

CIRCUIT CELLAR

INK[®]

THE COMPUTER APPLICATIONS JOURNAL

#82 May 1997

EMBEDDED PROCESSORS

Embedded Modem Chips

Selecting an 8-Bit
Architecture

Prototyping with the
PowerPC 403GA

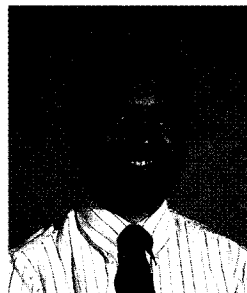
Embedding the
ARM7500



\$3.95 U.S.
\$4.95 Canada

TASK MANAGER

Let's Do Lunch



Networking brings people together—or so I'm told. I've been so busy in the last 10 months or so implementing a network around here in my "spare" time that I haven't had much time to "do lunch", as the Californians say.

What's the big deal with a network? Aside from the true lack of spare time, I've never put together a network before, I'm integrating PCs and Macs, I'm trying to do it economically, and I can't disrupt people's work. In addition, I decided to update our Internet presence by bringing the Web, ftp, and mail servers in-house. Plus, I wanted to make the BBS and all its files available via dialup, telnet, ftp, or Web browser.

Well, we're finally seeing the fruits of my labors. Aside from the free and easy flow of information from desktop to desktop, everybody now has E-mail on their desktop and can access the BBS through the in-house network. BBS users recently started learning the new system and can now access it using the Internet. See my summary of the new system on page 85, and give it a try. It's going to take some time to refine it and squish all the bugs, but the basics are there.

Of course, all these networking changes are having a direct impact on editorial. Gone are the days of hard-copy submissions. Even article proofs now go to authors via Acrobat PDF files. *Circuit Cellar INK* has truly gone digital.

And, how did we do it? Only through your continued work on the front lines, hammering out the processors and software that enable us to do our job better.

This month, we celebrate embedded processors. Walter Banks starts us off with some guidelines on selecting 8-bit architectures, while Randy Heisch gives us the low-down on the PowerPC 403GA, Bill Houghton brings us up to date with the 8xC576, and Art Sobel revisits us with news on the ARM7500.

This info on the big boys is followed up with a reminder by Brad Stewart that the 4-bit chip still commands a lion's share of the market. He points out how powerful these little guys have come to be with integration. Daniel Patten and Michael Miller wrap up our feature section by embedding an Intel 8051 microprocessor in a universal remote-control receiver.

Joe DiBartolomeo closes his four-part series on designing for EMI by looking specifically at how to choose a testing house. Jeff talks about infrared remotes, and Tom takes us off hook with the latest in modem-chip technology.

editor@circellar.com

CIRCUIT CELLAR®

THE COMPUTER APPLICATIONS JOURNAL

EDITORIAL DIRECTOR/PUBLISHER

Steve Garcia

ASSOCIATE PUBLISHER

Sue Hodge

EDITOR-IN-CHIEF

Ken Davidson

CIRCULATION MANAGER

Rose Mansella

MANAGING EDITOR

Janice Hughes

CIRCULATION CONSULTANT

John S. Treworgy

TECHNICAL EDITOR

Elizabeth Laurençot

BUSINESS MANAGER

Jeannette Walters

ENGINEERING STAFF

Jeff Bachiochi

ADVERTISING COORDINATOR

Dan Gorsky

WEST COAST EDITOR

Tom Cantrell

CONTRIBUTING EDITORS

Rick Lehrbaum

Fred Eady

NEW PRODUCTS EDITOR

Harv Weiner

ART DIRECTOR

KC Zienka

PRODUCTION STAFF

John Gorsky

James Soussounis

CONTRIBUTOR:

Pellervo Kaskinen

CIRCUIT CELLAR INK®, THE COMPUTER APPLICATIONS JOURNAL (ISSN 0896-8985) is published monthly by Circuit Cellar Incorporated, 4 Park Street, Suite 20, Vernon, CT 06066 (860) 875-2751. Periodical rates paid at Vernon, CT and additional offices. One-year (12 issues) subscription rate U.S.A. and possessions \$21.95, Canada/Mexico \$31.95, all other countries \$49.95. All subscription orders payable in U.S. funds only, via international postal money order or check drawn on U.S. bank.

Direct subscription orders and subscription related questions to Circuit Cellar INK Subscriptions, P.O. Box 698, Holmes, PA 19043-9613 or call (800) 269-6301. POSTMASTER: Please send address changes to Circuit Cellar INK, Circulation Dept., P.O. Box 698, Holmes, PA 19043-9613.

Cover photograph © 1997 Barbara Swenson

PRINTED IN THE UNITED STATES

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

HAJAR ASSOCIATES NATIONAL ADVERTISING REPRESENTATIVES

NORTHEAST & MID-ATLANTIC

Barbara Best

(561) 694-2044

Fax: (561) 694-2051

B.Best-Hajar@worldnet.att.net

MIDWEST & SOUTHEAST

Christa Collins

(954) 966-3939

Fax: (954) 985-8457

HajarChrista@worldnet.att.net

WEST COAST

Barbara Jones

& Shelley Rainey

(714) 540-3554

Fax: (714) 540-7103

shelley.hajar@worldnet.att.net

Circuit Cellar BBS—24 Hrs., 2400/9600/14.4k bps, 8 bits, no parity, 1 stop bit, (860) 871-1988. For information, mail to info@circellar.com. World Wide Web: 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*®.

Entire contents copyright © 1997 by Circuit Cellar Incorporated. All rights reserved. Circuit Cellar INK is a registered trademark of Circuit Cellar Inc. Reproduction of this publication in whole or in part without written consent from Circuit Cellar Inc. is prohibited.

- 1 4** Selecting an **8-Bit** Architecture
by Walter Banks
- 2 2** A **PowerPC 403GA-Based** Embedded Controller Prototype
by Randy Heisch
- 3 4** Add an I/O Coprocessor to Your Embedded PC
by Bill Houghton
- 4 4** Embedding the ARM7500
Part 1: The Chip and Development Board
by Art Sobel
- 5 2** Four Bits Unleashed
by Brad Stewart
- 5 8** A Universal IR Remote-Control Receiver
by Daniel Patten & Michael Miller
- 6 8** **MicroSeries**
Standards for Electromagnetic Compliance Testing
Part 4: Testing Houses
Toe DiBartolomeo
- 7 4** From the Bench
Infrared Remotes are Everywhere...
If You Can Find 'Em
Jeff Bachiochi
- 7 8** Silicon Update
Modem Déjà Vu
Tom Cantrell

INSIDE ISSUE¹ 82

- 2** Task Manager
Ken Davidson
Let's Do Lunch
- 6** Reader I/O
Letters to the Editor
- 8** New Product News
edited by Harv Weiner

- ConnectTime** 85
Excerpts from
the Circuit Cellar BBS
conducted by
Ken Davidson
- Priority Interrupt** 96
Steve Ciarcia
Processing-A
Matter of Definition
- Advertiser Index** 65

READER I/O

OFF THE MAP

I have a couple comments about "The Global Positioning System" by Do-While Jones (*INK* 77 and 78).

All the GPS receivers I've encountered display their position in WGS-84 or else give the user a choice of display datum. A practical problem is that most U.S. land maps, including such software as **Precision Mapping** (Chicago Map, Inc.) use the North American Datum of 1927 (NAD-27). Depending on where in the country you are, this can give quite a bit of error in transferring your GPS position to a map.

There are two DOS shareware programs that can help. **DATUM** converts one datum to another and includes 99 different systems the author located in worldwide use. (It includes both WGS-84 and NAD-27.) The other program, **GEOD**, calculates surface distance between a pair of Earth coordinates.

Ron Tipton

rtipton@zianet.com

A CALL TO ALARMS

Recently, my family's house was burglarized.

The thieves kicked in the front door and stole two complete AMD K5 computer systems, two televisions, a VCR, a halogen desk lamp, and my daughter's backpack.

Although they took the computers-speakers, mice, surge protectors, and all-they left my checkbook behind. Interestingly enough, they made off with a 6" silicon IC wafer and my copy of **Circuit Cellar INK**. Obviously, they were real connoisseurs!

This type of crime is on the rise. Computers are especially tempting targets because they can be disassembled, making them virtually impossible to trace.

As fellow computer users, don't forget to take steps to deter criminals from selecting your home. Protect your possessions. Invest in an alarm system. Make sure your doors and locks are secure.

Consider marking your valuables, especially the expensive electronics. A permanently marked item is worthless on the black market. Keep an accurate inventory of your possessions (i.e., the make, model, purchase data and price, and serial number).

And, don't make the big mistake I did-always back up your computer data. As it slowly dawns on me how much work I lost, I wish I'd invested a few hundred dollars on data insurance.

Chip Freitag

chip.freitag@amd.com

DON'T FORGET E-MAIL

I know the \$500 Web browser is attractive to many manufacturers since they see a huge, currently untapped market. But, I'd propose they also think about a \$50 E-mail interface with a built-in 2400-bps modem, character-based 80-character x lo-line (or even smaller) LCD, and built-in keyboard, maybe with a parallel port or an Access-bus port to interface to a printer or desktop system.

