL buffer chip was a 74HCT241, which steered the 4-bit-wide data or handshake signals into the status register of the parallel port. I eventually dropped it to achieve a one-chip design. Although data throughput on standard parallel ports slightly decreased, the completely different protocol used in byte-wide bidirectional transfers meant bidirectional port performance was not sacrificed.

My first instinct was to use two ZIF sockets-one for 40-pin and one for 18-pin devices. Since the sockets represent a good portion of Micro-bRISC's total cost, I needed a cheaper solution.

Then, it became apparent to me—why not use a 48-pin ZIF socket as a universal socket? Although it sounded easy, it involved strategically locating a 40- and 18-pin device within the 48-pin ZIF socket without causing V,,, V_{pp}, and signal contention in the 5-pin overlap.

When Microchip announced its 28-pin devices, I had to fit three different PIC sizes into a single ZIF socket without any hardware-assisted pin-switching techniques. Fortunately, Microchip's pin layouts are consistent. I was able to forgo analog pin-switching techniques that complicate hardware design and possibly degrade programming signal levels.

As you can see in Photo 1 and Figure 1, I found the only viable alternative. The 28- and 40-pin devices are oriented right side up and flush left in the ZIF socket (closest to the lever). Conversely, the 18-pin devices are oriented upside down and flush right.

With this placement, three PIC DIP sizes can be programmed successfully without adverse effects from the various voltages and signals that exist in the overlap area. This arrangement works very well after you get used to placing the 18-pin devices upside-down.

Since most users have a printer, I wanted them to be able to use a standard Centronics printer cable with Micro-bRISC. I incorporated a female Centronics socket connector in the Micro-bRISC target board. Even though it was more expensive than a DB25 socket connector, the extra cost is minimal compared to the cost and annoyance of buying a nonCentronics parallel port cable.

GETTING CARDED

I tested two bidirectional parallel port I/O cards with the Micro-bRISC, one from Essential Data and the other from Dalco Electronics. Both cards worked flawlessly.

Working with bidirectional parallel ports is extremely easy-1 now loathe working with standard, nonbidirectional parallel ports. However, since most PC users have standard parallel ports, it was imperative that the Micro-bRISC support standard parallel ports as well. Various standard parallel port cards were tested with the Micro-bRISC with excellent results.

Now that enhanced parallel port cards are becoming mainstream, I also tested the Micro-bRISC with an IEEE 1284-compliant EPP/ECP I/O card (F/PortPlus) by FarPoint Communications. Although the Micro-bRISC doesn't conform to the EPP/ECP protocols, the two worked together beautifully in the backward-compatible bidirectional Centronics mode. The IEEE 1284 standard ensures compat-

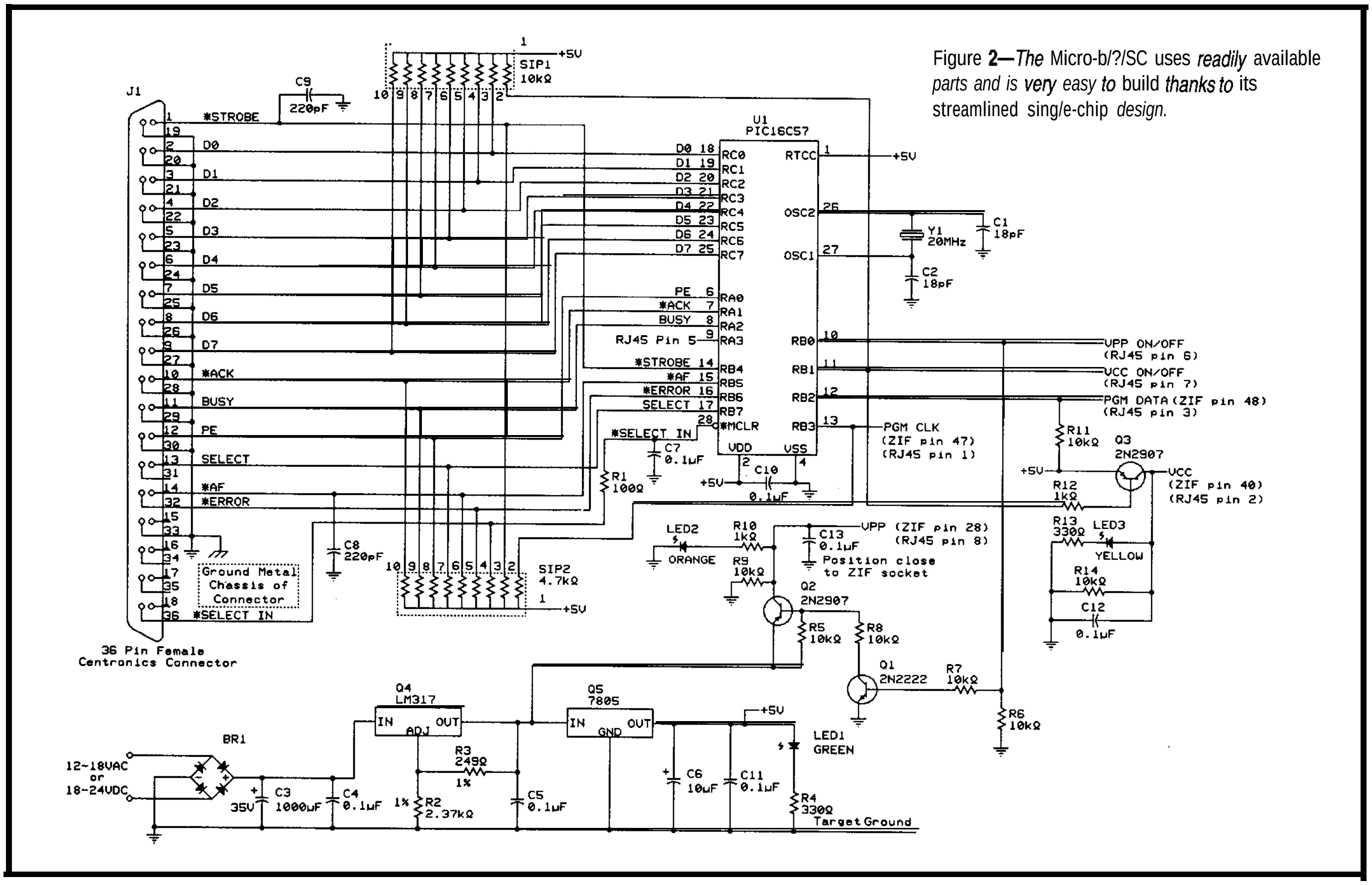


Figure 2—The Micro-b/?/SC uses readily available parts and is very easy to build thanks to its streamlined sing/e-chip design.

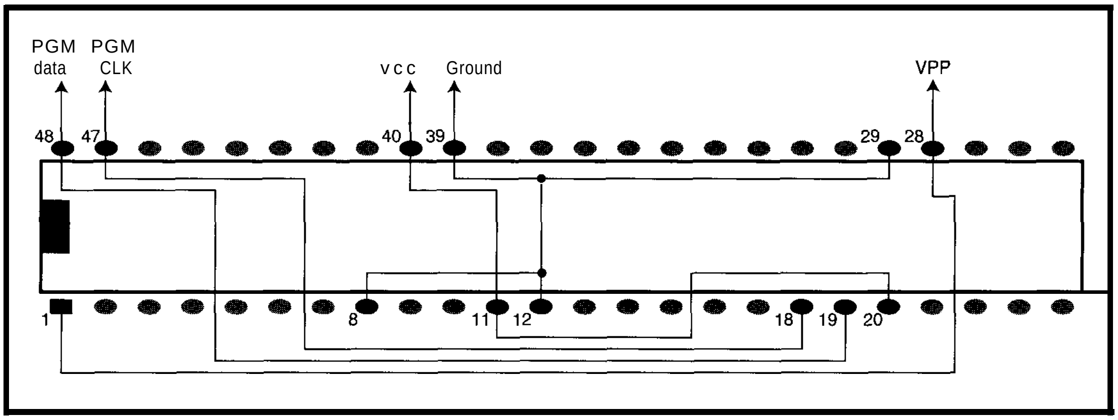


Figure 3—*Fortunately, wiring up the 48-pin ZIF socket for fhe Micro-bRISC involves only* 13 *pins. Target* V_{cc} *and* V_{pp} *are switchable voltages and connect to the collector of transistors Q3 and Q2, respectively.*

ibility with the existing base of standard parallel port peripherals.

One caveat concerns the quality of Centronics printer cables. You might think that all Centronics signals are always wired through, but nothing could be further from the truth since the Centronics specification is a de facto standard. In my printer cable quest, I sampled a few cables from vendors and was unpleasantly surprised to find that not all cables implement all signals in the Centronics specification.

Perform a cable continuity test to ensure that all signals are wired through. If they are not (barring the INIT signal), Micro-bRISC fails to communicate with the host computer. My suggestion: avoid "economy" printer cables since these types are more likely to skimp on shielding and eliminate certain signal lines.

Since the original parallel port specification, unlike the IEEE 1284 standard, calls for no set cable impedance and termination techniques, I recommend keeping the cable length to 6-10'. I have experimented with cable lengths as long as 18' and found no errors.

THE SOFTWARE DANCE

The GUI software that drives the Micro-bRISC device programmer can be evaluated without the Micro-bRISC board because it incorporates a demo feature.

The software provides typical program, blank check, read, verify, edit, file read, and save functions. Of course, programming, editing, and erasing the PIC16C84's 64-byte data EEPROM area is standard along with programming and editing the customer ID, device configuration bits, and checksum.

Version 1 .O was completed with QuickBASIC, but the user-interface and file I/O section left a lot to be desired. To create a more polished, professional, and functional user interface, I migrated to Visual Basic for DOS. Although this was my first Visual Basic application, I found it a tidy, easy-to-use environment. Its powerful event-driven programming and tools are a stark contrast to QuickBASIC's programming environment.

Since Visual Basic is a compiled rather than interpreted BASIC, software executes much faster. I also analyzed critical I/O sections and their assembly language output to optimize them for speed. While this is a cumbersome method, compiler optimization options are nonexistent in Visual Basic.

Photo 2 shows the main screen of the host PC software (MICBRISC. EXE) that drives and communicates with the target programmer board. Various programming options can be selected by using the keyboard or with a mouse.

The user doesn't need to be concerned with LPT numbers or whether the parallel port is a standard or bidirectional type. The software interrogates the system BIOS area to find all parallel ports installed. Then, all installed parallel ports are polled, so the software can lock onto the port Micro-bRISC is connected to. If a bidirectional port is detected, special software and firmware routines are summoned to take advantage of the port's higher throughput capability.

AVOIDING BABBLE

For proper data interchange between computers and peripherals, hardware and software protocols have to be defined and strictly adhered to—Micro-bRISC is no different.

Data transfers between the host computer and Micro-bRISC occur with the help of tightly interlocked handshaking signals that provide flow control. With this hardware handshaking method, throughput is highly dependent on the speed of both devices. In other words, maximum data transfer is obtained when both devices operate at their maximum potential. But, one device always limits maximum throughput.

Just as hardware plays a large role in the communication link, so does software. Since poorly written software brings the hardware to a grinding halt, I honed the I/O software routines for speed as well.

To combat spurious signals, the target firmware double polls the flow-control handshake signals from the parallel port. Although target and host communications work fine with single polling, I wanted a safety net. The extra overhead proved negligible.

SPLIT PERSONALITY

Since implementing the parallel port and high throughput were key design issues, the choice of target processor was very important.

To properly implement Microchip's serial programming algorithm, I needed host independence. If the PC provided all intelligence, obtaining precise programming delays would unnecessarily complicate the hardware design. Critical timing resolution in software would also be more difficult, especially if the user ran DOS from Windows. Interrupts, TSRs, and multitasking all lead to inaccurate PC software timing delays and could stress EPROM cells, leading to early device failure.

For design integrity, I refused to veer from Microchip's programming spec. I had to have distributed intelligence. I opted for PC independence and delegated work by using a microcontroller as the target intelligence performing the device programming algorithm.

In this way, the target processor takes care of all critical timing in the programming algorithm, leaving the host PC to perform the more mundane duties. Micro-bRISC is therefore impervious to a preemptive multitasking environment such as Windows 95. And, the PC only affects the speed of data transfers, not the highly time-critical protocols of the programming specification.

I chose the PIC16C57 as the intelligent liaison between Micro-bRISC and the PC. The PIC16C57, operating at its full speed of 20 MHz, is a true work horse (approaching 5 MIPS). More importantly, a processor with high I/O drive capability for the printer cable was crucial, and the PIC16C57 fit the bill rather nicely.

Although this high-speed distributed intelligence plays a vital role in the celerity of data throughput and programming cycle times, Micro-bRISC is aided by the inherent nature of EPROM cells and how they relate to Microchip's intelligent programming algorithm. EPROM cells most often program during the first attempt, thus drastically lowering the programming time of the selected device.

A programming attempt consists of a single 100-µs programming delay. An overprogram cycle is then applied to the same EPROM location to ensure proper programming margin. The overprogram cycle is defined as 3n program cycles, where *n* equals the number of programming attempts necessary before a successful verification. The overprogram cycle, rather than a single delay, is actually a series of 3*n* separate, 100-µs programming cycles.

The Microchip programming algorithm is similar to the Intel Quick-Pulse Programming (QPP) algorithm, except that Microchip uses overprogramming techniques. The QPP algorithm takes advantage of the fact that most EPROM cells program during the first 100-µs pulse [1].

For Micro-bRISC, the status variable *n* tracks the number of programming attempts that occur when programming the device. After programming is complete, the target transmits the value of *n* to the PC, which displays it on screen. As dictated by the Microchip algorithm, the maximum value that *n* can reach is 26, which means one or more individual cells within an EPROM location has failed to program correctly, thus rendering a device-program failure.

I/O CAVEATS

With PIC microcontrollers, a write immediately followed by a read on the same port can yield unpredictable

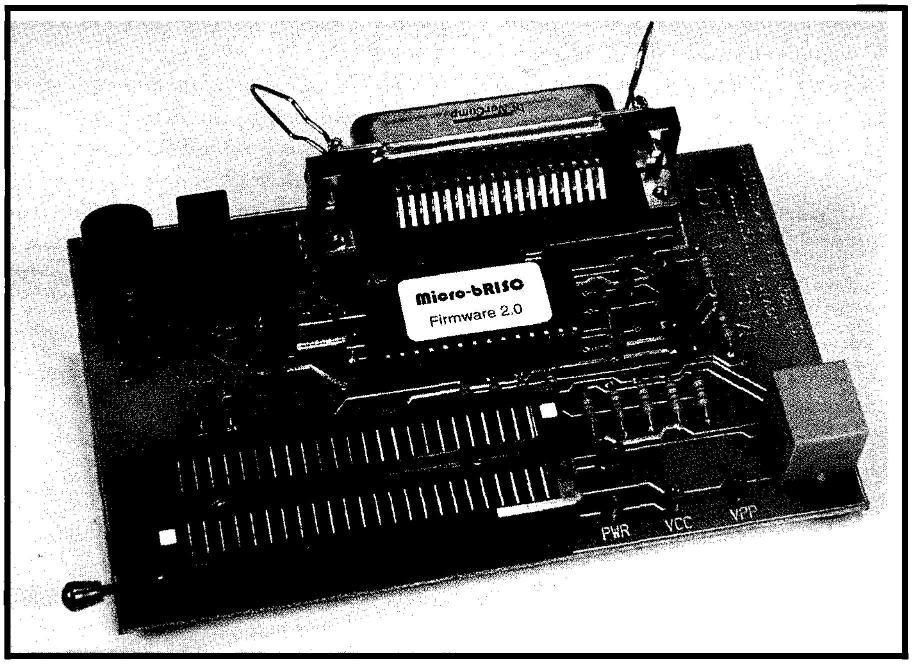


Photo 1 —*The single-chip* ***Micro-bRISC*** *device programmer is an elegant* ***solution*** *for programming Microchip's* midrange family of *PICs.* However, don't forget to fasten your *seatbelt—Micro-bRISC* is fast.

results. Carefully observe how your read, write, and read-modify-write instructions are placed in the program flow when they operate on the same I/O port.

An I/O port write occurs at the end of an instruction cycle and an I/O port read at the beginning. If the port pin is then presented with a load that has a significant RC time constant, the old pin voltage could be read instead of the new one before the capacitive load charges or discharges. This problem is exacerbated at high clock frequencies and is load dependent.

If necessary, a time delay can be sandwiched between successive, same-port write/read operations so the port pin can stabilize. This time delay could simply be a N 0 **P** or any instruction that doesn't access the same port. Unfortunately, this band-aid approach chokes the PIC RISC engine, severely limiting the maximum toggle frequency of an I/O port pin.

Because this PIC idiosyncrasy is clock speed and I/O pin load dependent, you need to decide on a case-by-case basis in all PIC I/O interfacing designs. Many designs don't *have* to address this issue, but can you afford not to? After all, a safety net costs nothing and, if problems arise, troubleshooting is much easier.

Since the target PIC16C57 is running at full speed and there is plenty of capacitance in printer cables, I liberally sprinkled N 0 **P** instructions between successive operations on the same port.

THE HARDWARE

Figures 2 and 3 show the Micro-bRISC circuit design.

Micro-bRISC accommodates either 12-18 VAC or 18-24 VDC. Current consumption is less than 100 mA worst case, so a wall transformer with a rating of at least 250 mA fairs well.

When AC outlets are not available, Micro-bRISC can be powered by batteries and used with a laptop computer. A rechargeable NiCd power pack is ideal as long as the DC input voltage requirement is met.

Don't worry about reverse polarity damage to the Micro-bRISC when using DC power supplies. A full-wave bridge rectifier produces the (peak) absolute value of the RMS input voltage less the diode drops. I chose to use an integrated bridge rectifier in a 4-pin DIP package for convenience, but discrete diodes work just as well. Just make sure the surge current rating of the diodes is acceptable. The 1000-µF capacitor is a fairly small filter capacitor and charges rather quickly. Diode damage due to excessive surge current is a remote possibility.

The LM3 17 voltage regulator produces a programming voltage (V_{pp}) of 13 V for target devices and feeds the 7805 voltage regulator. The output of the LM317 serves as the 7805's input to reduce the 7805's power dissipation. If the 7805's input was fed from the output of the filter capacitor (C3), it would dissipate a lot more power due to the higher voltages which can occur at that point.

Because of this and my aversion to heat sinks in small-scale projects, both voltage regulators use TO-220 packages. Although the smaller TO-92 regulator package saves space, it offers lower power dissipation and thus cannot be used. Also, since future PIC

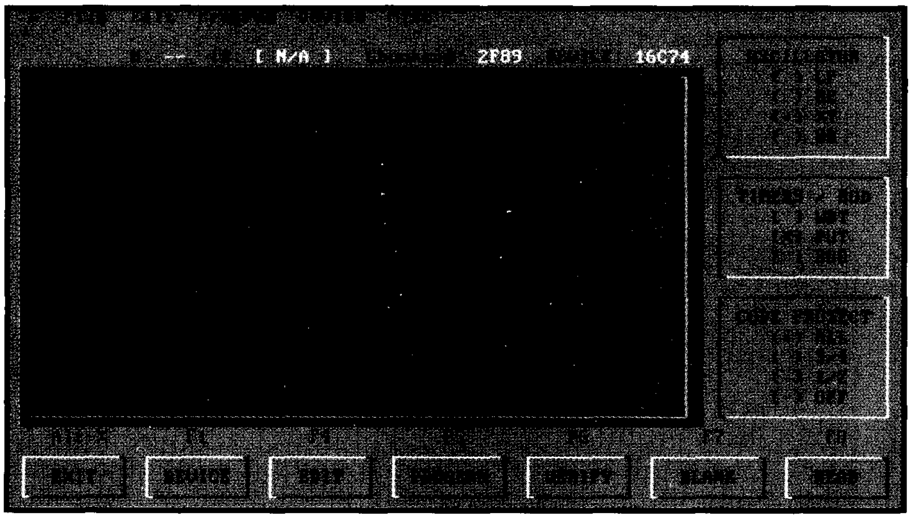


Photo **2—*The Micro-bRISC*** *Visual Basic for DOS* ***software*** *offers an easy-to-use and* ***powerful*** *user interface for programming* ***PICs.***

chips might require more power to program, using the TO-220 package size seems most appropriate.

An added benefit of having the 7805's input fed by the LM317's output is reduced output voltage ripple. By the time the 7805 receives its input voltage, the voltage ripple has already been reduced by the LM3 17's inherent ripple-rejection ratio characteristics, which in the case of the JRC LM3 17 is 65 dB.

On LM3 17 parts, the ripple rejection ratio significantly increases if the ADJ pin is adequately bypassed. For example, if the ADJ pin on the JRC LM3 17 is bypassed with a 10-µF capacitor, the ratio jumps to a whopping 85 dB (keep this in mind with linear regulator designs using this device).

Given that the filter capacitor value and the rectified AC frequency (60-Hz half wave or 120-Hz full wave) are held constant, a power supply's voltage ripple is proportional to load current. Since Micro-bRISC uses fairly low current, further voltage ripple reduction of the LM3 17 by bypassing the ADJ pin was not warranted.

The 7805 voltage regulator produces the system V_{cc} of 5 V. Even though the output of the 7805 is the result of two stages of ripple massaging, I decided to filter the output of the 7805 with a 10-µF tantalum capacitor (C6) because pristine voltage is paramount in any device programmer.

These voltage regulators manufactured by New Japan Radio are pretty hip! The TO-220 package size is totally encased in high-thermal-conductivity epoxy resin (no more messy silicone grease, mica shields, or plastic washers). Since its metal tabs are coated with heat-dissipative resin, they are electrically isolated from each other even when both must be attached to ground and are in close proximity.