E-mail used to be the most used 'Net application. I don't know if that's still true, but I think many people could be enticed into Internet computing through such a device, especially since E-mail is primarily text based.

David Bley

d.bley@genie.com

BACK IN THE FAMILY

My dirty little secret is that I read my cousin Chris Arndt's article ("XRaCS: The X-10 Radio-Control System," *INK* 80) in a **borrowed** copy of *INK*! I liked the whole magazine so much, I subscribed on the spot!

I used to read Steve's columns years ago when he wrote for **BYTE**. He stopped the columns, and I dropped **BYTE**. Somehow, I never got caught up again-till now.

L. M. Rappaport

rapp@tiac.net

Contacting Circuit Cellar

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

Mail: Letters to the Editor may be sent to: Editor, Circuit Cellar *INK*, 4 Park St., Vernon, CT 06066.

Phone: Direct all subscription inquiries to (800) 269-6301. Contact our editorial offices at (860) 875-2199.

Fax: All faxes may be sent to (860) 871-0411.

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.com**. Send new subscription orders, renewals, and address changes to **subscribe@circellar.com**. Be sure to include your complete mailing address and return E-mail address in all correspondence. Author E-mail addresses (when available) may be found at the end of each article.

For more information, send E-mail to **info@circellar.com**.

WWW: Point your browser to **www.circellar.com**.

NEW PRODUCT NEWS

Edited by Harv Weiner

I/O CONTROL FOR DEVICENET

The DN502 provides a wide range of analog and digital I/O capability for use with distributed I/O systems using the DeviceNet. OEMs may specify the exact I/O mix for custom applications.

The DN502 contains **16** digital inputs and outputs, 8 analog 12-bit inputs, and 4 analog 12-bit outputs. The digital inputs can be 120 VAC or 24 VAC/DC and the outputs may be specified with relays, 120-VAC triacs, or 24-VDC solid-state drivers. Status indicators show the state of all digital inputs and outputs. The eight 12-bit analog inputs may be software configurable for

0-5 V, 0-10 V, ± 10 V, or a current loop. The four 12-bit analog outputs are configured for 0-10-V operation.

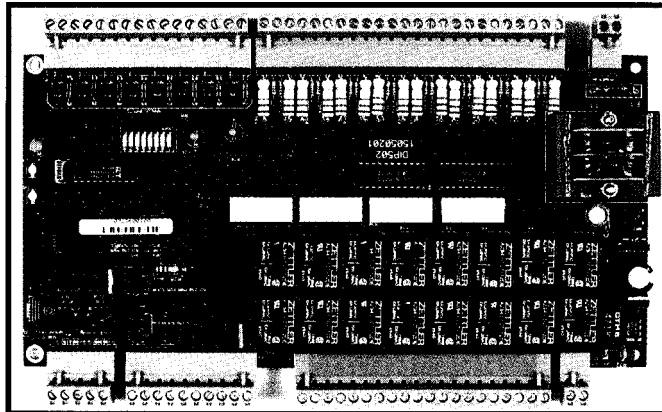
The DeviceNet interface may be configured to be fully isolated up to 500 kbps. A second option permits the DeviceNet power to provide power to the controller. The

MAC ID and operating speed may be configured from an onboard switch or under software.

The fully configured DN502 sells for \$710.

DIP Industrial Products
P.O. Box 9550
Moreno Valley, CA 92552
(909) 924-1730
Fax: (909) 924-3359

#500



EMBEDDED COMMUNICATIONS CONTROLLER

The VCOM-6 is an embedded communications controller that offers full European, U.S., and Canadian agency approvals. The unit is designed for OEM and end-user machine-control applications requiring ruggedness, a variety of I/O, and small size.

The VCOM-6 features a 486DX2-66 SBC with up to 16-MB DRAM. Since operating from solid-state memory greatly improves system reliability, the unit offers 2-32 MB of flash memory configurable as the boot drive. It's well-suited for embedded operating systems such as QNX, VxWorks, WinLight, and ROM-DOS. Program changes can be made remotely via serial port using a software utility program that enables firmware to be downloaded to flash memory. A 128-KB battery-backed SRAM is also included for nonvolatile memory.

The VCOM-6 video controller directly drives flat-panel displays (e.g., CRTs) and features Low Voltage Differential Signaling (LVDS) to lower EMI and RFI. All communications ports (six RS-232 ports capable of 115 kbps, 10BaseT Ethernet interface, and video connector) are mounted in the metal housing for easy access. An optional hard/floppy disk-drive interface card can be added via the unit's PC/104 connector.

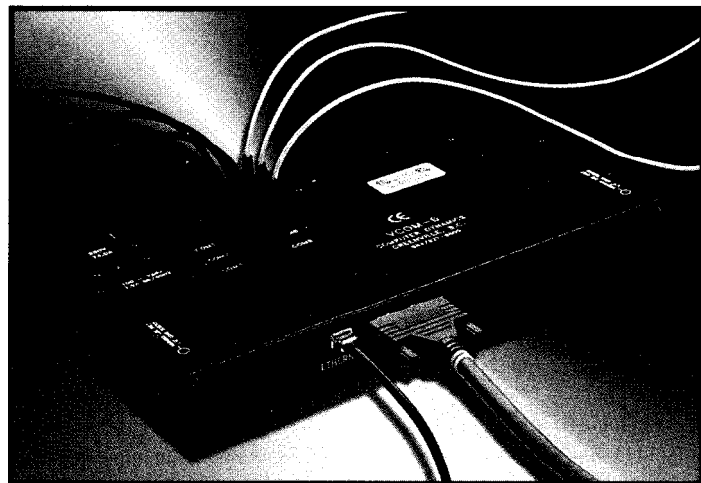
The complete system measures 14" x 9" x 2" and includes a 65-W autosensing power supply that accepts 90-240-VAC input. It operates from 0" to

70°C with no fan required. The VCOM-6 is available with an optional LCD/touchscreen kit that can be mounted remotely. Display choices include 6.4", 8.4", 10.4", 12.1", and 13.8" TFT LCDs with guided acoustic wave, IR, and resistive touchscreens.

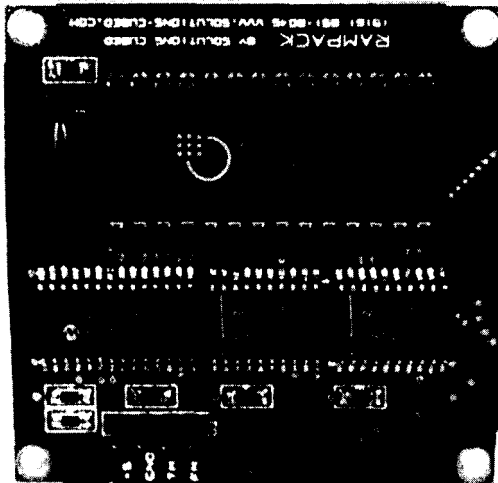
The VCOM-6 sells for \$1425.

Computer Dynamics
7640 Pelham Rd. • Greenville, SC 29615
(864) 627-8800 • Fax: (864) 675-0106

#501



NEW PRODUCT NEWS



SERIAL RAM MODULE

The RAMPack module provides low-cost serial storage of data into 8 KB of RAM. The module employs buffers and an onboard microcontroller to allow addressable access to the RAM using only two I/O lines. RAMPack can store data from data loggers or extend variable space in microcontrollers. It can also serve as a serial data buffer.

An automatic baud-rate detection scheme enables a variety of baud rates to be selected without hardware jumpers. Rates of 1200, 2400, 4800, and 9600 bps are supported using a specific serial 8N1 communications protocol. The SRAM provided with the system is socketed so that it can be replaced with a DS1225AB NVRAM, if desired. NVRAM retains data after power is lost or removed.

The 2" x 2" board is perfect for use with PicStic or BASIC Stamp designs. The RAMPack sells for \$24.95. NVRAM is sold separately.

Solutions Cubed
3029 Esplanade, Ste. F
Chico, CA 95973
(916) 891-8045
Fax: (916) 891-1643
www.solutions-cubed.com

#502

SYSTEM SUPERVISOR CHIP

The TC70/71 is a fully-integrated power-supply monitor, reset generator, watchdog timer, and battery-backup circuit in an 8-pin package. The chips are ideal for applications such as embedded-control and mission-critical microprocessor-based systems. Other typical applications include test equipment, instrumentation, and set-top boxes.

When power is initially applied, the TC70/71 holds the processor in its reset state for a minimum of 500 ms to ensure stable system startup. After startup, processor sanity is monitored by the onboard watchdog circuit. The processor must provide periodic high-to-low level transitions to the chip to verify proper execution. Should the processor fail to supply this signal within the specified timeout period, an out-of-control processor is indicated and the TC70/71 issues a momentary processor reset. The TC70 also features a watchdog disable pin to facilitate system test and debug.

The output of the TC70/71 can be wire-ORed to a push-button switch (or electronic signal) to reset the processor. When connected to a push-button switch, the chip provides contact debounce. The chip's integrated battery-backup circuit converts CMOS RAM into nonvolatile memory by first write-protecting and then switching the V_{CC} line of the RAM over to an external battery. The TC71 incorporates an additional 1.3-V threshold detector for power-fail warning, low-battery detection, or monitoring power-supply voltages other than +5 V.

Pricing for the TC70 or TC71 is \$1.57 in 1k quantities.



TelCom Semiconductor, Inc.
1300 Terra Bella Ave.
Mountain View, CA 94039-7267
(415) 968-9241
Fax: (415) 967-1590
www.telcom-semi.com

#503

NEW PRODUCT NEWS

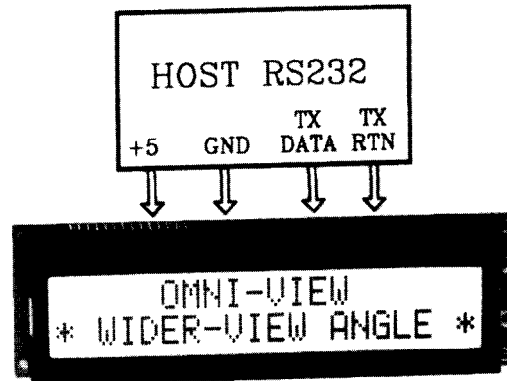
SERIAL-INTERFACE LCD

The CDSxxxC line of character-based LCD modules features a serial RS-232 interface to a host processor. The modules can be used for OEM applications where only serial I/O ports are available or where extended cable lengths for host-to-display data communications are required.

The full line of serial modules includes character displays of 8 x 2, 16 x 1, 16 x 4, 20 x 1, 20 x 2, 24 x 2, 40 x 2, and 40 x 4 (characters x lines). Versions are available with supertwist fluid for wide viewing angle, extended temperature range, and LED or EL backlighting.

The modules accept serial data at one of four jumper-selectable baud rates at RS-232, RS-422, or 20-mA current loop levels. Data word arrangements and parity are also selectable. Display instructions are a serial version of the industry-standard HD44780 LCD controller. Most modules require only +5-VDC input and a single mating connector or cable of 9-1 1 pins. Power consumption is typically less than 35 mA excluding backlighting, if required.

Prices in 100 units for the complete assembly start at \$35 for a reflective supertwist 16 x 2 character display



and go up to \$89 for an LED backlit, supertwist 40 x 4 character display.

Apollo Display Technologies, Inc.
194-22 Morris Ave.
Holtsville, NY 11742
(516) 654-1143
Fax: (516) 654-1496

#504

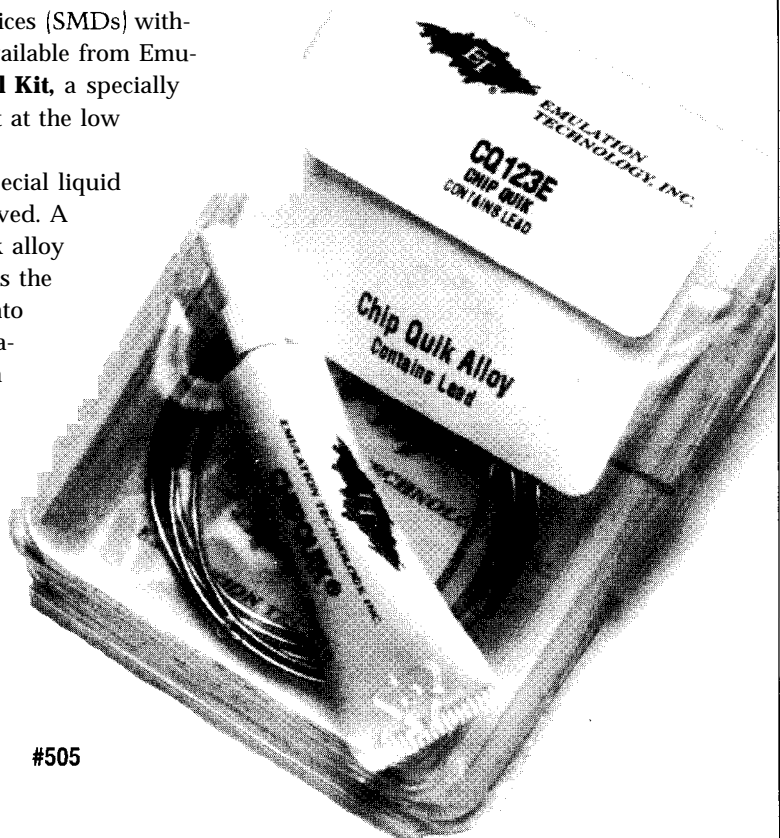
SMD REMOVAL KIT

A method for removing Surface Mounted Devices (SMDs) without damage to circuit boards or components is available from Emulation Technology. The **Chip Quik SMD Removal Kit**, a specially formulated alloy in wire form, is designed to melt at the low temperature of 136°F (58°C).

The removal process is surprisingly easy. A special liquid flux is applied to all leads of the SMD to be removed. A soldering iron is then used to melt the Chip Quik alloy uniformly on all leads of the SMD. The iron keeps the alloy in a molten state long enough to dissolve into the existing solder. The resultant molten temperature of the two alloys causes complete reflow at a temperature less than 220°F (93°C). While in this molten state, the SMD can easily be removed with a vacuum pen. The iron's temperature determines the speed of the process. A typical soldering-iron temperature of 600°F (315°C) works well.

Emulation Technology, Inc.
2344 Walsh Ave., Bldg. F
Santa Clara, CA 95051-1301
(408) 982-0660
Fax: (408) 982-0664

#505



NEW PRODUCT NEWS

MACRO CROSS-ASSEMBLER

The Windows-based Cross-32 **Meta-Assembler** features table-driven macro cross-assemblers that compile assembler programs for over 50 processors, controllers, and DSPs. The tables use the chip manufacturer's original assembly-language mnemonics. Full instructions are included so you can create new tables for other chips.

Cross-32 reads an assembly-language source file and a corresponding assembler instruction table and then writes list, error, and absolute hexadecimal machine files in binary, Intel, or Motorola formats. The hexadecimal file can then be downloaded to most EPROM programmers, EPROM emulators, and in-circuit emulators.

V.4.0 includes support for many new chips, such as Hitachi's H8/300H, Intel's 80251, Motorola's 68HC12, NEC's 78/KII, Philip's 805 1XA, and Zilog's 2380, among others. The text editors handle up to a billion lines of code with adjustable tab sizes and Windows font selection. A literal text replacement directive (RPTXT) lets you replace almost any register or variable with any text string, making code easier to read and the assembler more versatile. Support for IEEE floating-point number definitions (DFF) is provided, along with the option of making the assembler case sensitive (CASE).

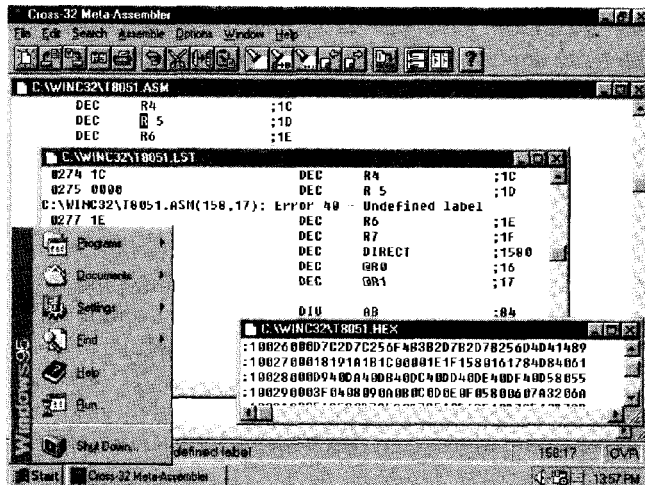
The Cross-32 Meta-Assembler sells for \$199.

Universal Cross-Assemblers

9 Westminster Dr. . Quispamsis, NB . Canada E2E2V4

(506) 849-8952 . Fax: (506) 847-0681

#506



C-PROGRAMMABLE CONTROLLER

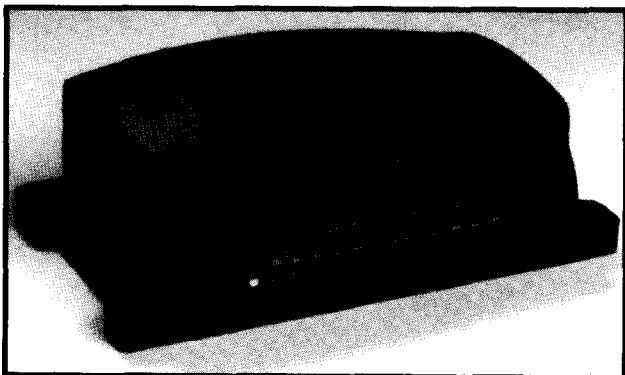
The Z-World PK2300 is a versatile controller that contains user-configurable I/O providing up to 16 protected digital inputs and 8 high-current outputs. Initially, the 19 I/O lines are set as 11 inputs and 8 outputs, but 5 of the outputs and 6 of the inputs are jumper selectable. Possible configurations include level-sensitive interrupts and protected inputs, an analog-resistive input, and an RS-485 port. Screw terminals facilitate quick wiring, and the rugged enclosure easily mounts to a flat surface or any of the three DIN rail sizes.