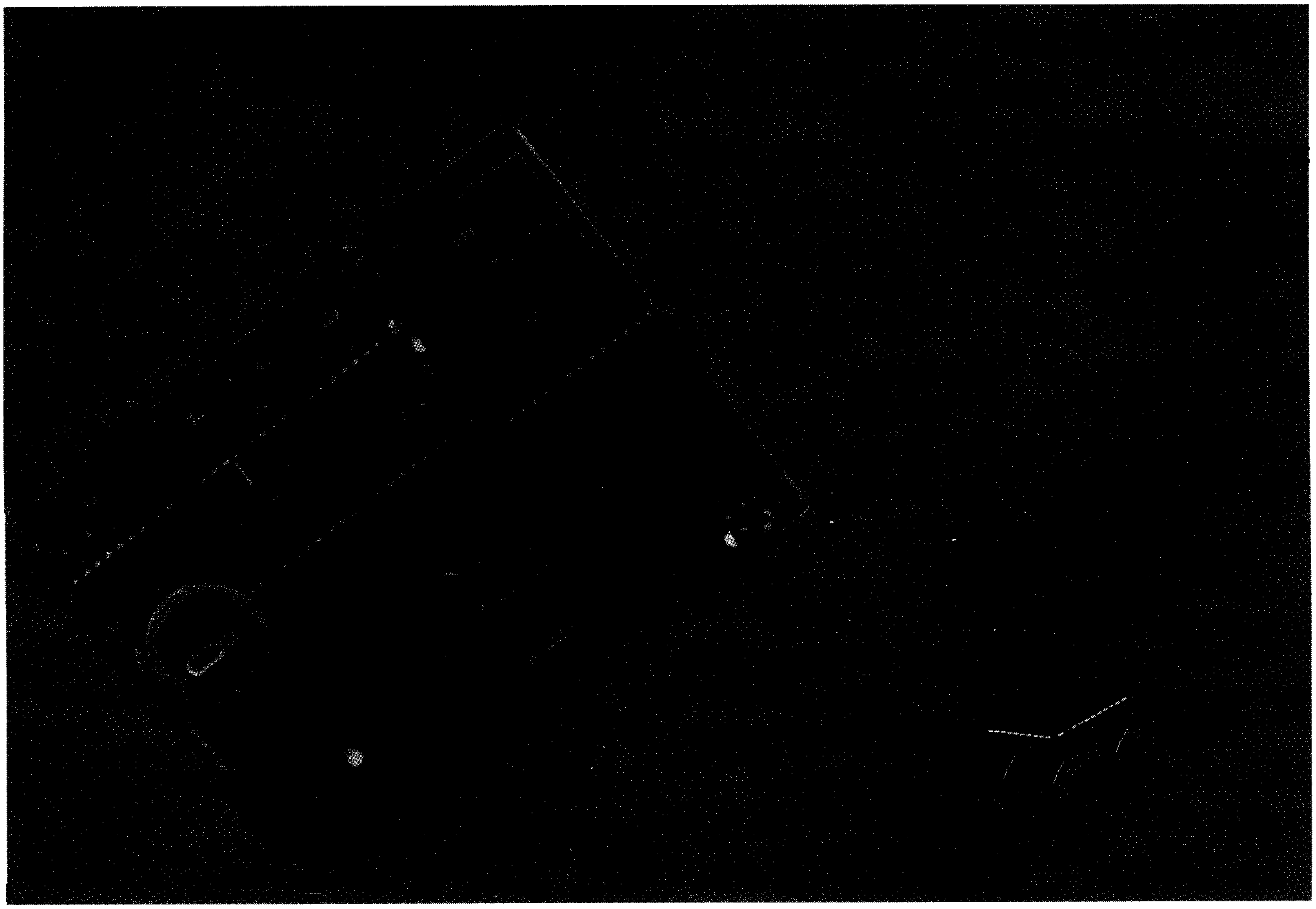
Remember to adequately bypass the input and output of the voltage regulators to minimize the chance of oscillation and to improve transient response and stability. Capacitors C4, C5, and Cl 1 serve this purpose.

The oscillator circuit consists of a 20-MHz crystal and two 18-pF load capacitors. Although a ceramic resonator provides an accurate clock source for the PIC16C57, a crystal clock source is more readily available (in high-frequency ranges) and accurate.

Here, I've used a fairly unconventional part as well. The Epson CA-301 is a cylinder-type crystal, which is inexpensive and compact. Besides having a frequency tolerance of ±30 ppm, the case is extremely small, like the tuning fork crystals found in wristwatches.

In space-critical applications, these crystals are a godsend, easily fitting in open-frame DIP sockets. For this particular 20-MHz crystal, Epson recommends a maximum series resistance value of 40 Ω to prevent overdriving the crystal. I did not use a series resistor and have not experienced any problems, but since they require a low drive level, the addition of a series resistor is not a bad idea.

The use of a 20-MHz crystal is mandatory since the precision microsecond



delays for programming the PIC's EPROM locations are based on a 20-MHz crystal. Using a lower frequency crystal would significantly lengthen them, which in turn could stress the EPROM cells of the device to be programmed.

LED indicators show the user system status. The green LED indicates system power is on. The V_{cc} and V_{pp} LEDs indicate the presence of the V_{cc} and V_{pp} voltages at the target ZIF socket, showing various functions such as programming, reading, or verifying.

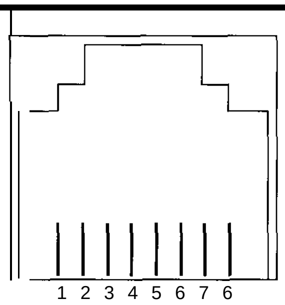
The V_{cc} and V_{pp} LEDs also serve another function. If a bidirectional port is detected, the V_{cc} and V_{pp} LEDs quickly flash three times after Micro-bRISC's software is invoked. If a standard parallel port is detected, no flashing occurs. When the Micro-bRISC program is exited, the V_{cc} LED flashes to indicate that all the I/O lines connected to the parallel port are forced into input mode [high impedance].

Putting Micro-bRISC in this known state, after exiting the software, prevents contention problems if another program tries to access the parallel port connected to Micro-bRISC. These auxiliary status functions can be turned off via command-line options by typing STATUS:OFF at the command line before invoking MICBRISC.EXE.

Transistor Q3 serves as a voltage switcher routing V_{cc} to the target ZIF socket when it's needed under program control. Similarly, Q1 and Q2 form a voltage-switching network that routes V_{pp} to the ZIF socket which is also under program control.

As suggested by Microchip, the PIC16C57's ● MCLR pin has a 100-Ω resistor serially connected to prevent the possibility of device latch-up. The *MCLR pin is asserted by the Select In signal from the parallel port control register. This assertion enables the software to reset the PIC16C57 via Micro-bRISC's programming software. Capacitor C7 filters the *MCLR line to prevent a system reset due to superfluous noise.

The two resistor networks provide pull-up resistors for all implemented I/O lines of the parallel port and certain PIC16C57 I/O lines. C8 and C9 help filter any noise on the Auto Feed XT and Strobe lines from the parallel port control register. These lines need extra care since they are critical handshaking lines.



RJ-45 jack connector
(front view)

Target signal pinout:

1 PGM CLK	(TTL level)	Connects to PIC16C57, pin 13
2 VCC: 5 V	(0 V, 5 V switchable)	Connects to collector of transistor Q3
3 PGM data	(TTL level)	Connects to PIC16C57, pın 12
4 System ground		Connects to system ground
5 RA3	(TTL level)	Connects to PIC16C57, pin 9
6 VPP on/off	(TTL level)	Connects to PIC16C57, pin 10
7 VCC on/off	(TTL level)	Connects to PIC16C57, pin 11
8 VPP: 13 V	(0 V, 13 V switchable)	Connects to collector of transistor Q2

Figure 4-h anticipation of future *developments, Micro-bRISC uses a RJ-45 jack* to *bring programming signals to the outside world.*

Signal termination can never be fully or properly addressed because cable standards and electrical standards were never formalized for the original parallel port. There can be a lot of variations between I/O cards or cables. Fortunately, the IEEE 1284 specification changes all of this-even a backward-compatibility mode is compulsory.

When working with the Centronics connector, care must be taken to ground all pins indicated in the schematic to system ground. The metal chassis of the connector may need to be grounded as well. with proper grounding and a high-quality printer cable, you should have good luck with the Micro-bRISC. (The INIT pin from the parallel port control register is not implemented in my design.)

The power supply pins should be adequately bypassed with decoupling capacitors. The capacitor (with leads cut short to minimize any ringing caused by lead inductance) should be placed as close as physically possible to the PIC. The PIC16C57 is running at top speed so it needs all the help it can get. Similarly, the V_{cc} and V_{pp} voltages entering the target ZIF socket should be bypassed to ground with 0.1 -µF capacitors, which are placed as close to the ZIF socket as possible.

CONSTRUCTION TIME

A Micro-bRISC printed circuit board can be assembled in under 2 hours. Or, you can build it using wire-wrapping or point-to-point techniques. Be sure to use a 28-pin machine socket for the PIC16C57 and two 24-pin machine sockets end-to-end to hold the ZIF socket.

Visual Basic for DOS software and PIC16C57 firmware are available on the Circuit Cellar BBS. When programming the PIC, make sure you set the oscillator fuse to HS and the watchdog timer off.

Use either a UV-erasable PIC16C57-JW or a PIC16C57-HS OTP [one-time programmable) part only. Other devices are not guaranteed by Microchip to operate properly at 20 MHz.

Directly across from the 48-pin socket is an RJ-45 socket that carries the programming voltages and signals to the outside world. This socket provides a starting point to implement in-circuit programming of microcontrollers. In the future, the RJ-45 socket programs package sizes with socket adapters for system flexibility. For the RJ-45 signal pinout, refer to Figure 4.

Although programming PICs is Micro-bRISC's main purpose, I specifically call it a device-not PIC—programmer. In the future, I plan to support nonPIC devices, such as 8-pin serial EEPROMs (2, 3, and 4 wire] and the Dallas Semiconductor DS 1620 digital thermometer.

THE NEXT GENERATION

Although the PIC 16C5*x* microcontrollers have been the most popular so far, Microchip's next generation has a lot to offer.

The Micro-bRISC device programmer and Microchip's free MPASM universal assembler enable you to burn PIC microcontrollers without burning a hole in your wallet.

So, set an evening aside, download the files, get out your soldering iron, and have some fun programming PICs with one of the quickest PIC programmers around! ❑

Special thanks to Marc DiCicco for making a PCB.

Ken Pergola holds a B.S. in Electrical Engineering Technology from SUNY Institute of Technology. He is particularly interested in digital design and microcontroller and microprocessor programming and interfacing. He may be reached at kenneth.pergola@circellar.com.

SOURCES

MPASM assembler
Microchip Technology, Inc.
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
(602) 786-7200

Bidirection parallel port boards
Essential Data
P.O. Box 640963
San Jose, CA 95164-0963
(800) 7954756

Dalco Electronics
275 Pioneer Blvd.
Springboro, OH 45066
(800) 445-5342

FarPoint Communications
104 East Ave. K-4, Ste. F
Lancaster, CA 93535
(805) 726-4420

Micro-bRISC components

Preprogrammed OTP PIC16C57-HS firmware	$20
Printed circuit board	$25
Kit (PCB, firmware, parts)	$99'
Fully assembled/tested Micro-bRISC	$125*
Registered upgrade software	$15
Shipping and handling (U.S.)	$5

* Includes full upgrade registered software
MA residents add 5% sales tax

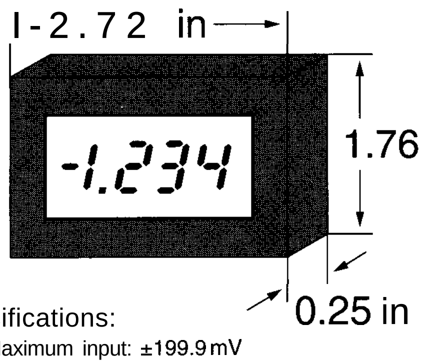
Ken Pergola
2088 Swamp Rd.
Richmond, MA 01254-9338
(413) 698-3167
kenneth.pergola@circellar.com

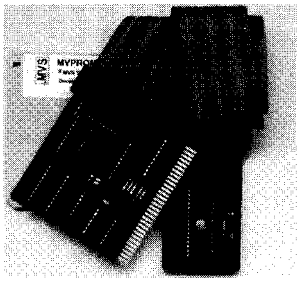
REFERENCES

[1] Intel, "The Quick-Pulse Programming Algorithm," *Intel Memory Components Handbook, AP-277,* 1988.

I R S

410 Very Useful
411 Moderately Useful
412 Not Useful





Alpha's PALcode

FEATURE ARTICLE

Eric Rasmussen

Building on the foundation of the Alpha article in *INK* 61, Eric discusses the role of PALcode in Alpha architecture. After defining what PALcode is, Eric suggests how to use it in the PALmode environment.

One of the Alpha architecture's most important features is the independence of its operating system from its hardware platform. The Privileged Architecture Library code (PALcode) consists of software routines that provide a common programming interface for the operating system across many different physical implementations of the Alpha architecture.

PALcode supports the OpenVMS, Digital UNIX, and Microsoft Windows NT operating systems. Digital offers a sample PALcode product and a set of development tools that designers use to customize PALcode for real-time, embedded, and teaching applications.

After discussing the role of PALcode in the Alpha architecture, I'll look at the characteristics and functions of PALcode that appear as a combination of microcode, ROM BIOS, and system service routines. I'll also describe the special execution environment known as PALmode and detail how and when PALcode is invoked.

WHAT IS PALCODE?

PALcode implements various low-level hardware support functions that are too complex, too costly, or otherwise impractical to implement directly in the microprocessor's hardware.

These low-level functions, which may be used to handle interrupts, dispatch exceptions, and perform memory management and other tasks, cannot be handled by normal operating-system software. In some architectures, microcode traditionally handles these hardware functions, but the Alpha architecture is careful not to add complexity to the chip implementation through the use of microcode.

Other functions implemented in PALcode must run atomically, even though they involve long sequences of instructions that may need complete access to the underlying computer hardware. Examples include a sequence that returns from handling an interrupt or exception, or a power-up sequence that initializes the hardware to a known state.

Instructions needed to maintain backward compatibility or ease of programming are also implemented in PALcode. Such instructions are not used often enough to dedicate them to hardware, nor so complex that they compromise overall performance of the computer.

For instance, PALcode could emulate an instruction that is lacking direct hardware support in a particular chip implementation, or an instruction that performs CISC-style interlocked memory access. In each case, PALcode offers a flexible way to handle hardware functions in a special environment situated between the chip and operating system.

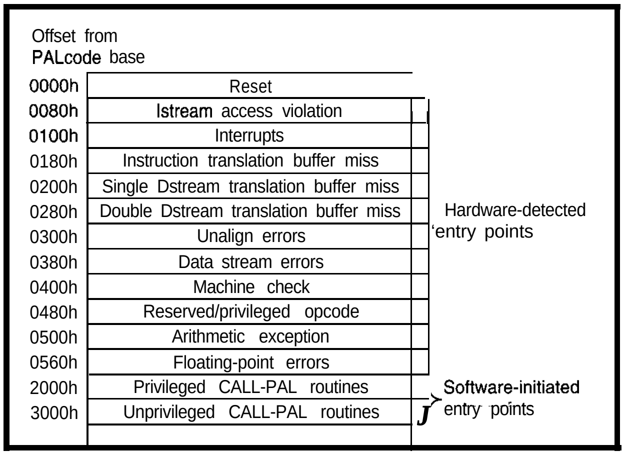
THE PALmode ENVIRONMENT

For most operations, PALcode is implemented with the standard Alpha instruction set. Code is read in as instruction-stream code in the same manner as any other Alpha machine code. Once invoked, however, PALcode executes in a special, privileged environment called *PALmode,* which differs from the normal operating environment.

Since PALcode implements memory-management tasks, instruction-stream memory mapping is disabled while executing in PALmode. Interrupts are also disabled in PALmode to provide long instruction sequences in the form of atomic operations. Most functions handled by PALcode are privileged and need control of the lowest levels of the machine, so programs in this environment have complete access to all underlying computer hardware.

The Alpha architecture sets aside five reserved opcodes for exclusive use in PALmode. These opcodes have implementation-specific extensions that facilitate access to low-level system hardware. For example, on the Alpha 21164 microprocessor, PALmode-only instructions may

- perform load or store operations on physical memory without invoking the memory management routines (HW_LD,HW_ST)
- move data to and from internal processor registers (HW_MFPR,HW_MTPR)
- return from an exception or interrupt (HW_REI)



Offset from PALcode base	
0000h	Reset
0080h	Istream access violation
0100h	Interrupts
0180h	Instruction translation buffer miss
0200h	Single Dstream translation buffer miss
0280h	Double Dstream translation buffer miss
0300h	Unalign errors
0380h	Data stream errors
0400h	Machine check
0480h	Reserved/privileged opcode
0500h	Arithmetic exception
0560h	Floating-point errors
2000h	Privileged CALL-PAL routines
3000h	Unprivileged CALL-PAL routines

Hardware-detected entry points
Software-initiated entry points

Figure 1—*PALcode* entry *points for the Alpha* 21164 microprocessor are listed in descending priority order along *with* their *relative offsets to the PALcode* base address.

These instructions generate an opcode decode (OPCDEC) exception if executed outside of the PALmode environment.

PALcode is processed in the same manner as any other Alpha machine code, so it is subject to the same scheduling and issuing rules as native-mode code. Schedule PALcode according to the rules governing instruction latency and function-unit availability for your specific microprocessor implementation. This technique is good practice in most RISC architectures.

When executing in PALmode, you encounter additional restrictions that involve internal-processor-register access and use of the privileged PALmode-only instructions. Since PALmode gives the programmer complete control over many of the internal workings of the microprocessor, violating PALcode restrictions may cause unintended side effects in what appears to be perfectly acceptable code.

INVOKING **PALcode**

PALcode is invoked at specific entry points under certain well-defined conditions. Think of PALcode as a series of callable routines, each indexed by an offset from a base address. The base address of PALcode is programmable, stored in an internal processor register, and normally set by the system reset code. Figure 1 shows PALcode entry points for the Alpha 21164 microprocessor and their relative offsets from the PALcode base address. Entry points are listed from highest to lowest priority.

When an event occurs that invokes PALcode, the Alpha microprocessor drains the pipeline, loads the current PC into the exception-address (EXC_ADDR) internal-processor register, and dispatches the appropriate PALcode routine. These operations occur under direct control of the microprocessor's chip hardware, then the machine goes into PALmode. On completion of the PALcode routine, the hardware loads the new PC, enables interrupts and memory mapping, disables all PALcode restrictions. and returns to native mode.

The Alpha 2 1164 microprocessor uses PC<0> as the PALmode flag both to the hardware and to PALcode itself.

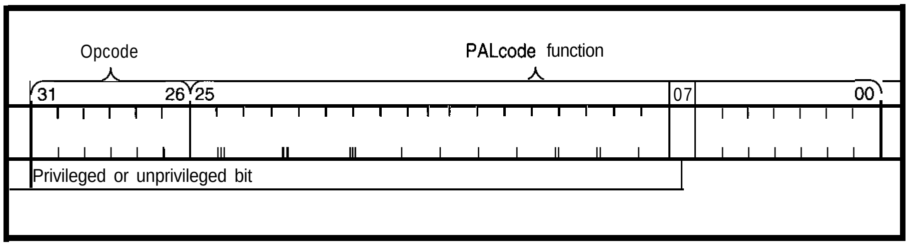


Figure **2—***The* CALL_PAL instruction *format* **specifies** *extended processor functions. The* CALL_PAL format contains a d-bit *opcode field* and a *26-bit* function field. *The* Alpha 21164 microprocessor uses bit 7 of *the* function field *to* distinguish *between priviledged and nonprivileged* CALL_PAL *functions.*

When the CPU enters PALcode, PC<0> is set to one. It remains set as instructions are executed in the PALcode instruction stream. The hardware ignores this bit and behaves as if the PC were longword aligned. On transition back to native mode, the new state of PALmode is copied from EXC_ADDR<0>.

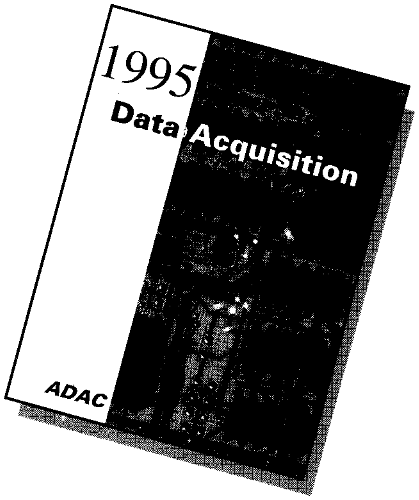
The most basic way to invoke PALcode is in response to a hardware-detected event. PALcode is invoked automatically when the particular hardware event is triggered. This method is analogous to other architectures' use of microcode. For example, when a translation buffer (TB) miss occurs, one of several PALcode routines performs the TB fill. Under control of an operating system, these routines consult the system's page tables and perform the fill based upon a page table entry.

The hardware-detected PALcode entry points can be grouped into four major categories:

- reset-low-level system initialization
- hardware errors-uncorrectable errors, arithmetic exceptions, illegal-opcode fault, data-fetch errors
- interrupts-hardware, software, and AST
- memory-management exceptions-TB miss, accessviolation, translation-not-valid fault

Another method used to invoke PALcode is the CALL_PAL instruction. This instruction is dispatched to PALcode at a specific entry point in the same manner as the hardware-detected method. However, the dispatch occurs whenever the CPU encounters CALL_PAL in the instruction stream, instead of through a hardware-detected event or error.

The CALL_PAL instruction accepts a function parameter that specifies which of 256 possible CALL_PAL routines to invoke. (The Alpha 21164 microprocessor supports only a subset of possible CALL_PAL offsets.) CALL_PAL routines may even perform different functions for different operating



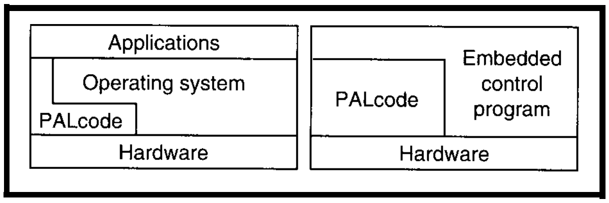


Figure 3—*PALcode adapts the hardware to the requirements of the operating system or embedded control program.*

system environments. CALL_PAL functions are largely optional and are based on system implementation needs. Figure 2 shows the format for the CALL_ PAL instruction.

CALL_PAL functions can be grouped into two categories: privileged and nonprivileged. The designation refers to whether the-caller has sufficient privilege to call that particular routine, not the execution mode of the CALL_ PAL routine. Without exception, every CALL_PAL instruction is dispatched to PALcode and runs in PALmode.

Both categories of CALL_PAL instructions are dispatched in exactly the same manner:

- enter PALmode when executed
- perform your function
- return to the caller in native mode

The only difference is that a check is made prior to execution to determine if the caller is in the correct mode. On an Alpha 2 **1164** microprocessor, the current-mode bit of the processor-status register (PS<CUR_ MODE>) is checked. If a nonkernel-mode user attempts to execute a privileged CALL-PAL routine,an OPCDEC exception occurs.

This mechanism gives both layered applications and operating system software access to low-level hardware functions. A layered application can choose to call nonprivileged CALL_PAL routines directly or allow the operating system kernel to intervene on its behalf for privileged operations. Figure 3 shows the relationship between hardware, operating system, layered application programs, and PALcode.

CONCLUSION

PALcode enables the Alpha architecture to accommodate a variety of operating systems. The architecture uses it to provide a consistent programming interface to whatever type of hardware without resorting to microcode. PALcode is supplied with Digital's operating system software and firmware and is designed to be sufficiently complete that users shouldn't need to change it.

PALcode has a high degree of specificity. It is invoked at specific entry points under certain well-defined conditions, and its functions vary under different implementations. System designers who need to modify PALcode for special requirements should exercise extreme caution. Failure to observe PALcode's special conditions and caveats can result in unintended side effects that could potentially render the system inoperable or jeopardize system performance.

If you want to customize PALcode for real-time, embedded, or other applications, contact Digital for information on how to obtain the sample PALcode product and development tools. ◼

Eric Rasmussen is a principal engineer in the Semiconductor Engineering Group at Digital Semiconductor, a Digital Equipment Corporation business. Eric has worked on PALcode development projects for the Alpha 21064 and 21164 microprocessors and other VAX and Alpha CPU development projects at Digital over the past 11 years. He may be reached at rasmussen@ricks.enet.dec.com.

CONTACT

Digital Semiconductor
75 Reed Rd.
Hudson, MA 01749
(508) 568-6868
Fax: (508) 568-6447

IRS

413 Very Useful
414 Moderately Useful
415 Not Useful

Nonlinear Graphics Transforms

Shortcuts to Stunning Graphics

In graphics, you need to bend, twist, squash, or stretch an image. Although linear graphics transforms are standard, Don is more interested in the exotic stuff of nonlinear transforms.

FEATURE ARTICLE

Don Lancaster

One of the key tools in graphics is the ability to select an image and suitably bend, twist, squash, or stretch it over other visual surfaces.

The linear graphics transform is the industry-standard tool used for simple graphics mappings. But for really exotic stuff, you may need more elegant tools using higher-level nonlinear techniques.

LINEAR GRAPHICS TRANSFORMS

A digital transform is simply any method of taking an existing pile of numbers and then following specific math rules to create another pile of numbers. This new set of numbers hopefully turns out to be better in some specified way.

The linear graphical transform is the stock method for changing the size, direction, or final position of a visual image. Matrix techniques are generally used.

Since my eyes gloss over when I see matrix concatenation, I'll substitute ordinary algebra here. I'll also limit this discussion to flat or two-dimensional images.

A linear transform accepts some pair of data values *x,y* and changes them into a new and different pair of values *x',y'*. The linear graphical transform is often shown in this form:

$$x' = Ax + By + C$$
$$y' = Dy + Ex + F$$

Constant ***A*** sets horizontal size, ***B*** the amount of lean, C the x offset, ***D*** vertical size, *E* the climb, and *F* the y offset. Three popular transforms include translation, scaling, and rotation. To reposition, pick a nonzero value for *C* (to shift to the left or right) or a nonzero value for *F* (to move it up or down).

To scale an image, change ***A*** and ***D*** to nonunity values. Parameter ***A*** sets the horizontal scale factor and ***D*** sets the vertical. Often, ***A*** and ***D*** are set to identical values. If not, you get anamorphic scaling. Changing the sign on ***A*** creates a mirror image. Changing the sign on ***D*** creates an upside-down image or redefines direction.

Rotation is a tad obscure. To rotate something, use these values:

$$A = \cos\theta$$
$$B = \sin\theta$$
$$c = o$$
$$D = \cos\theta$$
$$E = -\sin\theta$$
$$F = O$$

where θ is the angle of rotation.

Translation, rotation, scaling, and other alterations of ***A-F*** create different special effects. Beware, however. Changing the sequence of operations alters the final results! Rotating and then translating is vastly different from translating and then rotating, just as first multiplying and then adding differs from adding and then multiplying.

One subtle but important use of the linear graphics transform is to take you from math space to device space. It's a good idea to keep your set of plans in a device-independent form, having an arbitrary accuracy that's subject only to word-size limits. When it comes time to put the image on a screen, a piece of film, or a sheet of paper, the linear graphics transform converts your device-independent math-space data into numeric values matching pixel size, resolution, and media limits.

Another use for the linear graphics transform is for microsizing. Paper swells and shrinks. Print engines drift. Flexographic printing plates distort when wrapped around a press drum. Microsizing simply provides very small changes (such as ***A*** = 1.005 or ***D*** = 0.9961 in a scale factor.

ISOMETRIC

One useful linear transform is the isometric transform shown in Figure 1. Isometric drawings are often used for assembly diagrams. The original vertical or z-axis remains vertical on the page in the y' direction. The original x-axis slants up the page at an angle of +30° and the original y-axis slants backwards up the page at an angle of 150". Typical circles become 35.27" ellipses.

Isometric drawing offers the advantages that the original drawings are easy to do using pen and ink, and you can measure any value along any axis. One big negative is that the rear corners of boxy objects seem too big because the eye tries to see them in perspective.

The isometric linear transform looks like this:

$$x' = x\cos(30) - y\cos(30)$$
$$y' = x\sin(30) + y\sin(30) + z$$

which simplifies to:

$$x' = 0.866x - 0.866y$$
$$y' = 0.500x + 0.500y + z$$

These days, genuine perspective is nearly as simple and looks far better. But, isometric drawing is still useful when you seek a Drafting 101 effect or need to scale dimensions.

NONLINEAR TRANSFORMATIONS

Linear graphical transformations can be powerful, flexible, and computationally cheap. But, there are many things they cannot do. For instance, you can change a square into another square of any size at any angle. You can transform it into a rectangle, parallelogram, line, or even a single point. You can repeat images, flip them, or reverse them.

But a linear transformation can't convert a square into the odd trapezoid you may need for a 2D architectural perspective or the quadrilateral required for full 3D. A nonlinear graphics transform (NLT) takes a group of numbers and applies one or more rules to it. Some new numbers are created that look graphically different.

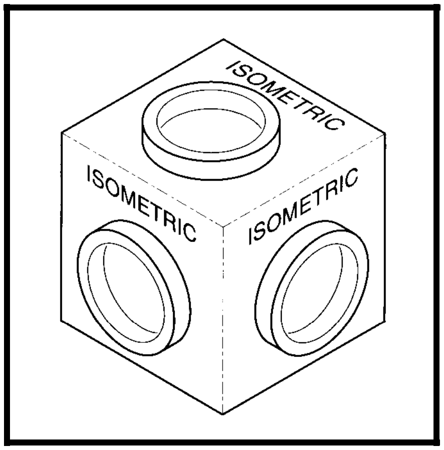


Figure 1—*The isometric linear graphics transform is useful for assembly diagrams or wherever you need to draw dimensions to scale.*

The key difference is that in a linear transformation, values ***A-F*** are constants over the entire current working area. In a nonlinear transformation, *A–F* are calculated values which may need to be recomputed every time the transform is used.

For instance, the constant value for ***A*** in a linear transform becomes a calculated value in a NLT. This value may depend on the x or y location on the page, involve trig, or invoke a random number or two. To do a nonlinear transform, calculate the immediately required values for ***A-F*** and then do a linear transform for these local values.

GRAPHICAL PRIMITIVES

In theory, you can take each pel (minimum resolvable data value) in the original and carry out a nonlinear transform on it. This computation generates a new image with the change or distortion you're after. Working pel by pel may be the only solution when you're rectifying aerial photographs or are stuck with bitmap data.

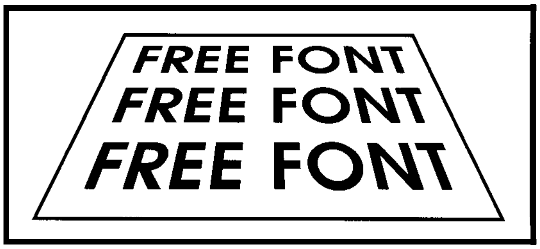


Figure 2—*The **Star-wars** nonlinear graphics transform gives the **effect** shown here. **Jo** create if, select a tilt angle θ where 0° is flat and 90° is vertical. Predefine a tilt factor with geometric constant k, k = fullheight fan θ. The nonlinear transform is then x' = xk/(k + y), y' = yk/(k t y).*

Obviously, taking each point in a high-resolution image and doing calculations on all the points is computationally expensive. Instead, work with a sparse data set which needs far fewer nonlinear transforms.

Graphical primitives offer one route toward sparse data sets. They are operators that cause an image path to get built up. Ideally, these operators demand little input data. They apply algorithms to generate far more detailed results. You need only four graphical primitives for image buildups.

The first primitive is a positioner. Given a pair of x and y values, it moves you to that new location. In deference to PostScript, I call this positioner a *moveto.*

The second primitive appends a line to the existing path. It assumes a previous pairing of initial location values and accepts a newer pair of x and y endpoints. Note the efficiency here—only four values are needed to specify a line which may be thousands of pels in total length. I call this a *lineto.*

The third primitive attempts to draw a smooth curve. While many routes exist, cubic splines might be a good choice. Certain cubic splines are also known as ***Bezier curves.*** A cubic spline is a pair of *x(t)* and *y(t)* polynomials, where *t* is a parameter which changes precisely from zero to one along the length of the generated curve.

You can think of *t* as time. Visualize a cubic spline as a certain 3D snake boxed into *xyt* space. Look into the end of your box, and you will see the x-y spline curve in two dimensions. Look into the box side and you'll see how y varies with *t*. Look down through the top to see how x varies with *t*.

Cubic splines draw almost any straight line, lots of graceful curves, and certain restricted curves with single loops, cusps, or inflection points. For fancier curves, any number of cubic splines can be linked end to end.

A cubic spline needs eight data points. Two of them are the already-known x_0,y_0 initial position information. A second pair at x_1,y_1

defines the location of the first influence point. The third pair (x_2,y_2) defines the location of the second influence point. A final pair sets an x_3,y_3 endpoint.

The endpoints of a cubic spline set the point at which the curve starts or finishes. The first influence point sets the direction and the enthusiasm the curve uses to launch away from its initial point. (Enthusiasm is also referred to as *tension* or *velocity.)*

The second influence point forces the direction and enthusiasm of the curve as it enters its final point. Influence points are usually well off the actual curve. I call a graphics primitive with two previous and six new data values a *curveto.*

A final (optional) primitive conveys the information needed to close the path back on itself. This information ensures that each joint in the path is treated equally. The path closure creates sparse data at best. No new data values are needed for closure! I call this a *closepath.*

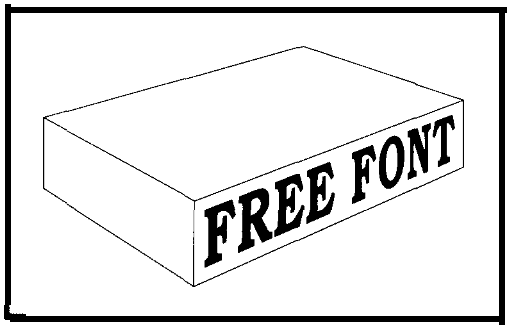


Figure 3—*The architect nonlinear* transform *creates an* architectural *perspective effect. Let x_0, y_0,* and z_0 *be the* distances from *the observer to the 0,0,0 perspecitve* origin, *where x is left-right,* y is in-out, and z is up-down. The basic *2-point perspective transform is $x' = y_0(x - x_0)/y + y_0)$ and $y' = y_0(z - z_0)/(y + y_0)$.*

Once you define a path using these four graphic primitives, you can build the path by suitable stroking, filling, shading, painting, tiling, or clipping. You can also use hundreds of high-level graphical operators, but they should internally reduce themselves to four absolutely positioned primitives.

To do a nonlinear transformation with graphics primitives, simply redefine the primitives to intercept and transform the data values. The new primitives might be defined as mt, *li, ct,* and *cp.*

An mt starts with two values, nonlinearly transforms them, and calls moveto. An li takes two data values, nonlinearly transforms them, and calls lineto. A ct accepts six new data values, nonlinearly transforms these values, and calls curveto.

Nonlinearly transformed sparse data may or may not be accurate everywhere. In general, if a nonlinear transformation maps a straight line into another straight line, sparse data is accurate. However, when the NLT maps a straight line into a newer curved line, sparse data could miss along the middle.

Let's look at two simple and very useful nonlinear transforms that end up accurate everywhere.

STARWARS

Surely, one of the most popular image distortions is the old Starwars effect shown in Figure 2. Think of this as drawing on a panel and then tilting the panel down.

Begin by defining a tilt angle θ such that 0” is “lying down” and 90” is “sitting up.” Then find a constant *k* called the *tilt factor:*

$$k = \text{fullheight} \tan \theta$$

The Starwars transform is:

$$x' = \frac{xk}{k+y}$$

$$y' = \frac{yk}{k+y}$$

Note that the zero x-axis routes down the center as shown. Add offset values to pick up an x slant to the left *or* right.

You handle lettering and typography the same way as lines and curves. Each letter is broken up into moveto, lineto, curveto, and closepath primitives and translated accordingly. Typography based on such sparse path descriptions is always preferable to bitmapped characters.

ARCHITECTURAL PERSPECTIVE

Architects usually don't use true perspective because buildings appear

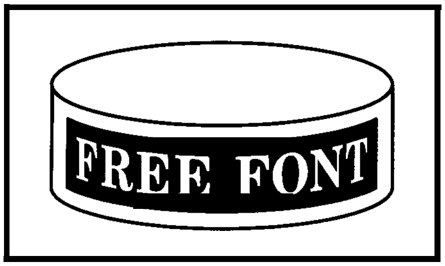


Figure 4—*The* tuna-can *nonlinear* transform is *also* useful *for grocery store ads and paint cans. Think of it as pasting a flat label onto a tilted cylinder. A tilt angle of 15° is shown* here.

wrong if their vertical lines are slanted. Instead, they apply a special two-point perspective in which all z-axis lines remain vertical, but x and y values diminish proportionally toward a pair of left or right vanishing points.

Figure 3 shows an example. Once again, the transform is surprisingly simple:

$$x' = \frac{y_0(x-x_0)}{(y+y_0)}$$

$$y' = \frac{y_0(z-z_0)}{(y+y_0)}$$

These $x_0, y_0,$ and z_0 values are the distance from observer to the O,O,O perspective origin. The basic NLT works point by point, transforming 3D points into 2D. Some redundancy and ambiguity is inherent in any perspective transformation which collapses three values into a pair of new ones.

To make the transform faster and more convenient, create a local transform that maps any flat plane into a designated "card" prepositioned in perspective space. For instance, if you draw a roof full of shingles, the entire roof gets picked up and rotated.

If you study the perspective math enough, one profound simplification pops out-most perspective mapping can be done by a linear transform! The only nonlinear parts divide by two identical $(1+y/y_0)$ factors.

As a general rule, do as much as possible with a linear transform, and only what you must with a NLT.

TUNA CAN

The tuna-can nonlinear transform shown in Figure 4 is especially useful for grocery-store ads or paint cans. With it, you're pasting a flat label onto a tilted cylinder.

Define a tilt constant k based on diameter D and tilt angle θ:

$$k = \frac{D}{2}\sin\theta$$

The tuna-can transform is:

$$x' = \frac{D}{2}\sin\left(\frac{114.591x}{D}\right)$$

$$y' = y - k\cos\left(\frac{114.591x}{D}\right)$$

While you can use the tuna-can transform in an isometric drawing, a shallower tilt angle often gives more pleasing results.

DON'T CUT CORNERS!

With the Starwars or perspective NLTs, straight lines remain straight. This is also true for some other NLT mappings. On the tuna can, however, only a vertical line stays straight. Horizontal or slanted lines go around the can, not through it!

If you throw any old art at your tuna-can NLT, corner cutting may result from sparse data. The transform specs offer only four lineto endpoints and eight curveto control points. Intermediate points are unpredictable. Since the computational penalties for not using sparse data are severe, you need to find tolerable workarounds instead of remapping each point.

Other nonlinear transformations create corner-cutting problems. Such problems occur whenever a straight

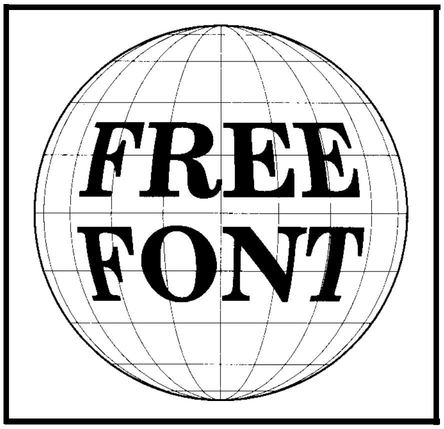
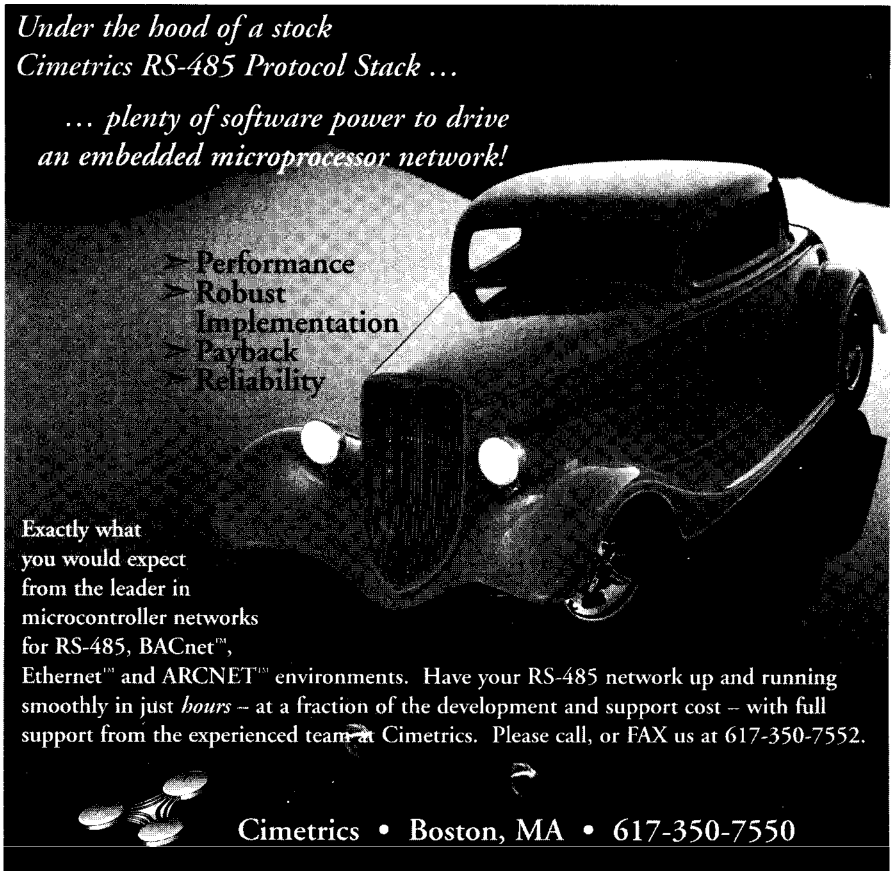


Figure 5—*The spherical nonlinear transform can paint any image onto a globe. For a longitude of x°, a latitude of y°, and a unit radius sphere, here's the transform: x' = sin(x) cos(y), y' = sin(y).*



line is curved on final mapping.

Several tricks can easily be automated to avoid corner cutting on typical input art. However, each corner-cutting avoidance trick costs in computing time and may increase file length. In general, use minimum repairs consistent with an acceptable final image. If a cut corner is small and doesn't look too bad, you should probably use it as is.

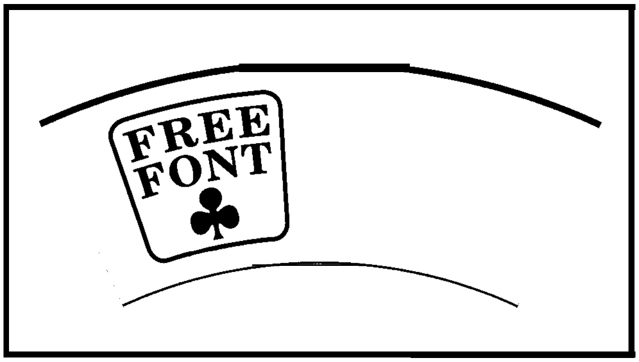


Figure 6—*The rootbeer nonlinear transform can be used to design paper drink cups and megaphones. Precalculate y_0, the vertical distance from the bottom of the cup to 0,0 or the point of conic origin. Find a current transformation angle θ. Then, nonlinearly transform $x' = (y t y_0) \sin \theta$ and $y' = (y t y_0) \cos \theta$.*

There is no corner cutting with moveto. Position is position. The worst corner cutting often accompanies closepath. For example, using closepath to complete the fourth side of a large square may cause severe corner cutting.

One way to deal with closepath corner cutting is to create artwork so that closepath never extends far. You can also intercept all input closepaths and replace them with a new lineto followed by a closepath.

Short lineto primitives often produce acceptable results. Medium ones might need some repairs, while long ones definitely need mods.

The first defense against lineto corner cutting is to replace a lineto with a single curveto. A spline with its first influence point ⅓ of the way along its straight-line path and the second influence point ⅔ of the way along creates a smooth curve that at least starts and ends in the correct direction.

But, the replacement spline still may miss in the middle, sometimes by a lot. To get around this problem, first split a lineto into a grouping of sequential lineto primitives aligned end to end. Then, convert each shorter primitive into a "one-third, two-third" cubic spline.

As few as four subssplines minimize the worst corner cutting. For a larger mapping, more subssplines are better. As usual, penalties include higher computation times and longer file lengths. In general, if a NLT has potential corner-cutting problems, set up an error tolerance that depends on the length and direction of the lineto.

Long curveto primitives can also cut corners, but not as badly as lineto or closepath. If necessary, split a long curveto into several smaller splines by replacing a long curveto with several sequential lineto approximations. Then, replace the shorter linetos with "one-third, two-third" splines.

To recap, some nonlinear transforms map straight lines into curved ones. To avoid corner cutting at plot time, pick only short closepath primitives and replace lineto primitives with one or more curveto primitives. In extreme cases, also subdivide long curvetos.

COMPILING FOR SPEED

Extensive calculations and corner-cutting routines need doing only at image-creation time. You can easily apply compiling techniques to save only the results of NLTs for later use.

The compiled or distilled code is simply a lot of fast-running moveto, lineto, curveto, and closepath operators. Compiled code can be linearly transformed to change size, rotation, or repetition or be exported elsewhere. There's no need to tow along custom or oddball fonts when all fonts are replaced by equivalent NLT paths.

SPHERICAL MAPPINGS

Figure 5 shows a spherical NLT. Use this one to paint any image onto a globe (i.e., world maps, fisheye effects, volleyballs, or balloons).

It's convenient to use latitude and longitude, having 90" west longitude define the left circle side, 90" east longitude the right side, 90° north latitude the top, and 90" south latitude the bottom. I use the convention of north and east defined positive and south and west, negative. I also assume larger values that would end up on the back side of the sphere are clipped or truncated.

Here's the spherical NLT:

$$x' = \sin(\text{long}) \text{ x } \cos(\text{lat})$$
$$y' = \sin(\text{lat})$$

This expression gives a sphere of unit radius, given inputs in degrees. Such results can be easily scaled. Major defenses against corner cutting are certainly needed-replace long lineto primitives with shorter end-to-end curvetos.