The PK2300 features high-current outputs that can drive inductive loads (e.g., solenoids and relays). It also

includes KS-232 and RS-485 serial communication ports, power supervision, a real-time clock, and a resistive sensor-measurement input for capturing temperature, position, and potentiometer values. Flash memory enables nonvolatile storage of program code and data and facilitates remote programming. Easy-to-use software drivers are included for all I/O, including PWM for seven of the eight high-current drivers. PWM frequency and resolution are adjustable under software control.

The PK2300 is programmed using Z-World's Dynamic C, a version of the industry-standard C programming language optimized for real-time control. Dynamic C is a software development system that's an integrated editor, compiler, and interactive debugger. The compiler, running on a host PC, compiles directly into the 128 KB of flash memory of the PK2300 for in-target software development. This feature eliminates the need for expensive test equipment (e.g., ROM or in-circuit emulators).

Pricing in single quantities is \$179.



Z-World

1724 Picasso Ave. . Davis, CA 95616

(916) 757-3737 . Fax: (916) 753-5141

www.zworld.com

#507

NEW PRODUCT NEWS

CHEMISTRY-INDEPENDENT BATTERY CHARGER

The MAX1647/MAX1648 are chemistry-independent battery-charger ICs, each capable of charging LiIon, NiCd, NiMH, and other battery types. The chargers provide 1 1-bit resolution for control of the charging current (4, 2, or 1 A max.) and 10-bit resolution for control of the applied compliance voltage (18 V max.).

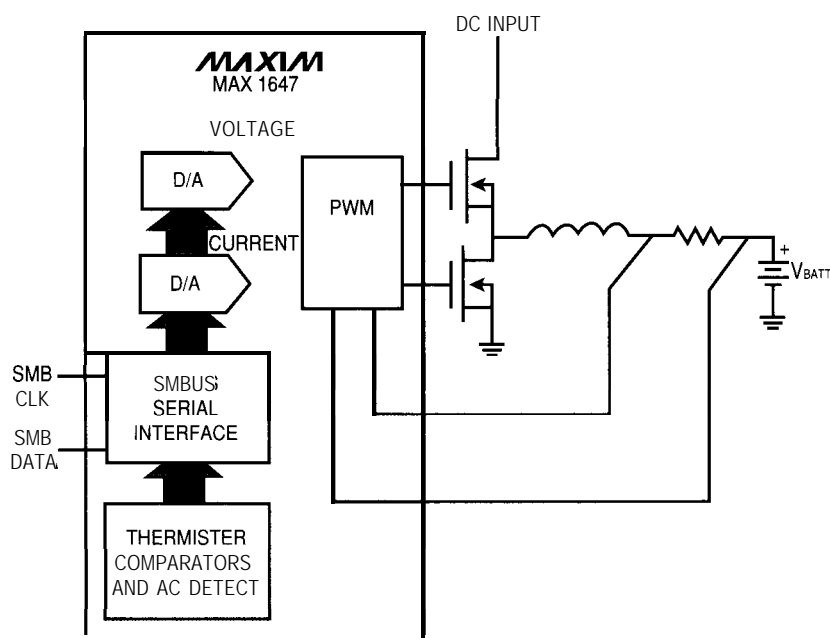
The MAX1648 delivers charging voltage and current to the battery in proportion to control voltages applied to its input pins. The MAX1647 sets these parameters via its interface to the two-wire System Management Bus (SMBus) by Intel. This bus enables the MAX1647 to set the charging voltage and current and provide thermal status information to the external system.

The MAX1647 is compliant with the Duracell/Intel Smart Battery Charger specification as a Level 2 charger. In addition, it generates interrupts that signal the host when power is applied to the charger or when a battery is installed or removed. Other status bits enable the host to check whether the charger has enough input voltage and whether the battery current and voltage are properly regulated. This capability allows the host to determine, without interrogating the battery, when a LiIon battery is fully charged.

The MAX1647 is available in a 20-pin SSOP, and the MAX1648 is available in a 16-pin DIP. Prices start at \$4.75 for the MAX1647 and \$4.25 for the MAX1648 in quantity.

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

#508



SUPERSCALAR MICROPROCESSOR

The MC68EC060 is an implementation of an MC68060 optimized for embedded applications. It provides the highest level of 680x0 superscalar integer performance of 102 MIPS at 66 MHz. Its high performance, low power consumption, and economical pricing make it an excellent solution for cost-sensitive advanced applications in embedded control.

The MC68EC060 employs a deep-pipeline, dual-issue superscalar execution, a branch cache, and 8 KB each of on-chip instruction and data caches. Its architecture permits simultaneous execution of one branch and two integer instructions during each clock cycle. The microprocessor automatically minimizes power dissipation via fully static design, dynamic power management, and low-voltage operation. It also automatically powers down internal functional blocks that aren't needed on a clock-by-clock basis. Power consumption can be controlled from the operating system.

The chip is available in a 206-lead ceramic Pin Grid Array (PGA) and operates off of a single 3.3-V power supply. It is also 5-V input tolerant. The list price for the 50-MHz version of the MC68EC060 is \$75 in quantity. The 66-MHz version costs \$180 in quantity.

Motorola
Communications and Advanced
Consumer Technologies Group
6501 William Cannon Dr. W
Austin, TX 787358598
(512) 891-2134
Fax: (512) 891-4465

#509

FEATURES

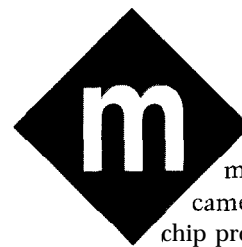
- 14** Selecting an 8-Bit Architecture
- 22** A PowerPC 403GA-Based Embedded Controller Prototype
- 34** Add an I/O Coprocessor to Your Embedded PC
- 44** Embedding the ARM7500
- 52** Four Bits Unleashed
- 58** A Universal IR Remote-Control Receiver

FEATURE ARTICLE

Walter Banks

Selecting an 8-Bit Architecture

With careful management of a chip's resources, you can use a lower end chip to accomplish the same task. Walter shows that tweaking here and there is just another way to get more bang for your buck.



Most embedded microcomputers came from a single-chip processor of the 1970s. And, those were based on the minicomputers of a decade earlier.

The dominant processor architecture was the classical Von Neumann computer with a uniform address space containing memory that could be used for either program or data storage. Figure 1 shows the Von Neumann architecture.

The systems' designs were simple and easy to program. Most of these computers used a small number of general-purpose registers for arithmetic calculations.

Designers found that an application's execution speed depended on the rate that data could be passed through the registers performing calculations. By increasing the number of general-purpose registers or by making access to the data registers quicker, substantial improvements in performance could be achieved while still using the same basic logic-operation speed.

By overlapping access to the instruction with data accesses, a style of processor with separate instruction and data spaces was developed. This style was further optimized by tailoring the size of the instruction and data space to the intended application's requirements.

Extending the width of the instruction word (i.e., increasing its parallelism) meant fewer instruction fetches were needed to execute a given task, resulting in even better system performance. It became reasonable that data paths were 4 or 8 bits and instruction paths were 12, 16, or more bits.

CARRY IN SUBTRACTS

Of all computer instructions, subtract is the one instruction that's least consistently implemented. Here's a nonexhaustive list of different implementations:

register \leftarrow register + *memory + 1
 register tregister + (*memory + 1)
 register \leftarrow register + *memory + carry
 register t register + *memory + *carry

The first two implementations may seem identical, and certainly they generate correct results. But, when zero is the value in memory, the carry bit is set in the first case and cleared in the second case.

In many microprocessors, the compare instruction (if it exists) is based on the subtract, so interpreting results can be difficult.

Let's look at an example where both *register* and *memory* are initially both equal to 0:

register \leftarrow register + *memory + 1

results in register = 0 and carry = 1. But, when:

register \leftarrow register + (*memory + 1)

then register = 0 and carry = 0.

The resulting carry may be different depending on the order of execution in the processor ALU. This can change how compares are evaluated between processors with similar architectures.

MEMORY-TO-MEMORY MOVES

On the surface, memory-to-memory move instructions appear to be a small savings in the ROM requirements as well as in one less opcode fetch. This is certainly true for data initialization.

Another saving from improved dataflow is often overlooked. Many times, a value has to be read from a

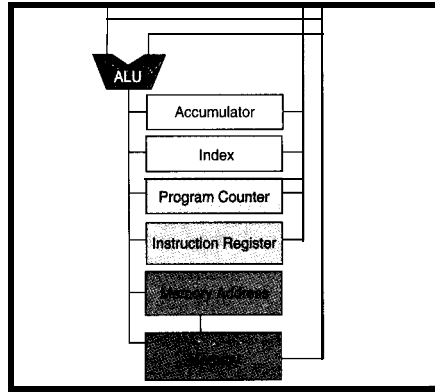


Figure 1—Classical Von Neumann computer architecture stores both program and data information in the same address space.

sensor or port, saved, and used later for calculations. Avoiding the use of the accumulator means that code savings can be substantial, saving and restoring the accumulator as well as the saved opcode fetch.

OVERFLOW BIT

Of all the condition code bits, the overflow bit (V bit) is the least understood. Primarily, its role is akin to the carry bit for signed numbers. A surprising number of embedded microcomputer applications are implemented on processors lacking an overflow bit.

Signed numbers are represented by a circular number system whose smallest number has only the most significant bit (MSB) set and the largest has only the MSB cleared. This system is unlike unsigned numbers where the smallest number is all 0s and the largest is represented by all 1s.