THE ROOTBEER TRANSFORM

On your next break, take a close look at the paper cup. Observe how the artwork gets fatter as the diameter increases. Take the cup apart and flatten it out. Note the truncated conical shape.

The rootbeer transform shown in Figure 6 can be used to design paper drink cups and megaphones. The x values map tangentially along an arc set by the current diameter. The y values plot radially along the vertical line set by the present angular position. The transform first finds y_0, the vertical distance from the bottom of the cup to the origin point:

$$Y_0 = \frac{\text{heightx bottom}}{\text{top-bottom}}$$

For *bottom* or *top, you* can use a radius, circumference, or diameter (but

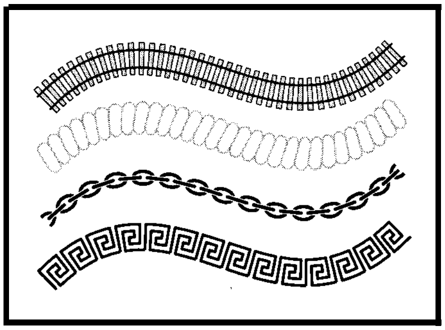


Figure 7—*The glyphpath nonlinear transform hand/es curves and corners in fancy border designs such as those shown here.*

be consistent). Next, find a current angle θ:

$$\theta = \frac{57.2958x}{Y + y_0}$$

This equation is a cleverly disguised $s = r\theta$ arc in degrees. Finally, we get:

$$x' = (y + y_0) \sin \theta$$
$$y' = (y + y_0) \cos \theta$$

One gotcha: that y_0 value to the origin can be rather large. Thus, the origin may end up off your page.

GLYPHS ALONG A PATH

For border artwork, you need ways to handle corners and closures cleanly. As border elements round a curve, the individual glyphs should compress on the inside of the curve and stretch on the outside.

The glyphpath transform appears in Figure 7. It can be used for fancy borders, rope effects (including knots and even rope signatures), model railroad layouts, chains, cords, braiding, and game paths.

Nonlinearly transformed x values go along the underlying path, while y values sit normal to the path. Thus, x values should walk along the path with you. The y values are always at your side: positive y on your left and negative y on your right.

Let's assume that the original path is a single cubic spline. Longer paths can use multiple splines. Each position on any cubic spline has an underlying value *t* associated with it which ranges 0-l along the spline. Sadly, *t* is not linearly proportional to spline position. So, *t* values run faster along the more bent portions of a curve.

A successive approximation finds an initial *t* value for the origin of your current glyph. It is assumed that *t* is nearly linear with the length inside any given glyph. Thus, all glyph x values are scaled to an initial *t* plus a fraction *At* proportional to glyph width. A linear delta is assumed.

Fortunately, you only need a successive approximation once for each glyph position. To do the transform, first find the *t* value that corresponds to x. Then, calculate your current on-path position:

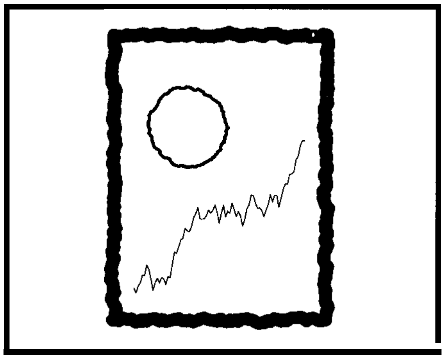


Figure 8—*The* scribble nonlinear transform introduces randomness *and* variation *into an* image by *rep/acing* solid lines with fuzzy ones. You can set the *fuzz factor from a slight hint of* **rattiness** *to drunken wandering.*

$$x_{path} = At^3 + Bt^2 + Ct + D$$
$$y_{path} = Et^3 + Ft^2 + Gt + H$$

Values *A-H* relate to the spline control points.

Next, find the slope of the curve and the angle of a normal slope vector:

$$\theta = 90 + \tan' \left(\frac{3Et^2 + 2Ft + G}{3At^2 + 2Bt + C} \right)$$

The glyph transform is:

$$x' = x_{path} + y(\cos \theta)$$
$$y' = y_{path} + y(\sin \theta)$$

The glyphpath transform works best with fairly narrow glyphs. If you venture too far from the underlying path, a glyph may turn itself inside out on sharp turns. Strive for a balance between glyph size and tightness of turns. To get fancy, you can alternate glyphs along your path, which is one way to do multicolor braiding.

THE SCRIBBLE TRANSFORM

One big complaint about computer art is that it looks as if a computer did it. There's often a need to introduce randomness and variation into an image. The scribble NLT of Figure 8 replaces solid lines with fuzzy lines. You can set the fuzz factor from a slight hint of rattiness to drunken wandering.

To apply the scribble transform, first reduce all elements in your path to short line segments of usable accuracy. Then, subdivide each segment into *n* resolvable steps. For each step, calculate the rattiness factor:

$$newrat = (oldrat + \text{random bipolar offset})(\text{homing instinct})$$

Next, plot a short line segment from your last value to a new point offset normally from the true line by the new ratticity value.

A random bipolar offset is obtained by centering and adjusting a random number. For instance, values in the range -3.45 to +3.45 might end up suitable. The scale factors selected set the violence of the variations.

One problem with random walks is that they sometimes end up wandering further and further astray from the intended path. The solution is to add a homing instinct that multiples the accumulated error by some value slightly less than one. This adjustment produces a software high-pass filter that stomps on long-term variations while passing the desired shorter ones.

Still, the scribble transform can't guarantee total path closure. If a path must close, select a different random seed until you get one that gives a tight enough closure.

FOR MORE INFORMATION...

The nonlinear graphic transforms I've shown can be done in nearly any language and on virtually any platform. I've found the PostScript general-purpose computer language to be a fast, powerful, fun, and friendly tool for exploring all graphical transforms. In particular, it lets you zero in on the transforms themselves and their results.

Several files have been posted to the Circuit Cellar BBS and to GEnie PSRT that give detailed NLG utilities. Lots more cubic spline info is included. ❑

Don Lancaster is the author of 33 books and countless articles. In addition to offering his own books, reprints, and various services, Don offers a no-charge technical helpline at (520) 428-4073. He may also be reached at synergetics@genie.com.

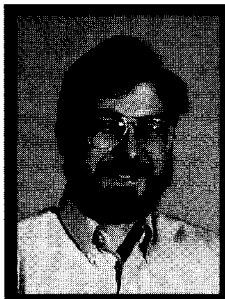
IRS

416 Very Useful
417 Moderately Useful
418 Not Useful

FIRMWARE FURNACE

Ed Nisley

Part 1: Getting Vid-Link in Sync



Ed updates an ancient project—the ImageWise receiver board—by adding HCS TV status display. In this first article, Ed looks at how the hardware generates video sync. His emphasis: wringing precise timing from overworked hardware.

Good projects never die. About a year ago, Steve dropped two irrefutable facts in my ear: he had a bunch of ImageWise receiver boards on the shelf, and you folks wanted an HCS TV status display. Solving both problems, as he put it, was a simple matter of software.

He laid down only one ground rule. I could do anything I wanted, as long as the new code fit into the same 2764 EPROM. Changing anything else would be an uphill battle: he wanted a true plug-and-play brain transplant!

This month, I'll review the (meager) ImageWise hardware and discuss how the Vid-Link code generates video sync. Next month, I'll explain character generation and how the network interface tucks 250 bytes of data into a 16-byte buffer. In each case, the emphasis falls on ways you can wring precise timing from overworked hardware.

WHAZZIT?

HCS status displays date back to the LCD-Link in *INK 27. /Editor's note: Circuit Cellar Inc. will be releasing a new and improved LCD-Link with a keypad in the first quarter of 1996.]* Two issues later, Bill Houghton described a TV-Link video overlay board that used a specialized Signetics/

Philips microcontroller and several other unobtainable parts. As more folks built bigger HCS programs, those widgets simply couldn't handle the job.

Besides, as Steve pointed out, all home-control junkies have big TV sets. It goes with the territory. Nobody squints at HCS info on a teensy LCD panel, at least not if they can avoid it. They want a Home Status Network right there on the pixellated screen.

Well, TVs come in all sizes, and Vid-Link provides characters to suit. The small, 8 x 8-dot font puts 30 characters on 28 rows, which is just right for complex status displays. The large, 16 x **16** font allows only 15 characters on 14 rows, making a Picture-In-Picture window readable from across the room. With a bit of care, you can even mix the two character sizes on the same screen.

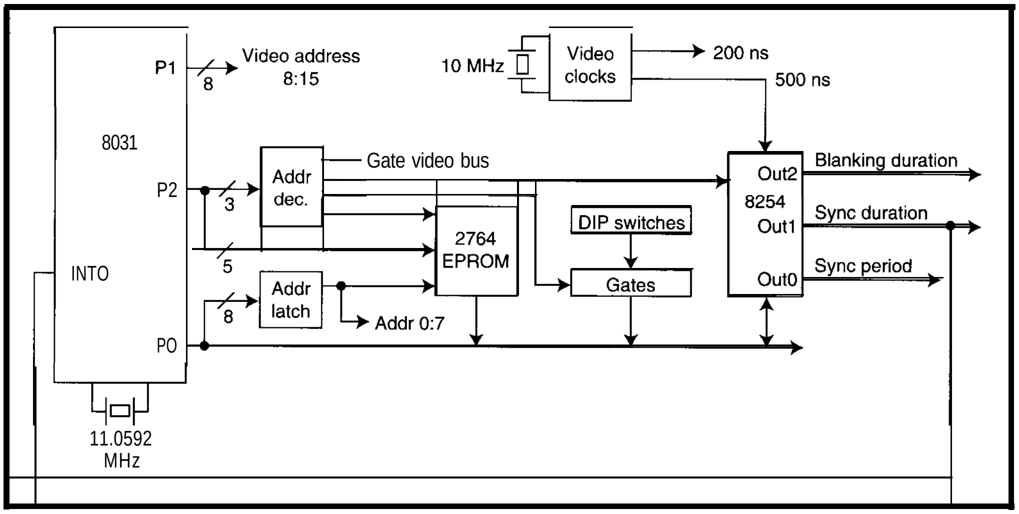


Figure I--The *Vid-Link system uses a simple 8031 CPU with an 8-KB EPROM. An 82.54 timer generates the pulses* required *for stable* video sync.

The ImageWise hardware supports grayscale video, so Vid-Link can display high-intensity and reverse-video text (see Photo 1). Sad to say, the hardware just doesn't support blinking characters or a hardware cursor. Next month, you'll see why those features won't work in firmware either.

Like the LCD-Link, Vid-Link supports a useful subset of the familiar ANSI cursor-positioning and screen-control commands. It also accepts control characters in more or less standard C notation: you can send ASCII 27, the escape character, using the string \ e, a tab with \ t, and a newline with \n.

ImageWise has an eight-position DIP switch that determines its startup options. It lacks other digital inputs and thus, Vid-Link has no commands to read them. Think of it as a write-only device and you'll be spot on.

Serial I/O passes through a standard female DB-25 connector on the back panel. You'll need an external RS-485 converter to drive it from the HCS network. Steve had a great stash of ±5- and +12-V triple-output supplies when we built the ImageWise, which means you'll need more than the usual +5 V. Nope, it does not use MAX232 chips.

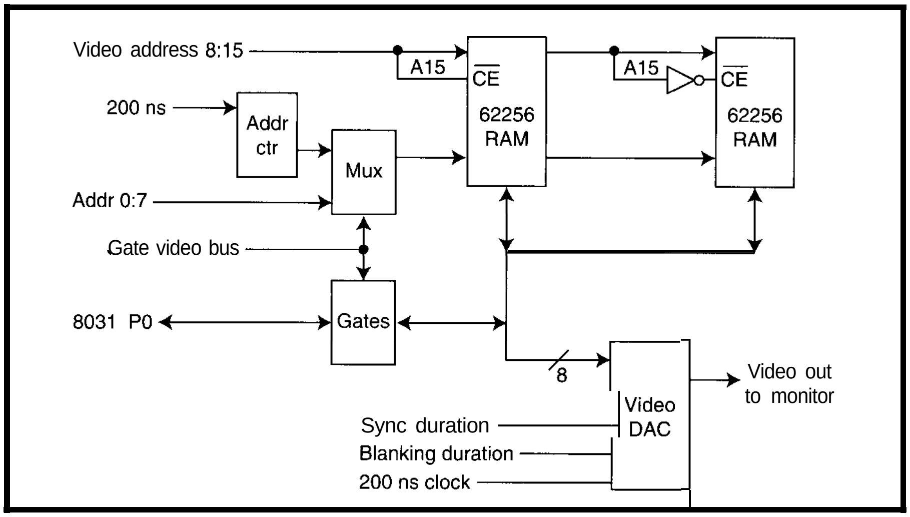


Figure 2 --*The TV* image displays data *stored* in the *video RAM buffer. Each RAM byte* corresponds to a sing/e dot on the *screen, with 256 bytes per line and about 242* lines per screen.

When Steve got the early prototype firmware, he immediately rushed into the Original Circuit Cellar and wired it into his whole-house video-distribution system. It sure kept him occupied for awhile..

Now you know what the Vid-Link does, I can discuss how we got there.

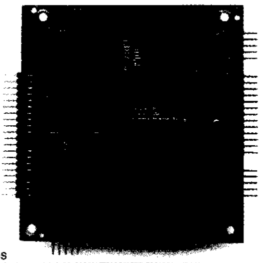
SIZING UP THE SITUATION

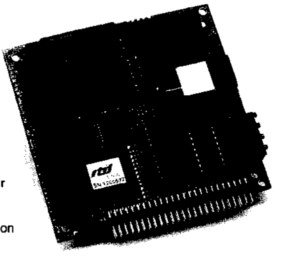
The ImageWise project was reprinted in Steve's *Best of Ciarcia's Circuit Cellar.* A little fast calculator work reveals the decade between then and now.

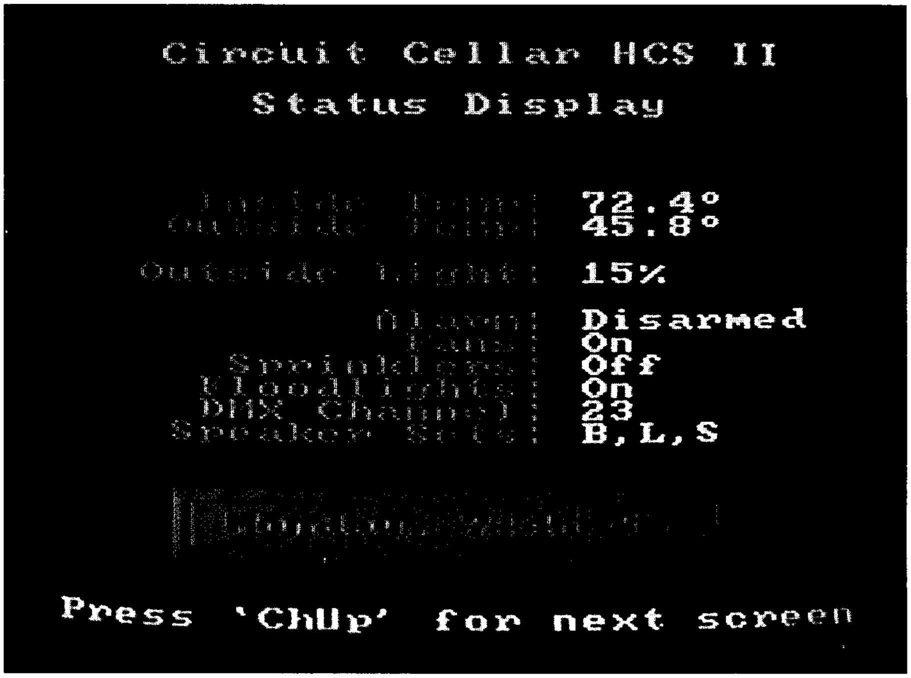
Should your memory need refreshing, the ImageWise system had two separate units. The transmitter flash-digitized a video field from any standard monochrome source and encoded it into serial data over a standard RS-232 link. The receiver converted that data back into a grayscale video image on a standard TV monitor. You could use the two boards independently, connect them over a phone line for remote viewing, or put a PC in series for image processing.

I don't have room for a tutorial, so if you're a little shaky on video fundamentals, those articles will get you up to speed. Time to hit the books!

Figure 1 shows the computer part of the ImageWise receiver. You'll recognize a standard 803 1 layout with firmware in a 2764 EPROM. The address decoder assigns 8-KB blocks to the EPROM, 8254 timer, video buffer, and even the DIP switches. Yes, the same







Circuit Cellar HCS II
Status Display

Inside Temp: 72.4°
Outside Temp: 45.8°

Outside Light: 15%

Alarm: Disarmed
Fans: On
Sprinklers: Off
Floodlights: On
DMX Channel: 23
Speaker Sets: B,L,S

Press 'ChUp' for next screen

Photo 1—The *Vid-Link* allows both normal *and bold characters* in addition *to full cursor* positioning, making *complete,* informative displays easy

DIP switch appears at every one of 8 192 addresses since there's no reason to decode it any more closely.

The 8254 timer controls all of the high-speed video timing with a resolution of 500 ns. Channel 0 determines the time between successive sync pulses. Channel 1 controls the width of each sync pulse. Channel 2 sets the delay from the start of the sync pulse

Field	Line	Sync#	Nom Pd	Err	Pulse	Count	Sync Event	Sync Handler	Char	Row
2	262	542	63.50		4.7	241	Last active	captured		
1	1	1	31.75	-1	2.3	1	Equalize 1 Start	captured		
1	1	2	31.75	-1	2.3	2		captured		
1	2	3	31.75	+1	2.3	3		reserved		
1	2	4	31.75	+1	2.3	4				
1	3	5	31.75	+1	2.3	5		VidVSync		
1	3	6	31.75	+1	2.3	6	Equalize 1 End	captured		
1	4	7	31.75	+1	27.1	1	v sync 1	reserved		
1	4	8	31.75	+1	27.1	2				
1	5	9	31.75	+1	27.1	3				
1	5	10	31.75	+1	27.1	4				
1	6	11	31.75	+1	27.1	5		VidEqShort		
1	6	12	31.75	+1	27.1	6	V Sync End	captured		
1	7	13	31.75	-1	2.3	1	Equalize 2 Start	reserved		
1	7	14	31.75	-1	2.3	2				
1	8	15	31.75	-1	2.3	3				
1	8	16	31.75	-1	2.3	4		VidF1_Blank		
1	9	17	31.75	-1	2.3	5		captured		
1	9	18	31.75	-1	2.3	6	Equalize 2 End	captured		
1	10	19	63.50		4.7	1	V Blank Start	reserved		
1	11	20	63.50		4.7	2				
1	12	21	63.50		4.7	3		Vid_DataPump		
1	13	22	63.50		4.7	4		Pump		
1	14	23	63.50		4.7	5		Pump		
1	15	24	63.50		4.7	6		Pump		
1	16	25	63.50		4.7	7		Pump		
1	17	26	63.50		4.7	8		Pump		
1	18	27	63.50		4.7	9		Pump		
1	19	28	63.50		4.7	10		Pump		
1	20	29	63.50		4.7	11	V Blank End	Pump		
1	21	30	63.50		4.7	0	Fully active	reserved		
1	22	31	63.50		4.7	1				
1	23	32	63.50		4.7	2		VidF1_Active		
1	24	33	63.50		4.7	3		captured		
1	25	34	63.50		4.7	4		reserved		
1	26	35	63.50		4.7	5				
1	27	36	63.50		4.7	6				
1	28	37	63.50		4.7	7				
1	29	38	63.50		4.7	8			0	
1	30	39	63.50		4.7	9			0	

Figure 3—A spreadsheet *organizes fhe minutiae associated with each video sync pulse. The section shown here covers the last line in Field 2 through the first character row in Field 1.*

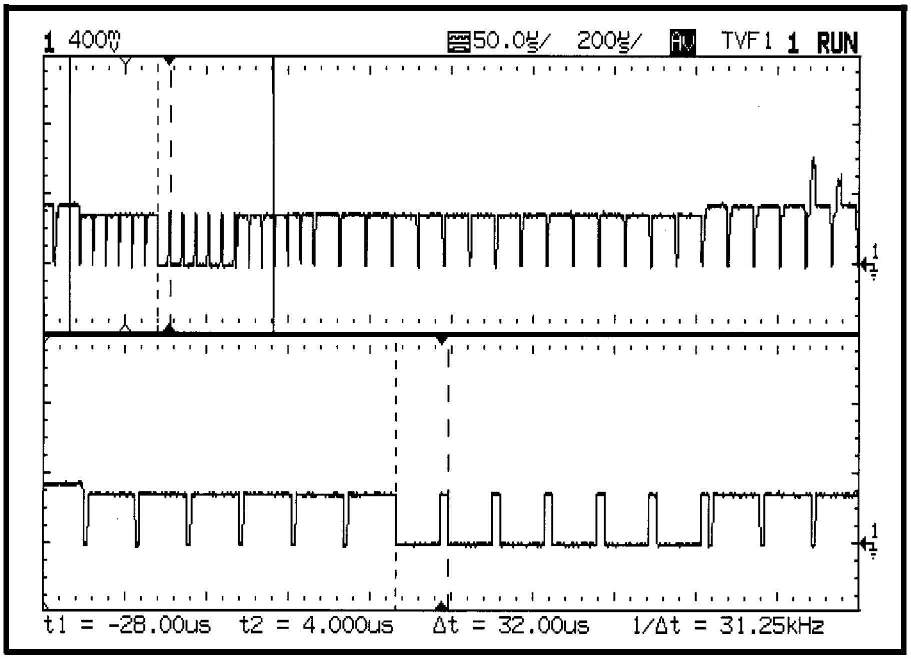


Photo 2—*Equalizing pulses occur at twice the normal horizontal-line rate. The upper trace shows the Field 1 vertical blanking interval, with the start of the vertical sync pulse magnified in the lower trace. The firmware must reprogram the 8254 timer whenever the pulse period or width changes.*

until the beginning of the visible part of the line, a duration known as the *blanking interval.*

The video side exhibits just slightly more complexity, as you can see in Figure 2. Addresses for the 64-KB video buffer have two parts. The low byte comes from an 8-bit counter chain clocked at 5 MHz that sends 256 bytes to the video D/A converter during each scan line. The CPU supplies the high-order video-address byte from port P1, which means the firmware must track each video line and update P1 during the horizontal blanking interval.

The original article discussed the subtleties that kept all the timings correct. I've omitted most of the control logic and lines from these figures to reduce the clutter. Check the articles for complete details before firing up your soldering iron.

The video D/A converter changes each byte from the video buffer into an analog voltage, with 00h becoming black and FFh appearing as glare white. Signals on the D/A converter's Sync and Blanking pins produce the proper voltages during the invisible parts of each scan line. A video display makes debugging easy-you can see your mistakes in real time!