During implementation, the code to compare numbers is the same for equality and nonequality for both signed and unsigned number systems.

Most microcomputer instruction sets lacking a V bit can clearly compare unsigned magnitudes, but signed compares of relative magnitudes can be a considerable problem. Most traditional approaches fail in some of the many possible combinations. (In compiler implementations I've done, there are rules for no fewer than 55 special cases of comparisons.)

The need for the V bit is extremely application dependent. To implement a signed compare without a V bit, I rotate the number system when executing comparisons so the signed number

system ranges from all 0s to all 1s. All comparisons are then executed with just the carry and zero bits.

Listing 1 compares the generated code for MC68HC05 and MC68HC08. The MC68HC08 has a V bit, and the MC68HC05 does not.

RISC IN EMBEDDED SYSTEMS

At least two vendors—Microchip and Sanyo—have developed RISC-architecture embedded micros that execute at essentially one instruction per clock cycle. In both cases, they use a Harvard architecture (see Figure 2) with different instruction word and data sizes.

The Sanyo processor instruction space is 16 bits, and the data space is 4 bits. This part is targeted to low-power, low-speed, high-volume applications involving time and LCDs (e.g., watches and consumer products).

RISC meets the processor's need for low power. The clock rate is 32 kHz, and the processor rate is 4096 instructions per second.

Microchip's PIC 16/17Cxx processors are unique. The instruction paths are 12, 14, or 16 bits wide, and the data path is 8 bits.

The RISC aspect of Microchip's processor is only one of its advantages. The instruction set is designed so many of the traditional load-operate-store sequences can be executed with the two-instruction load-operate sequence. By combining the operate-store pair in one cycle, the PIC processors can be impressively fast.

The separate instruction and data space of both these processors cause them to share the problems of constant data found in most Harvard-architecture computers. Constant data arrays are either not available or very difficult to reach.

In a similar manner, RAM addressing in a Harvard-architecture-based processor is a compromise of code representation, size, and address space. Instruction-set designers solve this problem by extending the number of addressing modes or implementing a RAM memory-management scheme.

Almost all Harvard-architecture-based processors have a memory-management scheme available. This isn't a

problem, but to achieve efficient code, care must be taken in locating variables in RAM.

What about I/O? There are three common ways of implementing I/O on embedded computer systems:

- map the I/O control registers into the data space on the microcomputer
- develop a separate I/O address space serviced with extra I/O instructions
- add I/O application-specific instructions

Starting with an informal survey of Motorola 68HC05, National COP8, Intel 8051, Microchip PIC series of processors, and Zilog Z8, I discovered remarkable similarities between the offerings from these 8-bit processor families. Although not included in the list, the Motorola 68HC11 and Zilog Z80 also have similar I/O resources.

Together, these processor families have 95% of the market in application designs and production volume. Although functionally similar, each family has implementation quirks that tax potential portability between platforms.

PORTS

Port support is perhaps the simplest I/O support for embedded systems, but initializing the data direction-control registers complicates it. On some processors, initialization requires a 1, and others, a control bit of 0.

The serial-port I/O support for both synchronous and asynchronous serial protocols is available from many vendors. Most serial-port implementations contain baud-rate generators and, in some cases, hardware to assist in implementing synchronous serial protocols.

This family of serial I/O support buses can provide a wide variety of I/O support facilities ranging from inter-processor communications to interfaces with external serial and parallel ports, RAM, ROM, EEPROM, ADCs, and DACs. Multiprocessor implementations of sub \$1 microcontrollers are using this technology for interprocessor communications.

CONVERSIONS

Embedded microcomputers have between one and eight analog input

channels with current resolution between 8 and 16 bits.

Most ADCs need some general setup. Typical setups require a reference source and sometimes resolution and conversion time parameters. In looking at application code, I've noticed that this doesn't change over the course of code execution in an application.

Very few embedded systems have a DAC built into them. In some applications, PWM ports generate an analog output voltage smoothed with a simple low-pass filter.

PWM ports are flexible output ports that can generate levels, sophisticated pulse trains, and with a simple low-pass filter, analog outputs. The number of PWM channels varies from 2 to 16, with the generated pulse-stream resolution between 6 and 14 bits.

WATCHDOG TIMERS

Embedded systems use watchdog timers as a check against runaway execution of code. Their hardware implementation varies considerably between different processors.

In general, these timers must be turned on once, often within the first few cycles, after being reset. They must then be reset periodically within the software. Some can be programmed for different time-out delays.

The reset sequence can be as simple as a specialized instruction or as complex as sending a sequence of bytes to a port. Watchdog timers either reset the processor or execute an interrupt on timeout.

HIGH-LEVEL LANGUAGES

High-level languages, the most common being C, are frequently used on embedded microcomputers. There are several standard arguments—development ease, maintainability, portability—to using high-level languages.

High-level languages let you avoid the need for alternative sources for the embedded processors. A well-supported high-level language can enable applications to run on several platforms.

Early implementations of C tended to be based on a mythical model of an ideal C machine. The elements of this model were then run on various target micros either by making library calls or through macro expansion.

Inline code was, for the most part, generated by using a subset of the whole instruction set. The result was a quick implementation with good portability between various targets. This approach was useful for prototyping and applications where execution time was not too critical, involving low production volumes.

With the arrival of the PC and the need for better application performance and reasonable costs, a new era of software tools began to appear in the marketplace.

Good optimizing C compilers are now available for most common embedded microcomputers. Their code competes with well-written assembler, requiring technology that could exhaustively optimize the generated code. Here's just how complex the task can be:

Listing 1a—The MC68HC05 code compares two signed numbers using the carry bit as an overflow indicator.
b—The MC68HC08 uses an overflow bit for comparing signed numbers.

```
a)
0100 B6 50 LDA $50      if (i < 6) i = 29;
0102 A8 80 EOR #$80    // Load I rotate number system
0104 A1 86 CMP #$86    // Compare against 6 in new number system
0106 24 04 BCC $010C
0108 A6 1D LDA #$1D
010A B7 50 STA $50
010C 81 RTS

b)
0100 B6 50 LDA $50      if (i < 6) i = 29;
0102 A1 06 CMP #$06
0104 90 03 BGE $0109
0106 6E 1D 50 MOV #$1D,$50
0109 81 RTS
```

```

0100 B6 44 LDA $44 i = i | 29;
0102 AA 1D ORA #$1D
0104 B7 44 STA $44

0106 C6 21 53 LDA $2153 j = j | 41;
0109 AA 29 ORA #$29
010B C7 21 53 STA $2153

010E 1C 44 BSET 6,844 i = i | 64;

0110 AA 40 ORA #$40 j = j | 64;
0112 C7 21 53 STA $2153

```

In each source line, syntactically identical code generates different results. The generated code varies between two and eight bytes, depending on data location, known data, and previous execution history in the application.

WHICH MICRO?

There are many available micro-computers on the market for embedded applications. At last count, about 50 varieties were competing for engineers' embedded designs. Just say the word, and there will be a flood of data-books and helpful field-application engineers at your door.

Think about the important issues in the application. To name a few, the

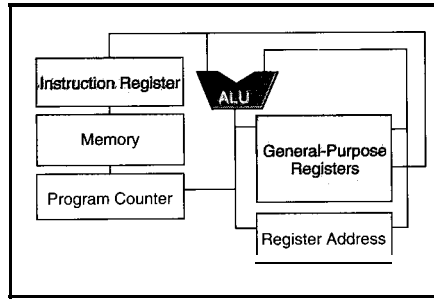


Figure P-Harvard architecture has separate code and data memory spaces.

underlying technology must address issues of power consumption, number of I/O pins, cost, size, and availability. Timely execution of the application code and the ability to transfer the application algorithms to the underlying silicon are also important.

SUPPORT

Embedded applications are complex, usually requiring both a clear understanding of the application area and the implementation technology. Chip-vendor field-application engineers are usually well-equipped to provide answers to all your questions.

Support also comes from vendor seminars, conferences, reference material, and software-tool vendors. Most silicon and software-tool vendors have Web sites supporting their products. These sources give you clues about the acceptability of a particular product or tool for a specific application.

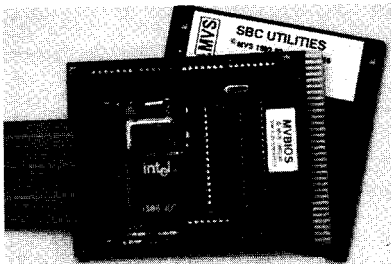
Finally, talk to the people who develop the tools. These individuals are usually willing to frankly discuss the strengths (and weaknesses) of their product.

TIMES ARE CHANGING

Starting about five years ago, most silicon vendors competed against each other with functionally similar products. Single-chip computers had some onboard RAM, ROM, or EPROM and perhaps 10 pins of I/O. At least half of the top-ten selling micros could easily be substituted with minimum board redesign and receding an already working application.

Application designers are now able to put out layout boards that accept silicon from more than one vendor.