The address decoder in Figure 1 connects the 803 l's data bus to the video buffer during accesses in an 8-KB block between 2000h and 4000h. The CPU emits the high byte of that 16-bit address on port P2, which drives the decoder and enables the RAM output. The high byte of the video address, however, still comes from port P1.

Accessing a video-buffer byte thus requires four steps. You must put the high-order address byte into P1, the low-order byte in DPL, and load 20h into DPH. An ordinary MO V X instruction then activates the decoder, connects the buses, and reads or writes the correct byte. The 8-KB address block holds 32 duplicates of the same 256 video-buffer bytes.

If that hardware could produce grayscale, surely it can handle characters!

KEEPING VIDEO'S 0th LAW

Having worked on several video projects, I know with certainty: Thou Shalt Maintain Rock-Solid Sync. Your programming skill shines directly in your customer's face. Both of you can watch your mistakes in real time.

The Vid-Link firmware must handle two tasks: generating sync and parsing serial data. Both have hard real-time requirements with slightly different time scales. Interleaving the two turns out to be more complex.

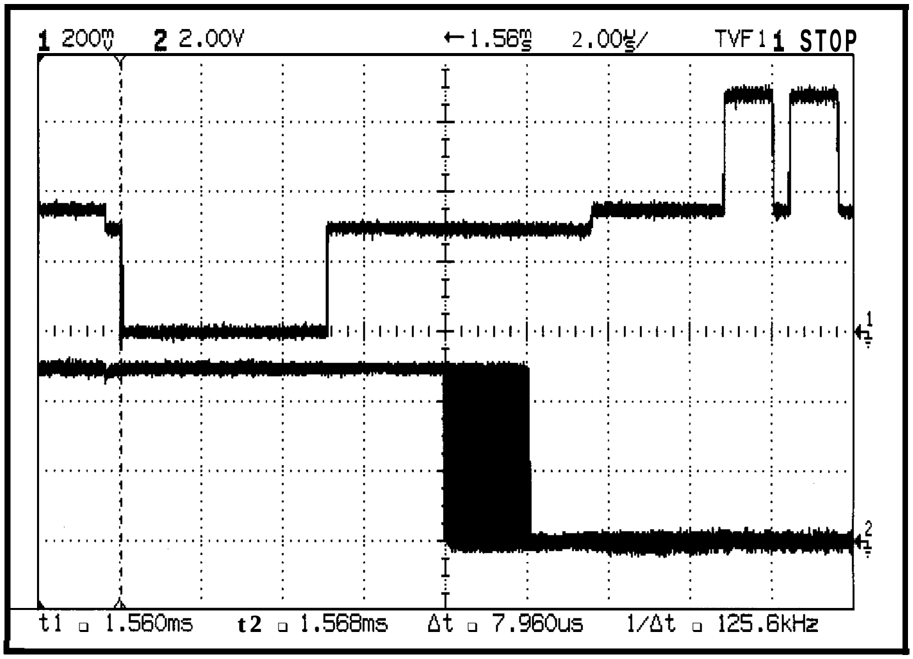


Photo 3—*The falling edge of the horizontal sync pulse in the upper trace triggers an* interrupt *routine that* updates *the video buffer* address on *port P1. This* multiple-trace record shows an *8-µs* delay and about 2 *µs of jitter in the port output. The small step in fhe video signal just affer the port update marks the transition from blanking to fhe black background behind each character.*

After a bit of doodling, I tabulated every NTSC sync pulse in a spreadsheet. Figure 3 shows the values for everything from the last line in Field 2 through the top of the first character row in Field 1. The first two columns show each sync pulse by field and scan line. The third column numbers each pulse, starting at the top of Field 1.

The Nom Pd and Pulse columns give the nominal time until the next sync pulse (measured between leading edges) and the width of each pulse. I started from first principles and compared the results with ImageWise. As near as I can tell, the two widgets differ by one sync pulse.

I think I've got it right this time because my HP54602B scope displays each sync pulse's NTSC line number. It turns out that TV monitors forgive consistent errors and react harshly to missing pulses or random jitter.

In any event, Photo 2 shows the spreadsheet data in action with the Field 1 vertical sync pulse magnified in the lower trace. The vertical cursors mark Sync 7, a 32.00-µs equalizing pulse at the start of the vertical sync pulse. Notice that the spreadsheet calls for a 31.75-µs period. Whoops....

The 500-ns clock driving the 8254 timer produces 63.5-µs horizontal sync pulses with spot-on accuracy. Getting the equalizing pulses right requires a 250-ns clock that you just can't get from a 10-MHz crystal oscillator. Doubling the oscillator frequency would require another flip-flop (to divide it back down) and another DIP on the board.

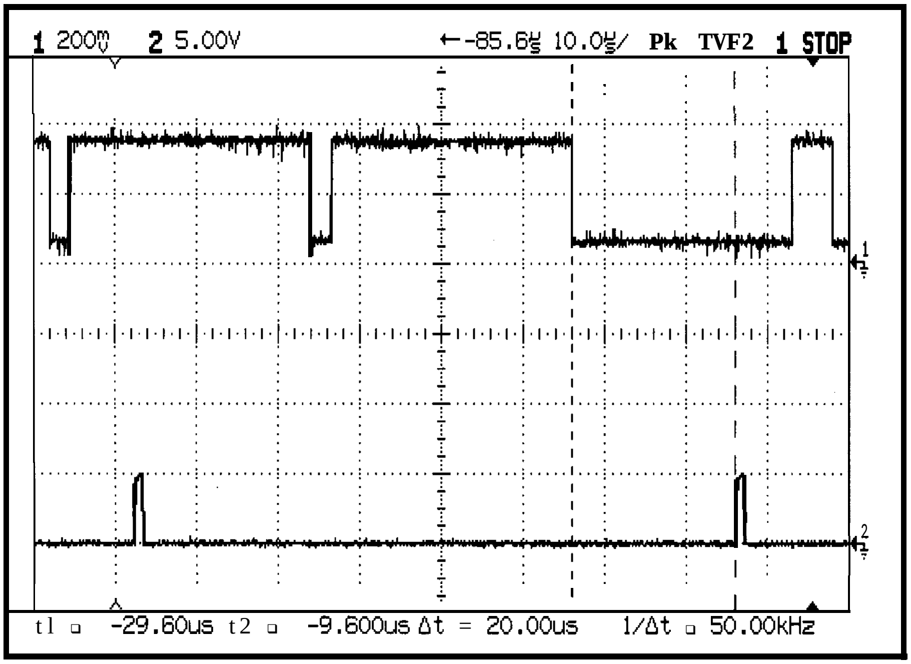


Photo 4-*The* vertical *sync routine gets* control *after* the first *sync pulse in Trace 1, shown by* the *special debugging output in Trace 2.* It *waits* until the *second sync,* sets the *8254* to *produce* vertical *sync timings, fhen executes an ordinary RETI. Those few instructions delay the* start *of* the next *interrupt about* 20 µs after the falling *edge of* the *vertical sync.*

The solution may seem like cheating, but both ImageWise and Vid-Link use equal numbers of 3 1 .50-µs and 32.00-µs equalizing pulse periods to get the right value, at least when averaged over an entire field. Be small errors occur only during vertical retrace, long before the visible part of each field, the monitor remains stable. The Err spreadsheet column shows the 250-ns units. Formulas elsewhere in the spreadsheet accumulated the total error and kept me honest while I experimented.

Similarly, the nominal pulse widths in the Width column become multiples of 500 ns after passing through µs instead of 2.3, 5.0 instead of 4.7, and so forth. These values fall slightly outside specification limits, but lie well inside the OK category. They're much better than some I've seen and worse than others sporting dedicated video-timing chips.

Bear in mind you're looking at a decade-old design optimized for cost. Those fancy chips either didn't exist or carried a prohibitive price tag. Yes, we'd all do things differently today.

A TICK IN TIME

All three 8254 channels operate in Mode 1, producing a repetitive series

Listing 1—*This interrupt handler loads the proper line number info port P1 on each sync interrupt during the active video time. The MO V instruction setting P1 occurs about 8 µs after the leading edge of the sync pulse, producing the results you see in Photo 2.*

```
IntOHandler(void) {
  asm {
?IntOhere EQU *
    ORG   $0003             * get to the external int vector addr

    JNB   B_VidEnable,?IntOP1 * don't change P1 if video is OFF
    MOV   P1,VidActiveLine *  set high address byte for this line
?IntOP1
    INC   VidActiveLine     * always maintain this count!

    JNB   TFO,?IntOBail     * bail out if timer is still running

    PUSH  PSW               * save bystanders
    PUSH  A
    PUSH  DPL
    PUSH  DPH

    MOV   A,VidHandlerID    * dive into the current handler
    MOV   DPTR,#HandlerTable
    JMP   [A+DPTR]

?IntOBail
    RET1   * otherwise, keep on trucking

    ORG   ?IntOhere
```

of pulses after initialization. The maximum countdown values fit neatly into a byte (63.5 µs = 127 x 500 ns) and, strangely enough, the 8254 has an LSB-only mode that zeros the high byte whenever you write the low byte.

Writing a new countdown value into an 8254 channel sets the pulse width produced *after* the current pulse times out. For example, the firmware must write the period, width, and blanking values for Line **1** of Field 1 during Line 262 of Field 2.

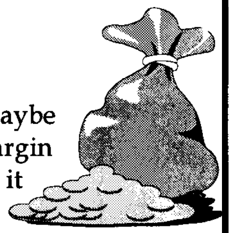
To save another DIP, the ImageWise hardware doesn't include the CPU's *RD and *WR signals in the address decoding. It also drives the RAM ● CE inputs with address A15 and *A15. As a result, whatever you write into the 8254 also goes into the video buffer at the current address. You can disable the RAM data outputs, but one of the two chips remains active at all times.

Once again, firmware comes to the rescue. Simply aim P1 at an invisible part of the buffer before writing anything into the 8254. In effect, I designate one 256-byte block of RAM as a write-only spoil area. Not ecologically sound, but.. .

Gaining control at the right time poses a problem. An 803 1 with an 11.0592-MHz crystal clicks off an instruction every microsecond or two. I tested several variations, only to conclude that spending nearly 40 µs in the sync interrupt handler made no sense! Vid-Link code must accomplish much more in response to the serial link than the original ImageWise firmware, as you'll see next month.

The interrupt handler in Listing 1 starts up at each sync pulse and loads the current line number into port P1. The mainline code prevents the port update by clearing the B_VidEnable bit variable, so a single JNB precedes the actual output.

Photo 3 illustrates how long a microsecond lasts when you're in a hurry. The falling edge of the video signal in the top trace triggers the 803 l's interrupt hardware and activates the code in Listing 1. The lower trace shows a bit in port P1, with the black blob marking the earliest and latest outputs. That MOV instruction occurs at



some random time between 8 and 10 µs after the sync pulse!

The 803 l's Timer 0 provides a simple hardware assist for the interrupt handler. The firmware loads the timer with the delay until the next video event, clears TFO, and starts the timer. The delay may be as long as the entire visible part of the screen or as short as a few half-lines during the vertical sync pulse.

As long as the handler finds TFO clear, it simply bails out without saving any registers. The few additional instructions bring the total handler to about 15 us, nearly a quarter of the active line time. That's the best I could do: five instructions, executed once per line.

When Timer 0 ticks through zero, TFO goes high and the interrupt handler sets up for real work. Perhaps surprisingly, saving a few registers and branching through, the Hand1erTab1e (Listing 2) requires about 24 µs from the sync pulse's leading edge. Recall that equalizing pulses occur every 3 1.5 µs.... Getting ready takes all the time!

The dozen instructions that return from the handler blot up another 20 µs or so, pushing the overhead to nearly a whole line. Remember that dropping a new crystal or CPU into the Image-Wise board wasn't an option.

Line syncs, however, come along like clockwork. If the interrupt handler can get control on the sync pulse before it produces output, it can simply poll the interrupt input, wait for the next sync, then do whatever it likes, right on time. Listing 3 shows VidVSync, the routine that creates the vertical sync pulse in each field.

Look at the spreadsheet in Figure 3 again. Line 6 shows that VidVSync starts up after sync 5 in Field 1 and captures the next pulse. The code in Listing 3 has just enough time to save and set P1 before polling IEO, the interrupt triggered by the sync pulse.

The 8031 hardware clears IEO immediately before branching to the interrupt handler. The Wait0N macro contains a single-instruction JNB loop that spins while IE0 remains low. When the sync pulse goes low, the hardware sets IEO and the loop terminates. I added a CLR to shut the bit off

Listing 2—The interrupt handler enters this code when TFO goes high. Each entry in Hand7erTab7e corresponds to a routine that writes new values info the 8254 and twiddles output bits in preparation for the next group of identical video lines.

```
VidIntHandlers(void) {

asm {
HandlerTable
    AJMP   VidIdle       * 0 idle line handler, nothing happens
    AJMP   VidVSync      * 1 start vertical sync pulse
    AJMP   VidEqShort    * 2 set equalizing pulses 1 count short
    AJMP   VidF1_Blank  *  3 start field 1 vertical blanking
    AJMP  VidF1_Active *  4 last blank line atop field 1
    AJMP   VidF1_End     * 5 last full line in field 1
    AJMP   VidF2_Blank  *  6 set up for last equalizing pulse
    AJMP  VidF2_Active *  7 last blank line atop field 2
    AJMP   VidF2_End     * 8 last active line in field 2
    AJMP   Vid_DataPump *  9 serial data pump

*--- common return point for all routines
*     we prepare for the next state,
*      which takes effect when Timer 0 ticks

VidReturn
    INC    VidStateID    * step to next state
    MOV    A,VidStateID * to get handler ID and rep count
    MOV    DPTR,#VidStates     * ... from state table
    MOVC   A,[A+DPTR]

    RL     A             * convert to handler table index
    MOV    VidHandlerID,A

*--- return from this interrupt

    POP    DPH           * restore bystanders
    POP    DPL
    POP    A
    POP    PSW
    RET1                 * supplants the normal function RET

}
}
```

Listing 3-This routine gains control two equalizing sync pulses before the vertical sync pulse begins. The WaitON macro stalls until the leading edge of the next sync pulse sets IEO, then c/ears the bit. The remaining instructions thus begin within a few microseconds of the correct sync.

```
VidVSync

  LoadT0 5*T0_HALFLINE-T0_QUARTERLINE * set delay to 2nd eqs

  PUSH   P1                    * set P1 to junk line
  MOV    P1,#VID_JUNKLINE

  WaitON IEO                   * capture last equalizing pulse

  SETB   TRO                   * start delay timer

  MOV    DPTR,#I8254_DURATION  * set up vertical sync width
  MOV    A,#SYNC_VSYNC
  MOVX   [DPTR],A

  POP    P1
  AJMP VidReturn
```

in preparation for the RETI. That's all it takes to stay in sync.

The spreadsheet (Figure 3) shows sync 7, the first one in the vertical sync pulse itself, as reserved. To see why, trace the code from WaitON in Listing 3 through VidReturn in Listing 2, then simulate another interrupt in Listing 1 to reveal 30-odd instructions. That adds up to roughly 40 µs at 11.0592 MHz.

Even though the 8031 has easily predicted instruction times, one careful measurement beats 1000 opinions. The lower trace in photo 4 shows a debugging output produced just before the PUSH instructions in Listing 1. The first blip shows the normal 10-µs delay after the equalizing sync.

The second blip, however, occurs slightly more than 50 µs from the falling edge of the second equalizing pulse-precisely the time required to exit the first interrupt handler and enter the second. The falling edge of the vertical sync pulse just right of center screen triggers that laggard interrupt. Got it?

All this means you must consider both interrupt latency and the path length through your code when you must handle high-speed, periodic interrupts. Don't assume that every interrupt handler starts on time just because you sweated bullets getting the front-end latency under control.

RELEASE NOTES

If you still have copies of Steve's past projects, start digging. If your collection doesn't go back to '87, the ImageWise columns also appear in Steve's ***Best of Circuit Cellar.***

K. Blair Benson's ***Television Engineering Handbook*** tells you more than you care to know about TV. My copy dates back to '86 and actually mentions HDTV. I presume newer editions give the grisly details.

Next month, a look at character generation and the network interface. Think small! ■

Ed Nisley** (KE4ZNU), **as Nisley Micro Engineering, makes small computers do amazing things. He's also a member of Circuit Cellar INK's engineering staff. You may reach him at** ed.nisley@circellar.com **or** 74065.1363@compuserve. **corn.

SOURCE

Vid-Link, LCD-Link, ImageWise
Circuit Cellar, Inc.
4 Park St.
Vernon, CT 06066
(860) 8752751

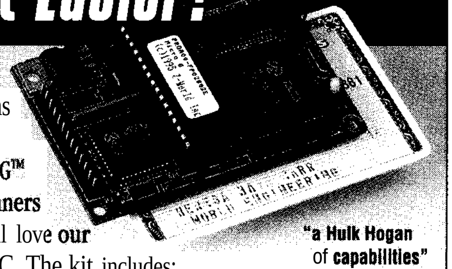
REFERENCES

Benson, K. Blair. ***Television Engineering Handbook.*** McGraw-Hill: New York, NY. 1986. ISBN 0-07-004779-O.

Ciarcia, Steve. ***Best of Ciarcia's Circuit Cellar.*** McGraw-Hill: New York, NY. 1992. ISBN 0-07-011025-5.

IRS

419 Very Useful
420 Moderately Useful
421 Not Useful

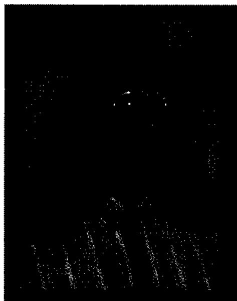


FROM THE BENCH

Jeff Bachiochi

Programmability without Volatility

Ditch Those Back-m Batteries



Jeff's in the market for fast static RAM. His find: Simtek's NVSRAM, a chip which brings together the speed of SRAM and the stability of an EEPROM's nonvolatile storage. Jeff shows us how this memory eases project implementation.

Yup! This new micro is really a screamer. Why if it wasn't for this slow memory I have, it would probably finish before it got started.

But, fast memory is expensive and hard to find. Right now, the imaginary boundary seems to be 100 ns. Anything slower isn't a problem to find or pay for, but just try to shuffle under the 100-ns limbo stick, and you'll be brushing dirt from your behind.

It wasn't long ago that we were happy with 650-ns EPROMs that needed three voltages. I suppose I should be grateful just to be see 55-ns RAMs and EPROMs in the data books, never mind actually getting my hands on some. OK.. .it's not quite that bad. You can find them, but they're not cheap.

In my search for some fast static memory, I stumbled on an intriguing family of NVSRAMs. This particular line of NVSRAM was created by Simtek of Colorado Springs. Although all their SRAMs are fast-speeds of up to [or should I say down to) 25 ns-that's only the warm-up act.

Headlining the show is an EEPROM cell paired with each of the SRAM cells (see Figure 1). This shadowing offers the speed of SRAM and the nonvolatile storage of EEPROMs to complement one another quite nicely.

I find the most useful parts to be the 8- and 32-KB devices using the industry standard pinout. To the user, the device is SRAM and can be read and written using normal SRAM timing. However, because the device also has a nonvolatile mode, there are special timing parameters which must be adhered to.

During system powerup, a recall of stored data is automatically initiated. This recall loads the SRAM with the last data stored and takes approximately 20 µs after V_{cc} has reached 4.0 v.

Your system should not attempt a write to the device before recall has been completed. During recall, access to the SRAM is denied. You may choose to pause for t_{RECALL} (time recall) and do some other initialization or read a known value previously stored until it appears as the not-busy status indicator.

In addition to this auto power-up recall, you may retrieve the last data stored at any time through a software command. However, the movement of data to and from the nonvolatile EEPROM happens in block mode only. It is an all or nothing transfer.

The command necessary to initiate recall through software is actually six sequential reads to particular addresses. For the 8 KB x 8 NVSRAM these are 0000, 1555, OAAA, 1FFF, 10F0, and OF0F. If no other access to the SRAM interrupts this sequence, a two-step recall cycle is performed:

- the SRAM is cleared
- the EEPROM data is transferred to the SRAM.

The nonvolatile data can be recalled an unlimited number of times.

As with the recall sequence, storing data to the nonvolatile EEPROM is accomplished through software. For the 8K x 8 NVSRAM the six addresses to initiate a store are 0000, 1555, OAAA, 1FFF,10F0, and OFOE.

The sequences differ depending on the size of the device you're using. Each of the addresses is masked within a boundary of the chip. You must actually use the physical address of the device, wherever it happens to be physically located in the address space of your system. This way the device is selected (* *CS)* by the upper bits, while the lower address bits stimulate the function.

There are situations when you want data to be saved if the power fails. Some versions of Simtek's NVSRAM contain an autostore. This function is similar to the software which initiates store, except that it automatically saves if the NVSRAM has been changed and V_{cc} drops below 4.0 V.

The only hitch is your power supply must not drop below 3.6 V in less than 10 ms or the storage cycle is terminated. If your system power disappears too quickly, you can hang a 100-µF capacitor across the NVSRAM and feed it V_{cc} through a diode.

With this alteration, system power can go away completely while the capacitor holds the NVSRAM alive long enough for the storage cycle to complete.

I like the Simtek devices which use the industry-standard 28-pin 0.600" DIP packages as they can be easily used in a many products they were not specifically designed for.

If you are beginning a design, don't overlook some of Simtek's other NV-SRAM packaging. Nonstandard packaging provides extra signals which add unique features. Storage and reset or recall can be initiated by control input bits. Feedback on the status of the store or recall cycle is also available through an output bit.

USER CONFIGURATION STORAGE

Many applications require that constants used in the program be altered from time to time. Rather than having to enter these manually each time the program is run, it is best if they can be saved on a semipermanent basis. This way a program uses them as necessary unless the user calls up a configuration routine. This storage could be accomplished by replacing a standard SRAM with an NVSRAM.

Take for instance a simple minimum 8031 circuit using two 32-KB memory sockets (like the 803 1/8052 board I presented back in *INK 8).* The first memory socket can hold an 8- or 32-KB RAM (for this application, NV-SRAM) addressed at OOOOH. The second memory socket can be 8- or 32-KB RAM or EPROM addressed at OOOOH or 8000H.

As an 803 1 system, the second socket contains the application in EPROM addressed at OOOOH. As an 8052 system (using an 80C52 processor with masked BASIC), the second socket holds your application in EPROM addressed to autostart at 8000H.

If you are writing the application in a language like BASIC which allocates memory space as it encounters each variable, predefine all variables so they are always placed in the same locations in RAM. Allow only configuration routines to change their values.

Before leaving the configuration, initiate a store. Now these values are protected even if the power goes down. The next time the system is restarted, the variables are initialized when the NVSRAM does a power-on recall.