386 SBC \$83



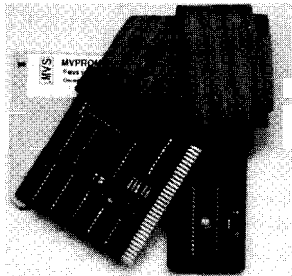
OEM (1K) PRICE INCLUDES:
 - 5 SER (8250 USART)
 - 3 PAR (32 BITS MAX)
 - 32K RAM, EXP 64M
 - STANDARD PC BUS
 - LCD, KBD PORT
 - BATT. BACK. RTC
 - IRQ0-15 (8259 XT2)
 - 0237 DMA 0253 TMR
 - BUILT-IN LED DISP.
 - UP TO 8 MEG ROM
 - CMOS NVRAM

USE TURBO C, BASIC, MASM RUNS DOS AND WINDOWS EVAL KIT \$295

\$95 SINGLE PIECE PRICE

UNIVERSAL PROGRAMMER

-DOES 8 MEG EPROMS
 -CMOS, EE, FLASH, NVRAM
 -EASIER TO USE THAN MOST
 - POWERFUL SCRIPT ABILITY
 - MICROCONT. ADAPTERS
 - PLCC. MINI-DIP ADAPTERS
 -SUPER FAST ALGORITHMS



OTHER PRODUCTS:
 8088 SINGLE BOARD COMPUTER OEM \$27 ... '95
 PC FLASH/ROM DISKS (128K-16M) 21 75
 16 BIT 16 CHAN ADC-DaC CARD 55 195
 WATCHDOG (REBOOTS PC ON HANGUP) 27 95

EVAL KITS INCLUDE MANUAL BRACKET AND SOFTWARE. 5 YR LIMITED WARRANTY FREE SHIPPING HRS: MON-FRI 10AM-6PM EST

MVS BOX 850 MERRIMACK, NH (508) 792 9507

PIC16Cxx C COMPILER

- Integrated software development environment (IDE) including an editor with interactive error detection/correction. A DOS command line compiler is also included.
- Access to all PIC hardware features from easy to use C functions.
- Built in libraries that work with all chips for RS232 serial I/O, I²C, discrete I/O and precision delays.
- Formatted Printf allows easy formatting and display in Hex or decimal.
- Source code drivers included for LCD modules, keypads, 24xx and 93xx serial EPROM's, X10, DS1302 and NUJ6355 Real Time Clocks, Dallas Touch Memory Devices, DS2223 and PCF8570 Serial SRAM, LTC1298 and PCF8591A/D converters and more
- Efficient function implementation allows call trees deeper than the hardware stack.
- Features such as bit variables are optimized for the unique hardware capabilities.
- Functions that call one another frequently are grouped together in the same page and calls across pages are handled automatically by the tool transparent to the user.
- Assembly code may be inserted in the source and may reference C variables.
- Integrates with MPLAB and other simulators/emulators for source level debugging. Standard Hex file and debug files ensure compatibility with all programmers.

```

#include <16C71.H>
#use delay(clock=15000000)
#use rs232(baud=9600, xmit=PIN_B0,rcv=PIN_B1)

main() {
  int value;

  setup_port_a(ALL_ANALOG);
  setup_adc(ADC_CLOCK_INTERNAL);
  set_adc_channel1(PIN_A1);

  printf("Sampling pin A1:\r\n");

  do {
    value = read_adc();

    printf("A/D value: %2x\r\n", value);
    delay_ms(1000);
  } while (TRUE);
}

```

PCB (DOS IDE) \$99

for 52-58 and 12Cxx

PCM (DOS IDE) \$99

for 61-84 and 14000

Professional Package \$350

Both PCM & PCB with a Windows IDE

Software Prototyping Card \$145

LCD/KBD for above \$55

Priority Shipping \$7

CCS INC. PO Box 2452

Brookfield, WI 53008

414-781-2794 x30

Fax 414-781-3241

http://www.execpc.com/~ccs/picc

ccs@execpc.com

Effective high-level language tools enable application code to be retargeted easily between the most popular processors.

Maybe I'm venturing too far into the future, but I'm starting to see vendors of standardized VHDL models waiting for the day when a large FPGA implements whole applications.

At the engineering level, local support is crucial. Getting answers to hard technical questions ranks as an engineer's most critical need.

From my experience with both vendors and customers from various competing companies, I know that the single most important issue to getting a product out to the marketplace is development support, closely followed by product availability. □

Special thanks to Bruno Bratti, who surveyed the I/O facilities on the most popular microcomputer products. His work served as a foundation for mine.

Walter Banks is president of Byte Craft Limited, a company specializing

in software tools for embedded microprocessors. His interests include highly reliable system design, code-generation technology, programming-language development, and formal code-verification tools. You may reach him at walter@bytecraft.com.

SOURCES

8051

Intel Corp.
5000 W. Chandler Blvd.
Chandler, AZ 85226-3699
(602) 554-8080
Fax: (602) 554-7436
www.intel.com

PIC16/17Cxx

Microchip Technology, Inc.
2355 W. Chandler Blvd.
Chandler, AZ 85224-6199
(602) 786-7200
Fax: (602) 786-7277
www.microchip.com

68HC05

Motorola
MCU Information Line

P.O. Box 13026
Austin, TX 7871 1-3026
(512) 328-2268
Fax: (512) 891-4465

COP8

National Semiconductor
P.O. Box 58090
Santa Clara, CA 95052-8090
(408) 721-5000
Fax: (408) 739-9803

RISC architecture embedded micros
Sanyo Semiconductor Corp.
453 Ravendale Dr., Ste. G
Mountain View, CA 94043
(415) 960-8591
Fax: (415) 960-8591

Z8

Zilog, Inc.
210 E. Hacienda Ave.
Campbell, CA 95008-6600
(408) 370-8000
Fax: (408) 370-8056

IRS

401 Very Useful
402 Moderately Useful
403 Not Useful

Parts List Manager

For Engineers, Product Designers, Prototypers

Windows-based software helps you stay organized...And makes the job easier!

Easy to create and manage multi-assembly parts lists for products in development...And after!

Keep track of:

- Part Specs & Dwg's
- Suppliers & Mfrs
- Boms and Kit Lists
- Product Costs
- Engineering Stock



Parts & Vendors™ Version 2.0

Version SE: \$99 + s/h Extended: \$299 + s/h

Call 800-280-5176

916-273-1985
fax 916-477-9106

Requires 486, 10 meg min. ram. Win 3.x or Win95

P.O. Box 2270, Grass Valley, CA 95945
http://www.trilogydesign.com

Trilogy DESIGN

Real-Time Multitasking Tools for Embedded Systems and DOS

RTKernel

Professional, high-performance real-time multitasking system for DOS and 16-bit Embedded Systems.
For Borland C/C++, Microsoft C/C++, and Borland Pascal.
Libraries: \$550 Source Code: add \$500

RTTarget-32

Cross Development System for 32-bit Embedded Systems.
Supports Intel 386 and higher, as little as 16KRAM/ROM.
for Borland C/C++, Microsoft C/C++, and Watcom UC++.
Libraries: \$1700 Source Code: add \$1000

RTKernel-32

Professional, high-performance real-time multitasking system for 32-bit Embedded Systems.
Supports Intel 386 and higher.
for Borland C/C++, Microsoft UC++, and Watcom C/C++.
Libraries: \$1950 Source Code: add \$1650

In North America, please contact:

On Time
88 Christian Avenue, Setauket, NY 11733, USA
Phone (516) 689-6654
Telefax (516) 689-1172
email info@on-time.com

Other Countries:

On Time Informatik GmbH
Hofweg 49, 22085 Hamburg, GERMANY
Phone +49-40-2279405
Fax +49-40-2279263
email fo2212.3101@compuserve.com

NO ROYALTIES! FREE DEMO DISK!
<http://www.on-time.com>

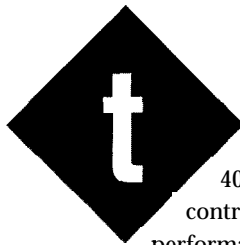
On Time
REAL-TIME AND SYSTEM SOFTWARE

FEATURE ARTICLE

Randy Heisch

A PowerPC 403GA- Based Embedded Controller Prototype

Randy takes a look at the PowerPC 403 architecture before popping it into a prototype implementation. As you'll see, the 403GA rides the midrange price/performance ground.



The IBM PowerPC 403GA embedded controller is a high-performance 32-bit RISC processor that supports the PowerPC user instruction-set architecture and provides a host of on-chip functions.

Targeted for applications such as printers, copiers, and PDAs, the 403GA is the first in IBM's 400 series of embedded controllers. It represents the middle ground in price performance between the low-end 401 and high-end 405 embedded controllers.

Its on-chip capabilities support direct attachment of memory and I/O devices, a 4-channel DMA controller, an asynchronous interrupt controller, multiple timers, and a high-speed serial port.

In this article, I overview the PowerPC

403 architecture and describe it in a minimal prototype implementation.

POWERPC FEATURES

As shown in Figure 1, the 403GA features direct attachment to memory devices (including DRAM), byte and half-word peripheral/memory support, and separate 2-KB instruction and 1-KB data caches.

It has single-cycle execution of most instructions as well as fast interrupt service. It also includes an on-chip serial port, hardware multiply and divide, and thirty-two 32-bit general-purpose registers (GPRs).

The 403GA comes in 25- and 33-MHz versions. It operates at low power (typically 200 mW) and interfaces to both 3.3- and 5-V devices.

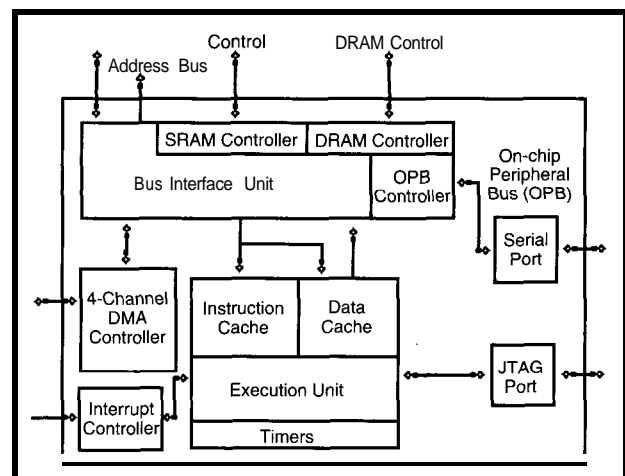
The RISC core contains the execution unit, split instruction and data caches, and a robust set of timer facilities, including a 56-bit time base register, a 32-bit programmable interval timer, a fixed interval timer, and a watchdog timer.

Both the instruction and write-back data caches are two-way set associative with 16-byte lines. The instruction cache can supply two instructions per cycle to the execution unit, which enables predicting and folding out of branch instructions (i.e., zero cycle branches).

Both caches load an entire line on a cache miss and provide bypass forwarding with programmable target-word first or sequential load modes.

The execution unit includes instruction fetch, decode, execute, queue management, and branch prediction and folding logic. All instructions

Figure 1—The 403 contains a highly pipelined processor core with separate instruction and data caches as well as multiple peripheral interface units, including direct-attach DRAM control, chip-select logic, DMA, and serial-port capability.



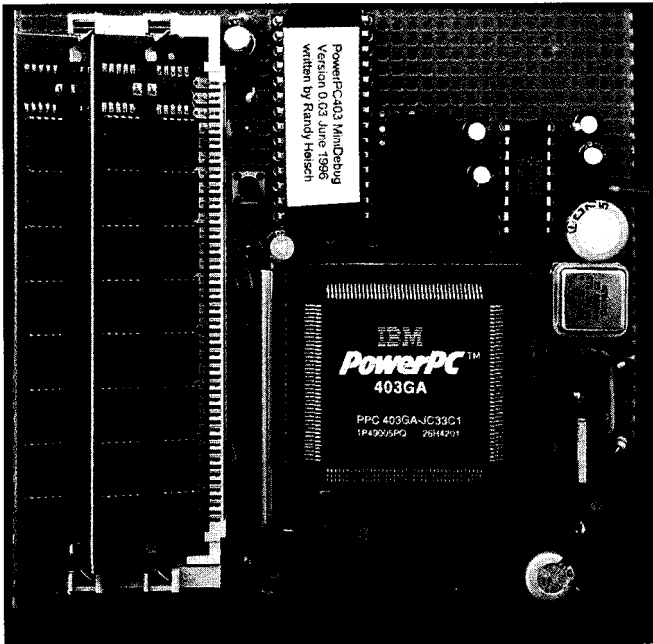


Photo 1—The wire-wrapped prototype hardware demonstrates a near-minimal implementation with processor, EPROM, DRAM SIMMs, and miscellaneous parts (e.g., crystal, RS-232 I/F, and power conditioning) contained in a 4" x 5" form factor.

execute in a single cycle with the exception of load/store multiple, load/store string, multiply, and divide.

The execution unit also contains thirty-two 32-bit GPRs and a set of special-purpose (for in-core control) and device-control (for outside-core control) registers.

The bus interface unit (BIU) supports direct attachment of combinations of 8-, 16-, and 32-bit DRAM or static (SRAM, ROM, or I/O) memory devices in eight-bank groups. Memory access parameters (e.g., set-up and hold cycle times and device width and size) are programmable for each bank.

All eight banks can be programmed for static memory support. The upper four banks (4-7) support configuration for either static or DRAM devices. Each bank controls an external chip-select signal (or DRAM RAS signal).

Interfacing byte-, half-word-, or word-wide DRAMs in 1-64-MB chunks is a breeze. External RAS and CAS signals connect directly to DRAM devices, offering access timing, precharge cycles, refresh interval, bank size/usage, and device width. Other parameters are fully programmable per DRAM bank.

The 32-bit address space is logically partitioned into 32-128-MB regions that may be cacheable or noncacheable for both I and D caches. If a region is noncacheable, load/store references or instruction fetches bypass the cache and directly access the target device-

used for external memory devices, so it results in dual mapping for DRAM and static devices. Both DRAM and static memory decode split into two cacheability regions, providing two groups of chip-select areas (or bank-register memory regions) that can be selectively cached for both memory types.

DRAM addresses need address bits A1-A3=000, and static devices require A1-A3=111. The BIU address-decode logic effectively compares the 8-bit base address select (BAS) fields from each bank register to address bits A4-A1 to generate the chip-select signals.

The DMA controller provides four independent direct access memory channels and includes BIU buffered memory-to-peripheral, fly-by (BIU bypass) memory-to-peripheral, and memory-to-memory transfer modes (supporting mismatched access time memories). Memory-to-memory transfers can also be initiated from software.

definite requirement when interfacing to most I/O devices.

Figure 2 shows the 403GA memory map. Address bit A0 (the MSB) isn't

Figure 2—DRAM and static (SRAM, ROM, and I/O) memory regions are fixed in lower and upper address ranges, respectively. Each memory type can be partitioned as cacheable or noncacheable. The most significant address bit (A0) is ignored, resulting in a double mapping across the address space.

Addr bits A0-All	Cache Region	Memory Type	Address Range
0000 0000 0011 012345676901	0 & 16	DRAM (BR4-BR7)	0x00000000 & 0x80000000 0x07FFFFFF & 0x87FFFFFF 0x08000000 & 0x66000000 0x0FFFFFFF & 0x8FFFFFFF
0001 0000 0000 001111111111	2-7	Reserved	0x10000000 0x3FFFFFFF
010000000000 010011111111	9-13	Serial Port	0x40000000 0x47FFFFFF
0100 10000000 011011111111	15 & 31	Reserved	0x48000000 0x6FFFFFFF
x111 00000000 x110111 1111		SRAM/ROM & I/O (BR0-BR7)	0x70000000 & 0xF0000000 0x77FFFFFF & 0xF7FFFFFF
x111100000000 x1111111 1111		Reserved	0x76000000 & 0xF8000000 0x7FFFFFFF & 0xFFFFFFFF
1001 00000000 11011111 1111		Reserved	0x90000000 0xEFFFFFFF

The built-in serial port provides RS-232 serial communication as well as a high-speed mode timed from 1/16 of the processor clock. The serial port offers internal loopback and auto echo modes and may be a peripheral for DMA transfers. Its control and status registers are located at various offsets from address 0x40000000.

Asynchronous interrupts from external sources and internal DMA, serial port, and debug inputs are supported through the on-chip interrupt controller. Five external interrupt inputs are individually programmable as active high or low and edge triggered or level sensitive. A negative edge-triggered critical interrupt is also provided.

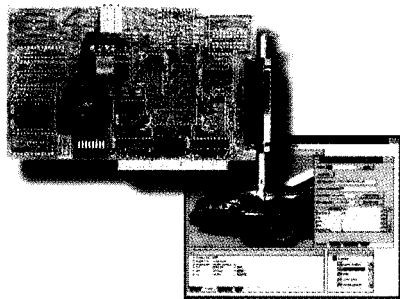
PROGRAMMING MODEL

Figure 3 shows the 403GA register set. The PowerPC User Instruction Set Architecture (UISA) is supported and includes the thirty-two 32-bit GPRs, an 8-field condition code register (CR), the fixed-point exception register (XER), the link register (LR), and a branch-on count (CTR) register.

GPRs 0-31 are supported by nearly all instructions. GPR 0 is used differently for certain effective address calculations, and GPRs 1 and 2 are reserved for the stack pointer and table of contents (TOC) register by convention only.

The condition code register is divided into eight 4-bit fields (CRO-7), with each field containing LT, GT, EQ, and SO bits. The XER provides carry and overflow bits and serves as the transfer byte count for load/store string instructions. LR holds the return address for the branch and link (b1) instruction, the PowerPC CALL.

In true RISC fashion, the programmer (or compiler) has the freedom and



Lowest Cost Data Acquisition

ADAC's new Value-Line has uncompromising design features and high quality components at prices below the low cost guys!

Just check out the specs:

Lowest Cost

5500MF
8 channels 12-bit A/D,
16 digital I/O, Counter/Timer

\$195

High Speed

5508LC
8 channels 12-bit A/D,
100KHz, DMA

\$245

Multi-Function DMA

5516DMA
16 channels 12-bit A/D,
DMA, 16 digital I/O

\$295

High Resolution

5500HR
16 channels 1d-bit A/D,
DMA, 8 digital I/O

\$595

learn more:

voice 800-648-6589
fax 617-938-6553
web www.adac.com
email info@adac.com

ADAC

American Data Acquisition Corporation
70 Tower Office Park, Woburn, MA 01801 USA

Figure 3—The 403 supports the PowerPC user instruction-set architecture, including thirty-two 32-M general-purpose registers as well as several implementation-specific peripheral-device control and special-purpose registers to manage 403-specific features.

responsibility to manage the GPRs (including the stack pointer [rl by convention] and LR) appropriately across procedure calls. A leaf procedure (i.e., one that doesn't call another procedure) need not save the LR and may avoid this extra overhead.