Listing 1 offers a simple example written in BASIC. The variables are defined without necessarily changing their values. The menu allows only two keys, 1 and 0. The 0 key, to set up the sensor's characteristics, may be a secret key (i.e., a key or sequence known only to the maintenance personnel). This way no one may change the sensor's characteristics without authorization.

This type of configuration works well with compiled and straight assembler code. Just be sure to do the busy test on the NVSRAM before writing anything to it so you can be sure it has completed its power-up recall.

USER PROGRAM STORAGE (BASIC)

Using the same hardware as the previous application, you might wish to have the flexibility of making changes to the application program while still retaining the security of ROM-based storage.

This time let's use the NVSRAM for program storage by moving it into the second memory socket in place of the EPROM, addressed at 8000H.

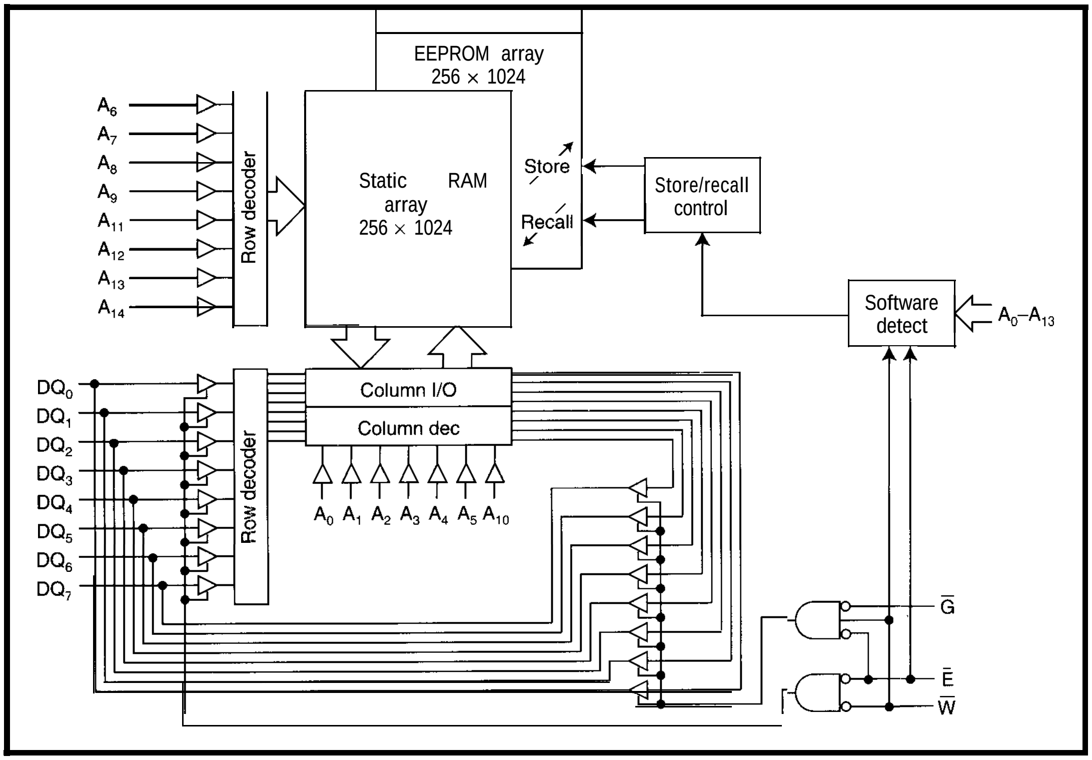


Figure I--The *Simtek NV-SRAM achieves its* magic *by* shadowing a *standard SRAM array with a second array of EEPROM cells.*

Place a standard SRAM in the first memory socket at address 0000H. The application can be typed or downloaded into RAM. It is in the volatile SRAM at this point and must be moved into the appropriate area so it automatically runs whenever the power is applied. Append the small BASIC program found in Listing 2 to the end of your BASIC application.

From the command line, execute the appended program by using the command GOT0 60000. A 55H is written to location 8010H, which indicates to the processor that a BASIC program follows. Your program is then moved up to address 801 1H, and to give some feedback, a period is printed for every line moved.

A 34H is written to address 8000H on powerup, which tells the processor to run the BASIC program. The present baud-rate-reload value is stored at 8001H and 8002H. This word sets up the console port when the program is autostarted.

Finally, 1FH is written to 8003H and FFH is written to location 8004H. This word is used to preset the MTO P value (normally the top of SRAM).

Now to make sure all this information is around after the power is lost, a store command is initiated, as before. Notice this time the dummy-read addresses are located in the 8000H–9FFFH area. Longer programs may require using a 32-KB NVSRAM.

If the program should need to be altered or edited, the process can be repeated without erasing an EPROM. Alternatively, the executing program can be halted and moved back into lower SRAM with two simple commands: control-C and X FE R. The first command halts the program and the second moves it into lower SRAM.

Now you can edit your application and, since the appended program is still attached to your application, use the GOTO 60000 command to resave it, quickly and simply.

USER PROGRAM STORAGE (ASSEMBLER)

I know many of you wouldn't think of writing an application in BASIC. Well, you C and assembly guys (and gals) can have your cake and eat it too.

Listing 1--This *80C52 BASIC listing demonstrates how to use the NVRAM for storing variables in the EEPROM array of fhe device.*

```
10  G=G: M = M: T = T: V = V : REM Define variables
20  PRINT"Hit"
40  PRINT" 1 to display the temperature"
50  PRINT" 0 to configure (mV/degree)"
60  G = GET: IF (G <> 30H .OR. G <> 31H) THEN GOT0 60
70  IF (G = 31H) THEN GOSUB 1000: GOSUB 2000: GOT0 20
80  INPUT"How many millivolts per degree?", M
90  XBY(1FFFH) = 55H: REM Location and value to check for busy
100 D = XBY(0000H): REM Sequence to initiate store
110 D = XBY(1555H)
120 D = XBY(0AAAH)
130 D = XBY(1FFFH)
140 D = XBY(10F0H)
150 D = XBY(0F0FH)
160 IF (XBY(1FFFH) <> 55H) THEN GOT0 160: REM Busy check
170 GOT0 20

1000 REM Read ADC
1999 RETURN

2000 T = V/(M*1000) : REM Temperature = volts/millivolts/degree
2010 PRINT"The temperature is ",T," degrees"
2020 RETURN
```

The same hardware can be used with a slight twist. The problem here is finding a way to load the application code and have it execute on its own. Since this hardware is flexible and can shift the memory devices in series for 64 KB of combined space (ROM/RAM) or in parallel for two 32-KB spaces-one code (ROM) and one data (RAM)-we can use this to our advantage.

Start out with a parallel setup. The NVSRAM in data space and a monitor ROM (e.g., MON31, which does not use external RAM) in the code space. When the system is powered up, you can communicate with the monitor using an RS-232 connection.

One of the monitor's commands is HEXL. This USERFILE.HEX loader places your application into RAM for you. When defining your data area, you

Listing *2-Append this BASIC listing to your 80C52 BASIC application to enable storage of the application to NVSRAM's EEPROM cells.*

```
59999 END
60000 XBY 8010H) = 55H
60010 FOR X = 200H TO (200H+LEN)
60020 XBY X+7E11H) = XBY(X)
60030 PRINT "" ,
60040 NEXT X
60050 XBY 8000H) = 34: REM ASCII '4' for PROG4
60060 XBY 8001H) = INT(RCAP2/256)
60070 XBY 8002H) = RCAP2 (INT(RCAP2/256) * 256)
60080 XBY 8003H) = 1FH: REM For 8-KB SRAM (use 7FH for 32-KB
SRAM)
60090 XBY 8004H) = OFFH
60100 D = XBY(8000H): REM Sequence to initiate store
60110 D = XBY(9555H)
60120 D = XBY(8AAAH)
60130 D = XBY(9FFFH)
60140 D = XBY(90F0H)
60150 D = XBY(8F0FH)
60160 IF (XBY(8000H) <> 32H) THEN GOT0 60160 : REM Busy check
60170 PRINT"Autostart program saved"
60180 END
```

must plan to share the same space with the code. For example, code (which starts with the reset vector at 0000H) uses address space through 5438H, data space begins above 5438H and can extend through 7FFFH (so you need to use a 32-KB NVSRAM).

Once your application is loaded using the monitor, initiate a store in the NVSRAM by using the display command to read the six addresses necessary. To determine when the store command has finished its internal programming (-10 ms), continually reread a location until the known data is returned correctly. If the board is reset now, the monitor executes again, and this is not the desired result.

Instead, reconfigure the hardware with a couple of jumpers to put it into series mode. If you remember, series mode places the memory devices so that each is in combined memory space. Your application now executes on powerup and actually runs from NVSRAM.

The data space is above your application, but within the same NVSRAM device. The monitor ROM is safely tucked away at 8000H where it never again executes. Never, that is, unless your application runs wild or you reconfigure the jumpers for monitor execution and another H EXL.

AUTOSTORE

It is not a requirement for every application that the NVSRAM be stored whenever power is lost. In fact, in most applications, it is unnecessary and undesirable (a waste of limited EEPROM writes).

However, there may be applications in which unstored data could be unintentionally discarded if the power was lost. Certain NVSRAM models are available with an autostore feature, which initiates a store if the power drops below 4.0 V and the contents of the NVSRAM have changed since the last recall.

EXPRESS LANE

So what does all this have to do with faster processors?

Replacing the old 803 1 with the Dallas 80C320 is supposed to crank through your code at three times the speed (for most instructions).

But, why stop there? You can push to 20 MHz or even 30 MHz for a 6-9 times improvement, and all it costs you is faster memory. Simtek's NVSRAM is already rated at under 50 ns (55 ns is the required SRAM speed for a no-wait-state 33-MHz system).

So, you get nonvolatility and speed without paying the additional costs!

EEPROM SPECS

This is a hot subject today because the standard specification for EPROMs guarantees a IO-year minimum data-retention life. If this magic lo-year boundary was going to be a problem, it would have happened by now. All the digital equipment manufactured over 10 years ago would be useless today.

For Simtek's EEPROM cells, data retention is a minimum of 10 years from the last storage cycle. The EEPROM cell's endurance is over 100,000 write cycles (read cycles are infinite). While we call 100k cycles limited, it still offers a lot. For instance, if your application stored data once every hour of every day, after 10 years of use, it still would not have reached the end of its life!

I think you'll agree that this kind of limitation in reality is not of concern to a well-designed application. ❑

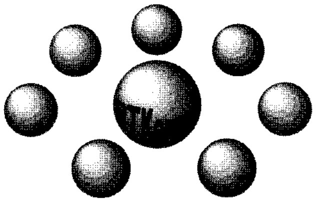
Jeff Bachiochi (pronounced "BAH-key-AH-key") is an electrical engineer on Circuit Cellar INK's engineering staff. His background includes product design and manufacturing. He may be reached at jeff.bachiochi@circellar.com.

CONTACT

Simtek Corp.
1465 Kelly Johnson Blvd.
Colorado Springs, CO 80920
(719) 531-9444
Fax: (719) 531-9481

I R S

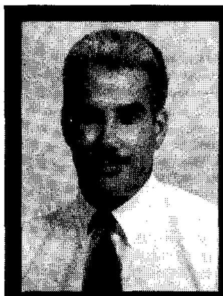
422 Very Useful
423 Moderately Useful
424 Not Useful



The Little '486 That Could

SILICON UPDATE

Tom Cantrell



It runs neither DOS nor Windows, but the NS486SX is still a '486. Although it lacks real and virtual modes and hence backward compatibility, in Tom's opinion, it still has a lot to offer. Listen in to see if it's the chip for you.

Given the hoopla over P6, Ml, K5, and all the other desktop 'x86s, you might have overlooked the recent announcement from National of their embedded '486, the NS486SX.

Keep in mind that Silicon Valley is a rather computing-centric locale. If it isn't 64+ bits and doesn't sit on (or at least near) a desktop, folks lose interest fast. Around here, a "controller" is the gray suit that minds the till lest your startup crash and burn.

Thus, when the gurus gather it's not surprising the NS486 isn't the main topic of conversation. "Why, the poor thing doesn't even run DOS or Windows, and it's only got a 16-bit bus," they sniff.

You've got to understand a market before passing judgment on a chip which targets it. From the embedded control perspective, I'd say the NS486 deserves a very close look.

DESIGN CHALLENGE(D)

The last time I wrote about embedded PCs (*INK* 37), I came up with a scheme to classify what's otherwise a rather confusing conglomeration of products that all fall under the heading of embedded PC.

Photo 1 summarizes my four-tier classification of embedded PCs based on their hardware form factor, bus compatibility, software compatibility, and so on.

OK, pencils down. Before continuing, let's take a little test to see if you've got a clue.

Your company has been commissioned to design a widget with color graphics, disk storage, and modem communications. Though production volume won't be high, your boss says the time to market was yesterday. Which design strategy should you follow?

a) After evaluating all the latest 64-bit SuperDuper megaCPUs, design and debug a 300-chip board, buying many expensive and specialized development tools. Write and document a full-blown operating system with intricate I/O drivers and file system. Then, look for a reliable source of the whizzy chips (some of which are actually in production) you designed in.

b) The design phase takes part of your lunch hour. You toddle over to the local PC emporium and pick up a clone with everything you need.

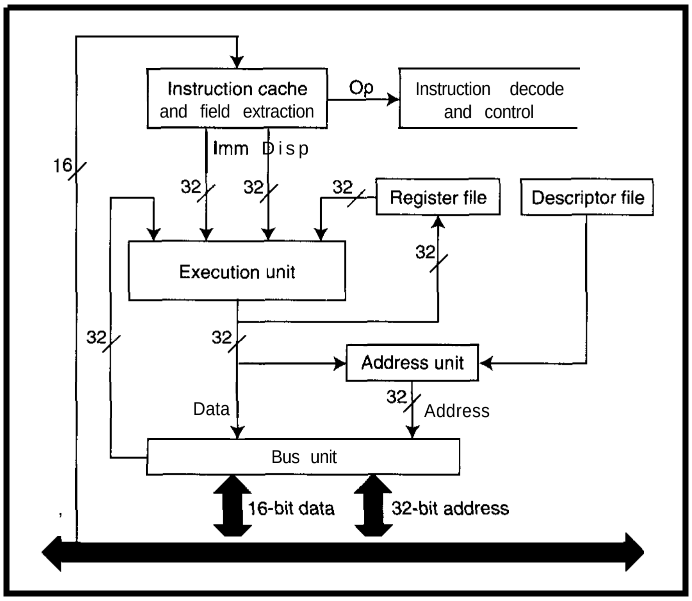


Figure I--The *NS486* core deletes *the desktop CPU's real and virtual 86* modes *and slashes cache size to* meet embedded price *and power-consumption requirements.*

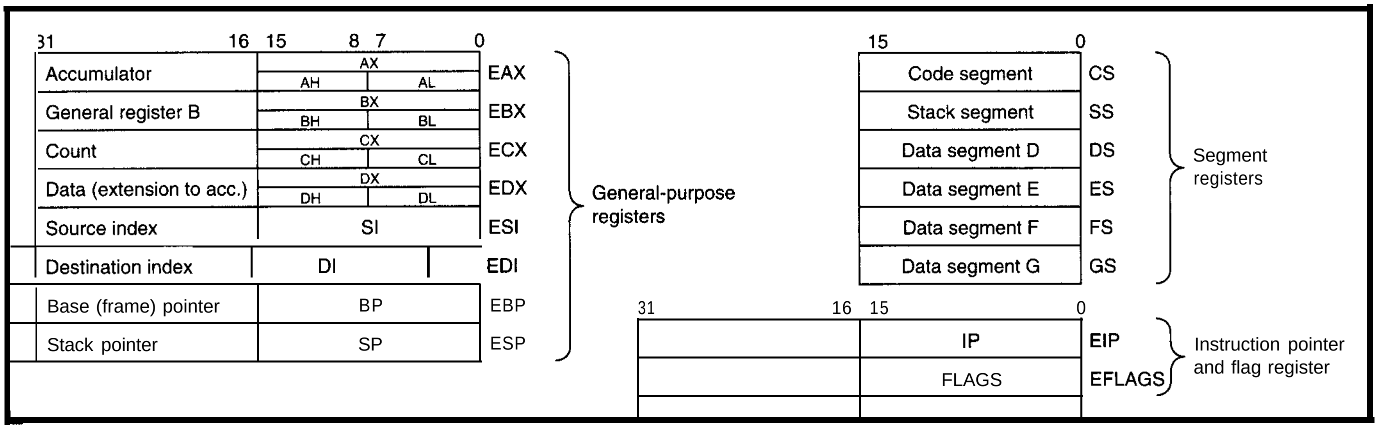


Figure 2—*The* '486 programmers **model should look** *familiar since, other than extension of the main registers to* 32 **bits, it** *hasn't* **changed in** *almost* 20 **years.**

If you chose a) your prospects for an engineering career appear grim. Your only chance is to read this magazine in the hope you finally get it.

If you passed, read on to see just what the NS486SXF brings to the party and whether it fits in with your embedded PC plans.

UNREAL '486

At the core of the NS486SXF is a CPU (see Figure 1) that's more or less the same as an Intel '486SX (i.e., no floating point, 16-bit data bus). Actually, it's probably better to say the National core is more and less a '48GSX.

The "less" part includes the previously alluded to no DOS or Windows compatibility. This is specifically a byproduct of the fact the NS486 only runs in protected mode and lacks the Intel real and virtual-86 modes that keep all that old PC software happy. The NS486 also deletes the TLB (Translation Lookaside Buffer) and virtual memory in favor of simple physical addressing.

For those of you who have been on another planet, Figure 2 summarizes the '486 programming model, which is little changed from the original 8086, except for an extension of the main registers to 32 bits. The protected mode (introduced in the '286) adds various (global, local, interrupt) de-

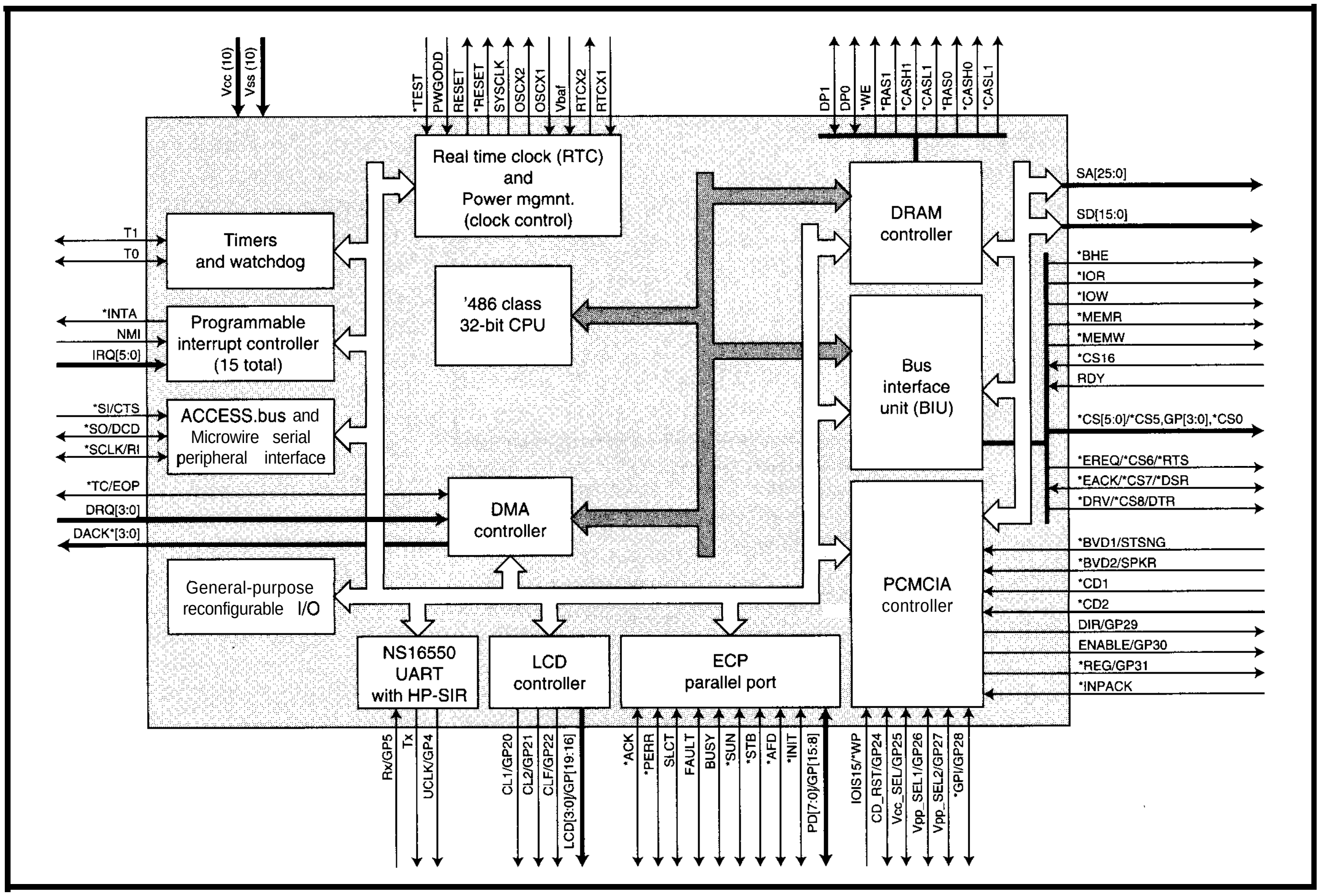


Figure *3-The* '*SFX CPU may* **walk** *softly,* **but** *it carries* a big *stick when it comes* **to** *I/O.*

scriptor registers that enforce memory access rights for various tasks.

Other than those few instructions rendered moot by underlying feature deletion (e.g., TLB), the complete '486 instruction set is there in all its CISC glory-everything from AAA to XLAT. I won't go into the gory details since there are plenty of other (not to mention better) places for you to find out how a '486 works. After all, no shortage of know-how is part of the embedded PC concept, so I don't feel guilty punting.

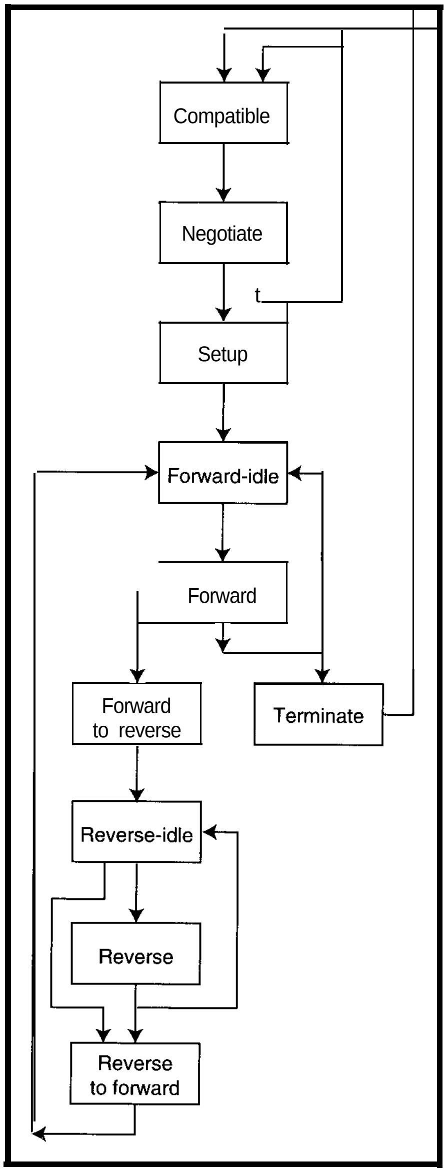
The NS486 lightens up on cache as well, replacing the desktop's 8-KB unified instruction-and-data cache with a l-KB instruction-only cache. In my opinion, while the frenetic race to ever larger caches may make sense for performance-at-any-price high-end CPUs, it's misguided in the adequate-performance-at-lowest-price embedded world. Remember, less cache in your chip means more cash in your pocket!

Proponents of giant cache claim they can cut the miss rate in half. That sounds marvelous, until you realize they mean from 10% to 5% which ultimately translates to a trivial system speedup.

Cache dogma also depends on whether you view the glass as half-full or half-empty. Thanks to the prevalence of tight loops, even a tiny cache may achieve surprising hit rates. One example in Hennessy and Patterson ("Computer Architecture-A Quantitative Approach", Figure 8.12) shows +80% hit rates for l-KB caches. Though a CPU with small cache may run slightly slower than one with a much bigger cache, it runs a whole lot faster than a CPU with no cache.

Underneath the hood, the NS486 downsizes further by cutting the pipeline from five to three stages. This sounds rather drastic, but mainly serves to limit the clock rate [to 25 MHz initially). Within the clock limits, NS486 performance is quite similar to a '486SX running at the same speed. Given the embedded orientation, the clock rate compromise makes sense to minimize die size, power, heat, RFI, memory speed, and so on.

Thanks to throwing all the desktop baggage overboard, the CPU core depicted in Photo I is lean-and-mean indeed, leaving room for a lot of neat system logic and I/O functions that make the NS486 eminently embeddable.



4—*Jhisn't* your *father's* printer *port. As* this figure shows, the *IEEE* 1284 *ECP* standard *adds major functionality and complexity to fhe ungracefully aging Centronics port.*

PICK A PECK OF PERIPHERALS

As shown in Figure 3, the NS486-SXF packs the CPU core and a veritable data book's worth of I/O into its 160-pin PQFP (Plastic Quad Flat Pack). Rather than reinventing the wheel, many of the peripherals including the UART, interrupt controllers, timers, real-time clock, and PCMCIA controller are lifted from the desktop. Exploiting the PC know-how advantage means less stuff for designers (and me] to wade through. However, though some of the peripherals are old hat, their integration into the CPU is leading edge.

For example, the RTC is basically the familiar desktop MC 1468 18 with a few minor changes such as reducing the general-purpose RAM from 113 to 50 bytes. Mundane as the RTC itself is, the decision to put it into the CPU has benefits far beyond saving a chip.

Besides the convenience of keeping track of days, months, leap years, daylight saving time, and so on, the cost of a 32-kHz watch crystal buys unique advantages when it comes to power reduction and reliability.

For example, the deepest sleep (i.e., lowest power) mode shuts off all the other clocks in the chip including the CPU's and peripheral's Normally, some (often too much) of the CPU has to be kept alive lest sleep turn into coma. With the RTC built-in, sleeping power consumption is minuscule, yet it's no problem to arrange or field a wake-up call. The RTC is also the heartbeat for the watchdog timer, an arguably more reliable separation of duties than typical.

On the other hand, sometimes the leading edge is a little too bleeding edge for my liking. It's no secret I've been rather skeptical about PCMCIA (in an earlier article, I referred to it as "The bus that never stops-changing"]. Juxtapose slippery slope standards with the PC plug-and-pray crisis, and you've got trouble. Fortunately, the feature only consumes eight I/O lines, and they're blessedly reconfigurable as general-purpose I/O.

In other cases, like a baby boomer in grunge, a traditional function is upgraded with Generation-X features. For example, the venerable NS16550 UART [with 16-byte FIFO) adds support for infrared communications. You may remember from my "IRDA-The IR Babel Buster" (*INK* 48) how IR schemes only turn on the LED for a portion of the bit-time to reduce power consumption. The 'SFX IR mode supports both the HP-SIR (3 or 16 bit time) and IRDA (1.6 us) variants.

The good old Centronics port also gets a makeover. While compatibility mode works as always (i.e., as a host or slave printer port), the recent IEEE 1284 spec defines new ECP (Extended Capabilities Port) features that go far beyond simply formalizing historic bidirectional hacks. In fact, as a glance at Figure 4 indicates, ECP has ballooned into a SCSI-like high-speed (2 MBps) data link. Check it out, it's even got RLE (Run-Length Encoding) compression built-in.

No doubt any controller worth its salt needs a clocked serial interface like Philips' I^{2}C, ABIG's ACCESS.bus, Motorola's SPI, or National's Microwire. You might expect Microwire to automatically get the corporate nod, but in fact the 'SFX supports them all.

The timer block, based on the popular 8254, features three 16-bit timers (TO-T2). TO and Tl feature the original's plethora of operating modes and each has a pin configurable as either a gate input or strobe output. The timebase for Tl and T2 is selectable as 1/8, 1/16, 1/32, or 1/64 of the system clock.

T2 is specialized to serve as watchdog and is fed by a separate 1-kHz clock from the RTC. It has some neat features including the ability (attempt) to interrupt the CPU on the first timeout and reset it on the second. Also, the functions that stop and retrigger the watchdog are protected by magic codes (i.e., specific sequence of address and data) and should be impervious to minor fender benders.

The LCD interface is one of those features that can prove compelling if your application exploits it. Seven output lines comprise the interface, which targets midrange (i.e., 320 x 240 mono or four-level gray) panels. The interface is CRT-like, consisting of frame, line, and shift clocks along with a 4-bit data bus.

The LCD bitmap resides in system memory as opposed to a separate frame buffer. Periodically (at the line clock rate), the controller fetches a line's worth of data from system RAM into an internal FIFO. From there, it is shifted four bits at a time to the display. Like a CRT, the complete screen is refreshed at a +60-Hz rate for a pleasing display. Thanks to the limited

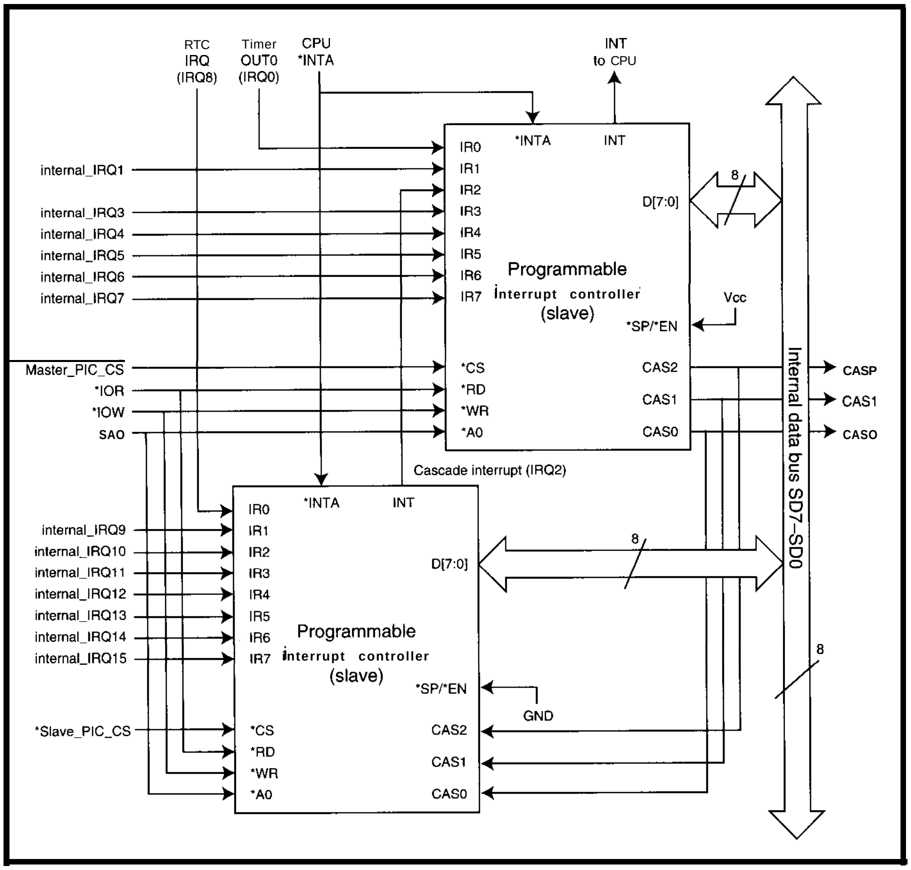


Figure 5—*With so much I/O competing for attention, the 'SFX needs dual 8259A interrupt controllers, six external IRQ pins (plus NMI), and flexible "steering" logic* to route *requests.*

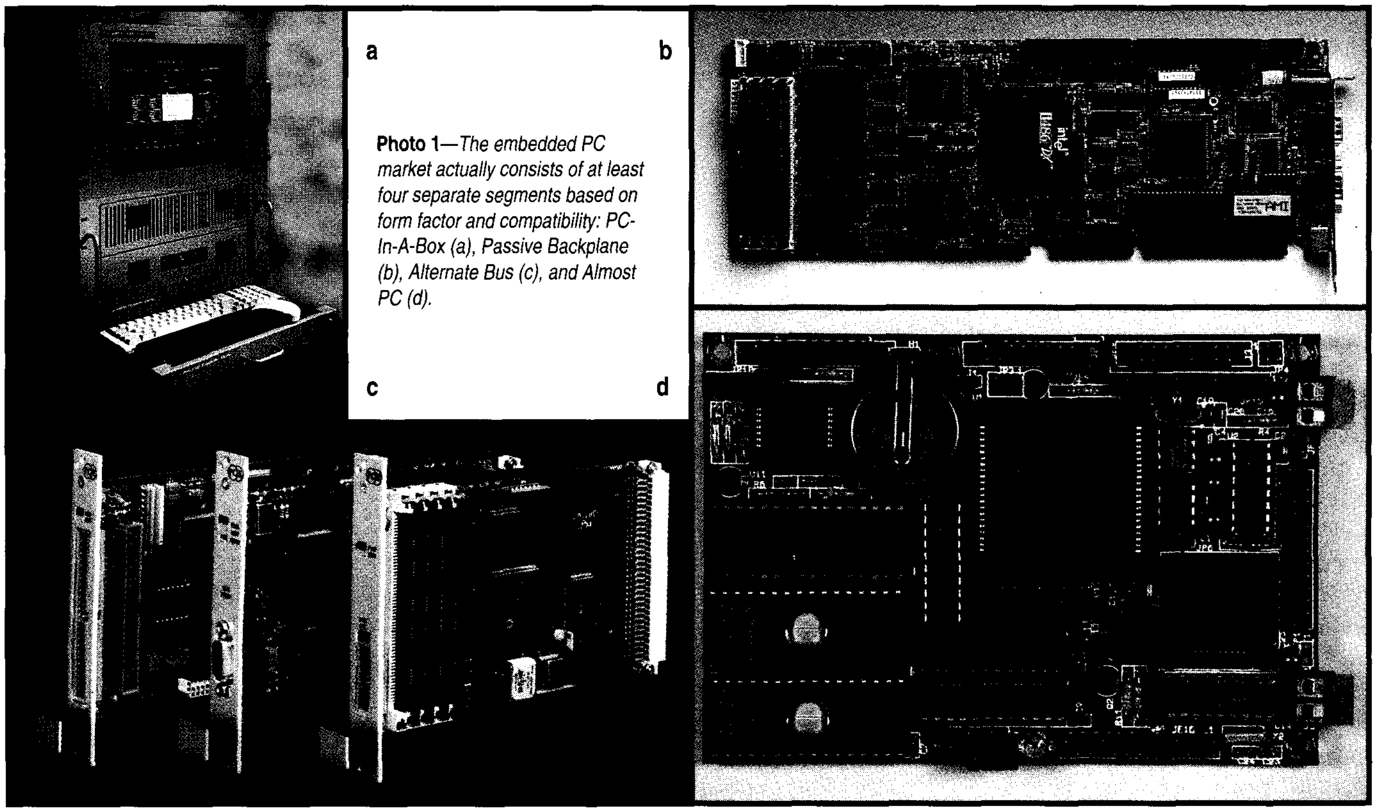


Photo 1—*The embedded PC market actually consists of at least four separate segments based on form factor and compatibility: PC-In-A-Box (a), Passive Backplane (b), Alternate Bus (c), and Almost PC (d).*

resolution and relatively speedy CPU, refresh overhead is typically less than 10%.

Unlike a CRT, gray scale is tricky for an LCD since a pixel is either on or off. The LCD controller mimics the varying intensity of a CRT gun by switching pixels on and off at high speed in a PWM-like manner. However, if adjacent pixels go on and off at the same time, flicker may become noticeable. The 'SFX LCD controller uses a clever scheme to avoid display artifacts by fetching dithered (modulation phase shifted] gray-scale look-up tables from the frame buffer along with the bitmap data.

KINDERGARTEN COP

Just like the movie, all those pesky little I/O critters might quickly bring even a mucho-macho CPU to its knees. To keep 'em in line, the 'SFX devotes plenty of resources to managing I/O.

Keeping track of and prioritizing a bunch of internal and external interrupt requests is handled by two 8259A interrupt controllers. As shown in Figure 5, the controllers operate (as in a PC) in cascade mode, a concept that can be extended further with an external 8259 if necessary. However, the seven 'SFX interrupt pins (IRQO-5, NMI) should be more than adequate in most cases.

Of the sixteen interrupts, two are dedicated to the RTC and one is used for cascading. The 'SFX adopts a steering logic approach that offers rather flexible assignment of internal and external interrupt sources to the 13 available interrupts.

Further semblance of order is maintained with a powerful six-channel DMA controller. Like the interrupt steering logic, the 'SFX lets three internal (LCD, ECP, PCMCIA) and four external (DRQ/DACK) pin pairs be flexibly allocated to DMA channels.

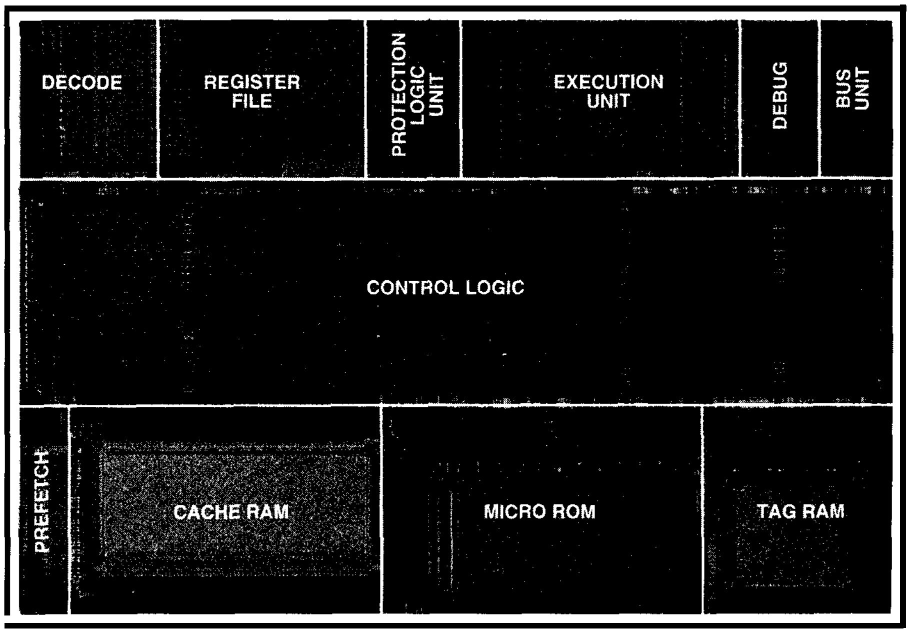


Photo **2**—*The NS486 CPU* core *photo makes c/ear the cost savings, especially* ***that*** *attributable to cache-size reduction (8 KB to 1 KB). Notice the influence of the latest DA (Design Automation) techniques that rely on optimized "cells" for regular logic (i.e.,* ***RAMs,ROMs)*** *and synthesize* ***the*** rest.

The DMAC features the usual modes (i.e., single, burst, block) and 32-bit addressing, though the byte count remains stubbornly stuck at 16 bits (i.e., max 64 KB in a single transfer).

Particularly to support the high-bandwidth on-chip interfaces, the DMAC also includes a chaining mode that semiautomates and streamlines buffer transfers (e.g., LCD refresh) with minimal CPU overhead. Best-case bandwidth is a speedy 1 byte per clock (i.e., 33 MBps@ 33 MHz}, though typical transfers are somewhat slower depending on the particulars (internal vs. external, wait states, etc.).

Getting all that data on and off the chip relies on the 'SXF's ISA-like interface in which most of the PC's bus pins are represented. Though the timing is slightly different, most newer (i.e., faster) ISA-type peripherals can be accommodated.

One major addition is a built-in DRAM controller that includes complete control-signal (RAS, CAS, etc.) generation, refresh (CAS-before-RAS) timing, and address multiplexing for a no-glue interface with up to 16 MB of DRAM (note that x4 or wider DRAMs should be used).

The controller is notably aggressive when it comes to exploiting DRAM page mode. While a simple-minded design usually limits bursts to a fixed block size, the 'SFX stays in page mode as long as possible (i.e., potentially until the next refresh cycle). Meanwhile, thanks to the realistic clock rate, it's easy to get good performance (i.e., 1
60–70-nssDRAM

Accessing the EPROM and peripherals likely to fill out the typical application is easy thanks to the built-in address decoder and a generous eight chip-select pins. Actually there are nine pins {CS0–CS8}, but CSO is assigned boot ROM duty.

The eight available logical chip selects are defined in the usual way—address space, timing, bus width (8 or 16 bits), and so on. Then, relying once again on the steering concept, the logical chip selects are assigned to physical pins.

What's interesting is that multiple chip selects can be assigned to a single pin. National points out this may be useful in dealing with certain PC-oriented chips that combine multiple functions on a single chip while retaining historically discrete address maps. Since cache support is also a chip-select area attribute, this technique could also establish both cached and uncached paths to a single device.

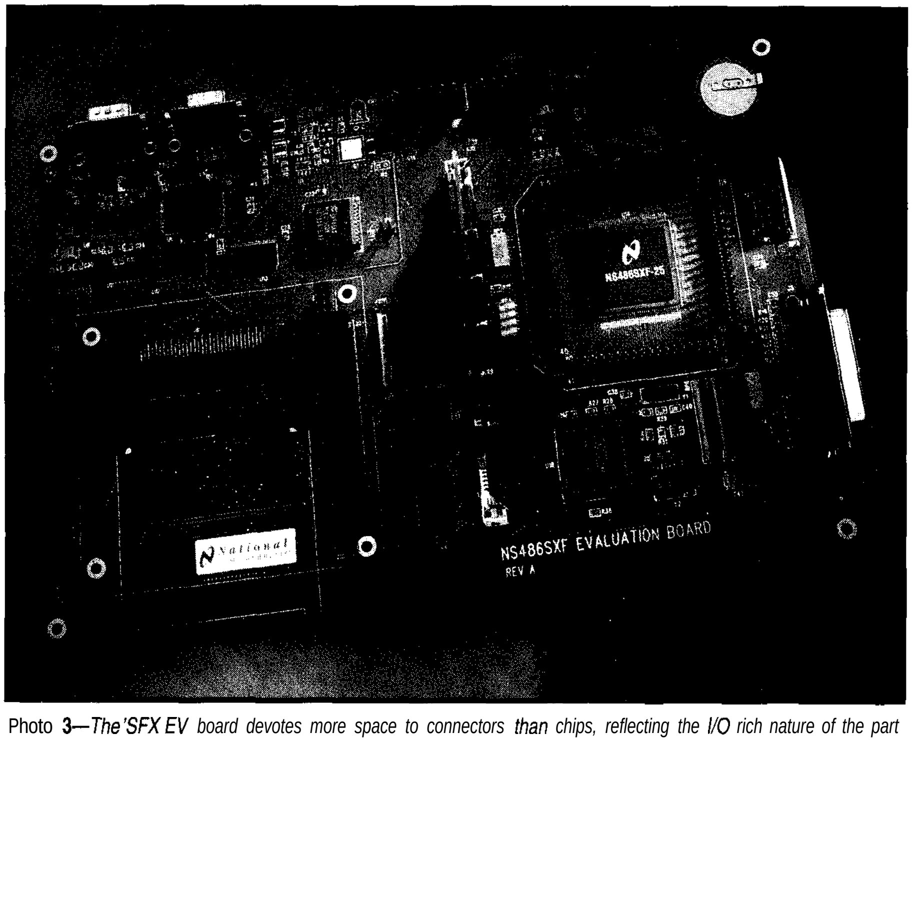


Photo 3—The 'SFX EV board devotes more space to connectors than chips, reflecting the I/O rich nature of the part

ALMOST PERFECT

Actually, it's just as well the 'SXF doesn't run DOS or Windows, a trait that clearly positions it in the Almost PC segment. In fact, prospects might be worse if it did run the PC software since designers of desktop-compatible embedded PCs usually stick with desktop CPUs.

Emphasis on PC-based development rather than compatibility lets National focus their tool strategy in a couple of categories. Simplest and lowest cost are remote debuggers and standalone I/O libraries (i.e., no OS) that allow designers to exploit the popular desktop C compilers [Microsoft, Borland, etc.).

More complicated applications rely on industrial-strength RTOS packages that integrate everything: OS, C compiler, and kernel-aware debugger. National's list of third-party supporters includes most of the big names in the embedded development tools business.

To get started, consider National's evaluation board (see Photo 2) that combines the 'SFX with a flash chip (128 KB-1 MB), DRAM SIMM **(1-16** MB), and a bunch of connectors including UART, PCMCIA, ECP, and PC/ 104. A monitor and loader are provided that use an external UART so the internal one doesn't have to juggle both debug and application chores. The $486 price seems a little rich, but I suppose the marketeers couldn't resist.

At only $25 in volume, the 'SFX performance and features are nearly ideal with extra brownie points for the RTC, IRDA support, universal (multi-protocol) clock serial port, ECP, and easy DRAM interface.

The LCD controller, PCMCIA interface, and ISA-like bus seem rather specialized, but if you need them, they're there. However, if these PC-type features are important to you, perhaps your application might best be served with a desktop-compatible CPU.

In response, National says they'll be following the 'SFX with an 'SXL that bags the LCD and PCMCIA ports. With reduced pin count (132 PQFP) and price ($15 in volume), the 'SXL seems almost perfect for an "Almost PC." ❑

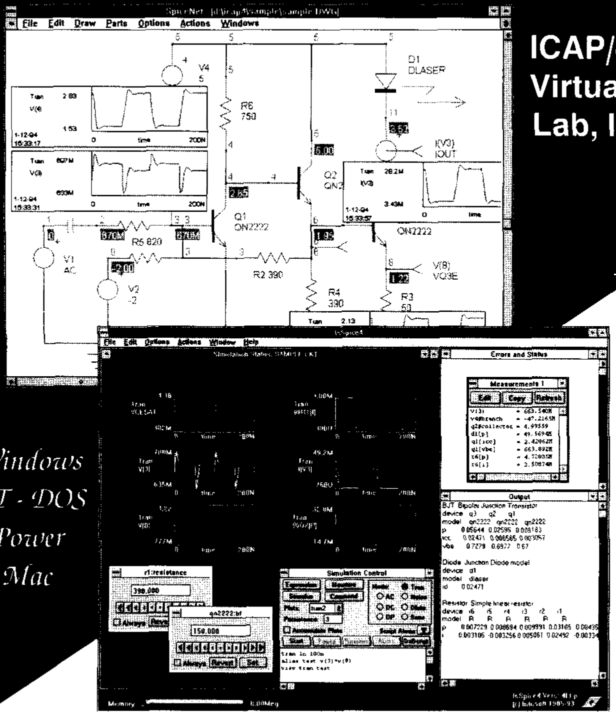
Tom Cantrell has been working on chip, board, and systems design and marketing in Silicon Valley for more than ten years. He may be reached at (510) 657-0264, by fax at (510) 657-5441, or at tom.cantrell@circellar.com.

CONTACT

National Semiconductor Corp.
2900 Semiconductor Dr.
P.O. Box 58090
Santa Clara, CA 95052-8090
(800) 272-9959
Fax: (800) 432-9672
BBS: (800) 672-6427 (8N1)

IRS

425 Very Useful
426 Moderately Useful
427 Not Useful



CONNECTIME conducted by Ken Davidson

The Circuit Cellar BBS
300/1200/2400/9600/14.4k bps
24 hours/7 days a week
(860) 871-I 988-Four incoming lines
Internet E-mail: **sysop@circellar.com**

I guess this is the month for noisy electronics. In our first thread, we look at reducing the EMI generated by a servo motor. The solution turns out to be quite simple.

Next comes reducing radiated noise generated by a triac. The solution is almost as easy, but still may need refining.

From noise we go on to hydraulics and the potential problems in putting together a simulation ride. There are plenty of factors *to* fake into account.

Finally, *just* where do those mysterious *mean-time-between-*failure numbers come from? We take a quick look at *the* math and *suggest some references for further reading.*

Brushless servo motor EMI reduction

Msg#: *4238*
From: Mike Mcquade To: All Users

Does anyone have any experience with 3-phase brushless motor EM1 noise reduction?

My system is a 160-VDC bus type, 10-kHz switcher, and most of the noise is between 100 kHz and 5 MHz. I would like to reduce the amplitude of this noise. Is it better to filter at the motor or at the servo amplifier?

Msg#: *4368*
From: James Meyer To: Mike Mcquade

The noise is probably the result of currents flowing through the wires connecting the controller to the motor. In that case, the "best" point for any filtering is as close to the source as possible. That means the switcher.

Most switchers require a minimum value of equivalent capacitance across their outputs in order for them to function properly. Whatever filter you come up with should ensure that this requirement is met.

After that, you would be well advised to make your filter present a high impedance to both the switcher and the motor at high frequencies. This means the filter should probably have inductors connected at both its input and output.

If this makes any sense to you, and if you'd like me to be a bit more specific, let me know.

Msg#: *4553*
From: Pellervo Kaskinen To: Mike Mcquade

As Jim Meyer already stated, the best place to filter is near the source of the noise. In almost all the cases, the source is the electronics, not the motor. The only exception I can think of is the "smart motors," where the controller is embedded in the motor housing.

Then to the practical components. One of the most immediate candidates is the series of commercial line filters, such as Corcom or Curtis. But there is a very important principle to observe: Do not use a capacitor input architecture in this particular application! In fact, even the inductance input versions may not have high enough inductance to work properly with the pulse-width-modulated drive.

Let's see what we are talking about, in a little more detail.

Most motors have enough inductance to allow the driver to chop the supply at about IO-30 kHz without an undue inrush of current at each pulse edge. But if you add a capacitor, then each starting pulse is first met by a virtual short circuit. Although the current spike is short, from a 160-V bus it can reach several hundreds of amperes.

Some PWM motor drives have fast-acting current limit that cuts the pulse short in a case of overload (short circuit). In such a case, the filter would not get destroyed, but neither does the motor get its power.

I ought to know, I've been there. It was not a brushless motor, but otherwise the same as your system: 160-V bus and PWM with a brush-type DC motor (about 1 kW). We also had noise and tried a Corcom filter. Lost all the torque from the motor and the filter got too hot to touch.

The solution is either a combined common mode and differential choke or just individual chokes of about 100–500 µH with the appropriate current rating in series with the output of the PWM driver, then the Corcom filter and then out of the cabinet to the motor.

One last detail to be wary of: The inductance/capacitance combinations may get into resonance. If this happens, you will see extraordinary voltage levels and possible destruction of the capacitors or the chokes or both. If the filter capacitor is 10 nF and the choke is 100 µH, then the resonance frequency is about 160 kHz, well above your 10-kHz switching frequency. But if you increase the choke and the capacitor values, you might get into the resonance danger zone.

Be aware that you might resonate with the 3rd or 5th harmonic of the 10 kHz with pretty bad results as well. The 7th and higher harmonics pose somewhat less danger in this case.

Msg#: 4906
From: Mike Mcquade To: Pellervo Kaskinen

You are right about the filtering. I was able to make the system work by putting a good Corcom (medical grade) AC line filter on the system. The noise was being radiated off the AC line and getting into our system. Thanks.

Triac noise

Msg#:25114
From: Jason **Wolfson** To: All Users

I am trying to get my PIC-controlled fan speed and timer unit working and have several noise problems. I am using a TECCOR 6-A logic triac driven directly from the PIC. I use a snubber made up of a 10-kΩ resistor and 0.1-µF capacitor to get rid of the audio noise, but now I am interfering with my AM radio. I want to be able to drive up to a 2-A load. How do I get rid of the RF noise?

Msg#:27630
From: George Novacek To: Jason **Wolfson**

First of all, if RF is your problem, you must use a zero-crossing switch. RCA, Motorola, and probably others make such ICs, which you stick between the triac and the PIC. You may want to use an optocoupler for isolation. If you don’t want to be reinventing the wheel, you can buy solid-state relays with zero crossing already built in.

With good zero-crossing design, you will still have a couple volts of the switching noise left on the power lines (triacs need a couple of volts before the junctions can be fired).

If that is still a problem, you will have to shield the switch to get rid of emissions. You will also have to put low-pass feed-through filters on the lines (Murata, for example, makes them) to cut down on the conducted emissions.

Hydraulic control feasibility

Msg#: 6308
From: Joel Clark To: All Users

At our science center we want a four-passenger simulation ride for one of our exhibits. Commercial units cost $70,000 and up. Looking at the individual components:

hydraulic pump and reservoir	$2000
six 2” diameter cylinders	$1200
six cheap proportional solenoid valves	$1500
PC plus I/O cards	$2500
hoses and stuff	$1500
video projector	on hand
structure, seats	cost separately
labor	cost separately

Over the past three years we have gotten hundreds of hours of volunteer work donated by retired engineers, machinists, and programmers, so for the moment I’m just looking at component costs. Here is the big question: Are the estimated costs reasonable?

Msg#: 6442
From: Pellervo Kaskinen To: Joel Clark

Sounds like it might get close, but my experience (from other fields, mind you) suggest you ***always*** need a contingency amount, something like 30% or more over your best estimate. If you are making guesses, then the contingency amount should be even higher.

One issue: does your insurance cover the inevitable leaks?

Another touchy question is the cheapie proportional valves. Do they handle the pressures and the flow without overheating?

One final area on my immediate thoughts is the power unit. It is going to require electrical supply. It ***probably*** also needs a certain amount of cooling, or the fluid needs a heat exchanger, which may be air cooled with a fan or water cooled. In the second case, you need an oil separation pond and/or oil monitoring system.

That is as much as comes to my mind immediately. Hope this serves you as a starting point in your effort to refine the estimates.

Msg#: 6547
From: Joel Clark To: Pellervo Kaskinen

Thanks for your fast reply. Some good points. I need to add some for cooling. There is a nearby electrical breaker box set up for 3-phase, 15-hp motor. The floor is concrete

CONNECTIME

with vinyl tile which is old; Easy-dry could handle a certain amount of leakage. Maybe a sheet metal pan under the power unit at least might be helpful.

The part I'm probably most worried about is the valves and how degraded their performance is versus the aircraft electrohydraulic servo valves that cost several thousand each. I need to do more research on those. I appreciate all your comments. I'm trying to talk my boss into giving this a try and need to avoid overlooking any big potential problems.

Msg#: 6726
From: Ed Nisley To: Joel Clark

Mmmm.. .a high-pressure hydraulic system leak doesn't drip. It fills the entire room with a fine, highly combustible, petrochemical mist. Everything in the room emerges with a thin layer of oil in addition to a huge slick on the floor. Cleanup takes approximately forever and some things just get replaced because you can't get the oil out/off/away.

Most unpleasant, reports a friend who had the misfortune to be standing on the spot marked X when a prototype airplane hydraulic system popped a hose in a big room. It's all over in a second or so.. .no time to react.

Your situation may not be quite so dramatic, as the pressures and flow rates go up as the system performance increases. Nonetheless, do run through the numbers and see just how much of a mess you'd have if or when the pump emptied the reservoir through an accidental pinhole nozzle..

Msg#: 7194
From: Joel Clark To: Ed Nisley

Thanks, Ed and Pellervo. This is just the kind of cautionary tale that can save a novice a lot of grief. I believe the commercial entertainment simulators use hydraulic actuators. Would you say there is a safety hazard (i.e., personal injury danger)?

Hey, I could consider huge stepper motors and ballscrews, but the requirements are about 600 pounds at 20 inches/second, 9-inch travel, and six actuators. What comes to mind for this application? I need cheap stuff or I can't do it. We do have two IO-hp air compressors all installed that could be used, but I figured air would have bad compressibility problems.

Msg#: 9109
From: George Novacek To: Joel Clark

You can hardly beat hydraulics for this type of application. I would stay away from high-pressure systems (3,000 psi), but the technology is mature and if you get help from someone who knows what he's doing, you should have no problems.

There are safety issues to consider and some fluids, such as Skydrol are deadly. They eat away the soles off your shoes if you spill the stuff on the ground.

Msg#:14860
From: Pellervo Kaskinen To: Joel Clark

Here is a classical motion control problem! I have seen numerous magazine articles covering the power demand issues, which of course is the basis for all component selections.

To get a rough idea, let's see one step by step approach (I still work only in the metric domain.. .):

600 lb. = 272 kg
20 in./s = 0.5 m/s
9 in. = 0.23 m

Power requirement is the sum of the gravity effect, if vertical movement, plus any acceleration effect on the 272 kg.

First step, ignore the acceleration. Then the time would be as short as 0.45 s. Keep this in mind for reference.

Fighting gravity only, $P = vF$, or power is the velocity times force. The force is m x g, or mass times the acceleration of gravity. Then the power is 0.5 x 9.81 x 272 or about 1.35 kW.

Now, you cannot use an infinite acceleration. More likely the speed will ramp up, then continue for a while at the 0.5 m/s you mentioned, and then taper down to zero by the time you reach the end of the travel.

Any guess is as good as the next one here. We just want to get an idea of where we stand. So, I'll take a trapezoidal pattern that starts with zero speed, reaches the full speed at ⅓ of the total time, slews ⅓ of the total time, and then for the last ⅓ of total time decelerates to zero.

Solving the speed-versus-acceleration and travel-versus-speed equations produces a duration of 0.23 seconds for each segment or 0.69 seconds total. Compare this to the 0.45 seconds obtained with infinite acceleration. Reasonable?

Then, the acceleration is a little over 2 m/s^2 or some 20% extra power is needed beyond the gravity requirement. Still reasonable?

Then the equipment side. No machinery works with 100% efficiency. Depending on the controls and so on, anything in the 20–80% efficiency range sounds likely. This actually is the most open issue that you have to come back to after tentatively selecting your components. Sanity check about a good match.

All this is a momentary situation. ***During the actual move, you*** need this much power (up to 10 kW with a really poor efficiency). But if you can use a large enough accumulator and the move only comes so seldom that the pressure does not drop much during it and can be rebuilt between the movement periods, a lot smaller power unit ***may*** be feasible.

So much for the power estimates. Then to some possible choices for the power plant.

The hydraulic unit is known to produce some of the most compact driving heads, at the risks of high-pressure oil leaks and so on. The bulk of the equipment can be "in the basement" or otherwise out of the scene.

An air-operated system is more bulky and, like you say, is difficult to control in a smooth fashion. But you could implement a hybrid system. Again, out of sight, you would have two additional cylinders. The bottom of each cylinder is connected to one or the other end of your actuator cylinder at the load. To the top, you bring the air supply through simple 3-way solenoid valves. In the cylinders, there is oil. In effect, you apply air pressure to the oil surface that acts like a piston.

The oil line to the ends of the actuation cylinder is equipped with adjustable flow restrictors. These determine the maximum speed of the actuator piston movement, pretty much independent of the load. The oil being incompressible, you get a smooth control.

Of course, you also need flow restrictors on the air discharge lines after the solenoid valves, otherwise the oil escapes in spurts every time you release the pressure on either intermediate cylinder.

Like the hydraulic accumulator, this also can benefit from an air tank storage so that the average power can be lower than the peak power demand.

Then, the motor and ballscrew approach. Rather than using low- efficiency stepper motors, plain DC motors would be my choice. If you have six actuators, the total power requirement of about 1.8 kW could be met with six 300-W DC motors. The ballscrews have a good efficiency, and the motors are rated for their output power.

Naturally, you would need controls to feed the motors. SCR-type controls are mostly for unipolar or single quadrant load, but there are reversible devices that are still reasonably low in cost.

I hope this clarifies rather than muddies the issues you are battling with.

MTBF

Msg#:10104
From: Terry Jones To: All Users

Where I work, mean time between failures is being used as controlling factor in the purchase of devices. Some of the values quoted for these devices (mainly environmentally hardened PCs and controller boards) range from 17 to over ***50*** years of continuous use.

Can anyone explain how the manufacturers come up with these values? Is there some statistical method they are using to extrapolate (guess??) the length of time a device will run before failing? Can anyone recommend any books or articles that will give me a better idea of where these numbers come from?

I want to put a cheaper off-the-shelf PC option in rather than spend seven times as much for a hardened device.

Msg#:12342
From: George Novacek To: Terry Jones

The bible is military handbook MIL-HBDK-2 17 which, I believe, is currently at revision F. The "Automotive Electronics Reliability Handbook" published by SAE is excellent reading, although MIL-HBDK is next best thing to a cookbook when you actually need to calculate reliability prediction.

Very simply, you determine reliability for every single component in the product. Then, by using simple arithmetic, you determine the overall MTBF of the product. There are several different methods, all described in the handbook.

The main problem is determination of the reliability of each and every component. This was very straightforward with military (sometimes referred to as ***established reliability*** components). The base reliability was determined by statistical methods through years of testing and field data feedback. All MIL-rated components come with the reliability number. Many parts are listed in the HBDK, some on a generic basis.

For example, an IC with so many transistors has historically such and such reliability. The base number is further modified depending on the environment, operating temperature, and derating.

For example, a load resistor will dissipate 200 mW. Significantly greater reliability will be achieved with a ½-W as opposed to ¼-W resistor. Or, a capacitor rated for 50-V working voltage in a 30-V circuit is more likely to fail than a 100-V capacitor.

The process is very straightforward, but tedious. You must understand how the circuit works. The analysis should be done concurrently with design, as it uncovers many potential weaknesses.

CONNECTIME

Now, with commercial parts, the situation becomes very tricky. Military standards are obsolete and often unrealistic, assuming (wrongly) that modern commercial components are much less reliable than military ones. Quality of the design is another factor, which is hard to express numerically. Consequently, MIL standards place such a reliability penalty on commercial components, that based on the calculation results, the products should fail even before they are turned on.

On the other side of the spectrum are reliability data provided by component manufacturers. Here, commercial number games are played. The numbers are often so inflated or based on unpublished, proprietary test assumptions, as to be completely useless for meaningful analysis. We have been unsuccessful in trying to convert data from several electrolytic capacitor manufacturers to a form which could be useful in reliability prediction.

Generally, if the product with labeled MTBF *is not a* MIL product, the manufacturer is basing his claim on historical data collected either on this particular design, or collection of component data and calculation. The former is much more representative of the true picture, as it takes the design quality and operating environment into account. The latter is a numbers' game.

Let's say you have 10,000 widgets working for a full year 8 hours a day. Ten had died. That means you have 2.92 10^{-6} failures/hr.-an excellent record. Still, you must keep in mind that the MTBF does not guarantee that out of the 10,000, a widget does not bite the dust in the first hour of operation.

The working environment is extremely important. The above MTBF measured at room temperature will deteriorate drastically if the widgets work at an elevated temperature. Therefore, keep in mind, MTBF is not a guarantee. All it can give you is a warm feeling that the product is well designed.

871-
1988. Set your modem for 8 data bits,
14.4k300, 1200, 2400, 9600, or

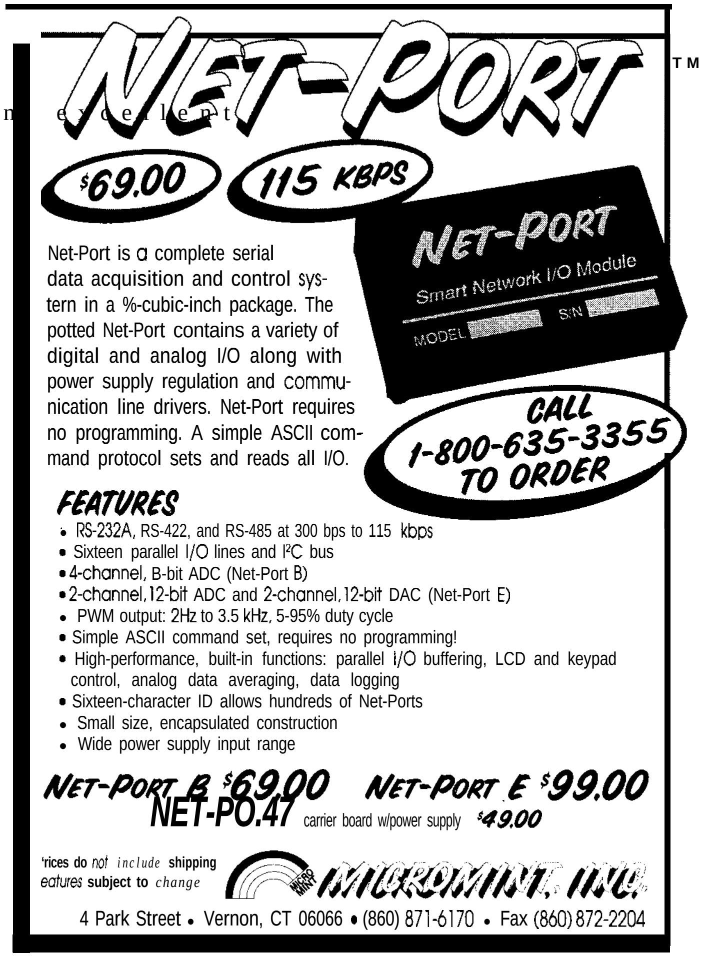
ARTICLE SOFTWARE

Software for the articles in this and past issues of *cuit Cellar INK*

ftp://ftp.circellar.com/pub/circellar/.
http://www.
circellar.com/.
info@circellar.com to find out how to request files through E-mail.

For those unable to download files, the software is also available on one 360 KB IBM PC-format disk for only $12. To order Software on Disk, send check or money order to: Circuit Cellar INK, So Disk, P.O. Box 772, Vernon, CT 06066, or use your Visa or Mastercard and call (860) 8752199. Be sure to specify the issue number of each disk you order. Please add $3 for shipping outside the U.S.

IRS



PRIORITY INTERRUPT

Livin' and Learnin'

Ordinarily, power-supply design wouldn't be a big issue with me, but it does arouse some significant memories. Most everybody remembers that before we started CC *INK, I* presented projects in *BYTE* for 11 years. Because most people think of that activity as the start of my publishing career, nobody ever asks what happened before. I wasn't selling used cars one day and then invited to be *BYTE*'s first regular columnist the next.

This CC *INK*'s focus on power supplies got me thinking about how all this started with a power supply. Like many technical people in the early '70s, I was very interested in the new stuff from a little company called Intel. At the time, I was working as an electronic engineer for Control Data. Being with a computer company gave me an immediate understanding of the power and performance of computing devices, but as you might expect, their attitude was that any computer smaller than a house and costing less than a few $million wasn't even worth a new product notice. Like many of you, my only recourse was "personal computing"!

The megabuck terminals and mainframes at work seemed boring next to the homebrew TV Typewriter (Don Lancaster's design) and the 8008-based computer I had built at home. Apparently, I wasn't alone.

One of the first personal computing publications I joined was the *Mark-8 Newsletter* from Lompoc, CA. My earliest contribution was a letter detailing how to add scrolling capability to the TV Typewriter. Seeing that I must have some technical capacity, another reader from upstate New York—Maury something--contacted me about designing a high-current, +5-V power supply for computer users. He even sent me a transformer and a functional specification should I be at a loss for an essential ingredient or a definite objective.

Designing and prototyping the 15-A power-supply circuit wasn't so much a question of engineering capability, but securing a cost-effective solution taught me a lot about component power losses, high-current trace paths, and heat dissipation. Today, we take 25-and 50-A switching supplies for granted. But, back then, you either rolled your own or sold your car to buy one.

Eventually, I wrote the whole thing up as a build-it-yourself project and published it in the Mark-8 Newsletter. Besides my first published project, it was an education.

Maury quickly put together a kit of parts to go with the transformers he had in stock and marketed the power supply. When I contacted him about how I fit in his production game plan, he thanked me for finding such a lucrative use for an otherwise "dead stock" transformer and enlightening the readership by publishing it. On this one eventful occasion, I had published my first project, seen it as a manufactured reality, and then gotten a first-hand understanding of what "public domain ownership" can mean.

Today, I write off this experience as just being young and naive. The career chain really started when my next project was stolen and published with another authors name added. Of course, that's a story for another time.

Steve

steve.ciarcia@circellar.com