The count register (CTR) is used by the conditional branch-on count instruction. It offers a robust combination of decrement count and branch on CTR==0 or CTR!=0 and/or condition code field (CR) bits.

A set of 403-specific device-control and special-purpose registers are provided for in- and out-of-core control and status. The device-control registers offer eight BIU memory-bank control registers, bus error status, DMA control, external interrupt enable/status, and an I/O configuration register.

The special-purpose registers use debug breakpoint compare, cacheability, memory write-protection bounds, programmable interval timer, processor version, time-base, and context switch save/restore. These registers are accessed using `mtspr/mfspr` and `mtdcr/mfdcr` (move to/from special-purpose/device-control register) instructions.

The machine state register (MSR) is a 32-bit register containing various machine state bits. It is read and modified using `mfmmsr/mtmsr` (move from/to MSR).

The MSR includes the problem state (PR) bit, external interrupt enable (EE) bit, machine check (ME) bit, and little/big Endian (LE) mode bits, among others. The PR bit determines supervisor or user mode.

When an exception or interrupt condition occurs, the instruction address register (IAR) and MSR are copied to save/restore registers SRRO and SRR1, respectively. The MSR PR bit is then changed to a supervisor state, and

the processor branches to the specific exception vector based off the exception vector prefix register (EVPR).

INSTRUCTION SET

Instructions are contained in a single 32-bit word with the high 6 bits specifying a primary opcode. The remaining bits are separated into various fields for several different instruction formats, including an extended opcode for many instructions.

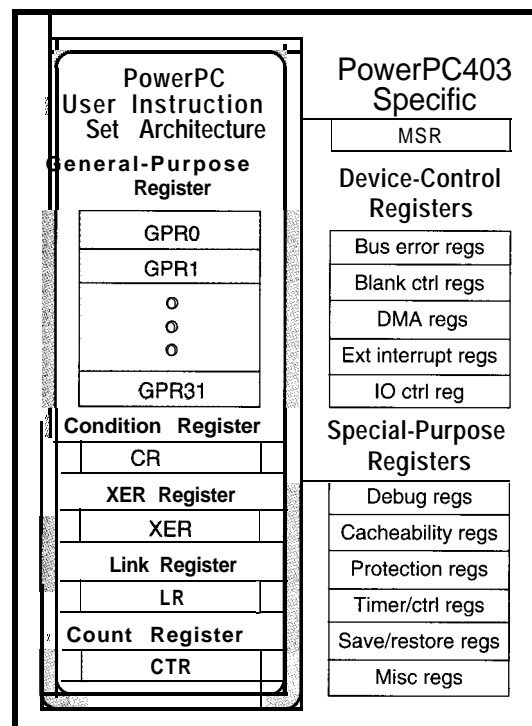
Most arithmetic, logical, and shift/rotate instructions have both register-register and register-immediate forms such as:

```
add rD, rA, rB
addi rD, rA, si mm
```

For the add immediate, if register rA is 0, then the sign-extended immediate value is placed into rD (i.e., a load immediate).

Several instruction forms employ the convention "use the contents of rA if rA != 0, otherwise use the value zero." Many instructions include a record bit, Rc. It allows the instruction to selectively alter the contents of condition register field 0 (CRO).

The compare instructions include register-register and register-immediate forms and offer selection of which



condition register fields to alter (CR0-7). This example compares the contents of GPR 5 to the value of 87 and sets the appropriate condition bit in CR3:

```
cmpi CR3,r5,87
```

The conditional branch instructions include five instruction fields—the S-bit BO field specifying the conditional branch type, the 5-bit BI field indicating which condition register bit to test, a 14-bit branch displacement field, and the AA [absolute address] and LK [procedure return link] fields.

The BO field offers combinations of decrement CTR and branch if CTR==O

or CTR!=O, branch if bit BI in the condition register is true or false, and branch always.

For example, in order to decrement CTR and branch and link [i.e., save the return address) to the subroutine at absolute address 0x100 if CR1 indicates less than or equal and the CTR!=O, use:

```
bclae 0,5,0x100
```

Various extended branch mnemonics provided by the assembler allow simplifications such as:

```
beq label1
```

which branches to relative address 1 a be 1 1 if the EQ bit is set in CRO.

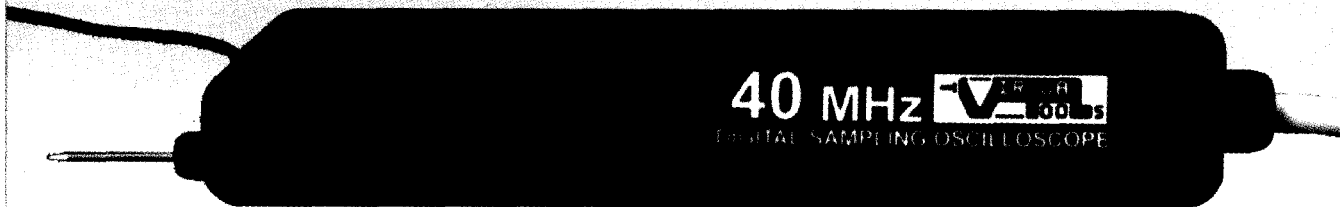
Most conditional-branch forms also support a branch-prediction bit that enables the programmer to indicate the branch's usual path. The processor prefetches instructions down the specified taken or not-taken path.

For example, to indicate that a conditional branch is usually taken, use:

```
bne+ label2
```

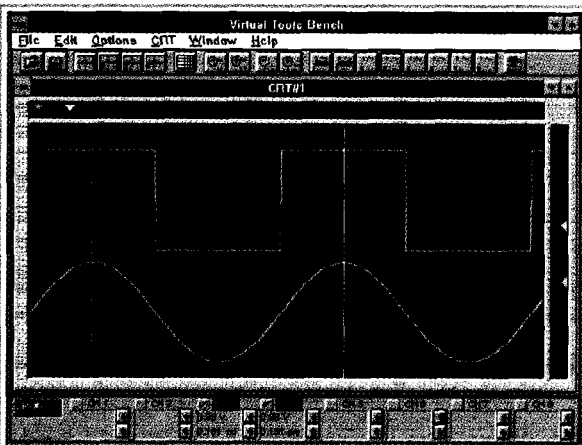
Load and store effective addressing includes both indexed and direct forms with or without update. Example formats are:

Parallel port connected and ready to go...



Our Digital Sampling Oscilloscopes have 20 or 40 MHz maximum sampling rates with 8-bit resolution. Both have 32 Kbytes of storage; 7 sampling depths; 24 sampling rates; 6 input voltage ranges; and multiple trigger options.

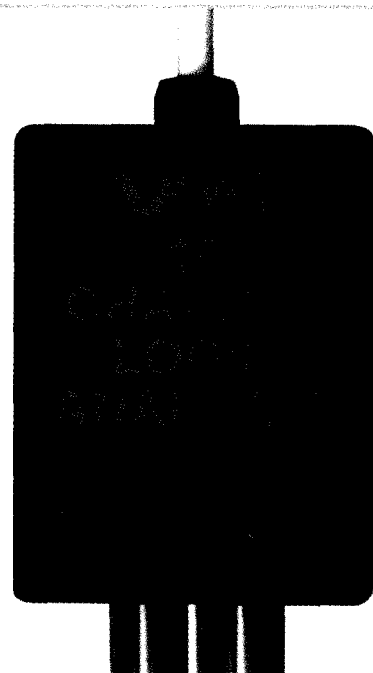
20 MHz \$ 199/\$249 40MHz



Our Virtual Tools Bench software detects connected devices and installs appropriate control surfaces. Features include 4 data display screens with rapid zooming and panning. Supports up to 8 devices with trigger interconnects and chaining.

FREE at website

Our Logic Analyzer has 16 channels with 3.3V/5V compatible logic inputs. The maximum sampling rate is 40 MHz with 6 internal clock rates and an external clock input with + or - going slope. The internal trigger setup allows bits to be low, high or disabled. An external logic level trigger is provided as well as the ability to trigger from our DSO. The internal storage is 32 Kbytes with 7 sampling depths and 3 trigger position options. Units can be chained for larger data widths.



\$199



VIRTUAL TOOLS, Inc.

(619) 940-0259 FAX (619) 940-1427



<http://www.virtualtools.com>

lwz rD,d(rA)
 stwx rD,rA,rB

The first instruction loads the word at the specified effective address into register rD, and the second stores the word in register rD to the specified effective address. Effective addresses are calculated as:

- displacement (d) form:
 if (rA == 0) b <= 0
 else b <= (rA)
 EA <= b + SignExtend(d)
- indexed form:
 if (rA == 0) b <= 0
 else b <= (rA)
 EA <= b + (rB)

For update forms, the calculated effective address EA is placed in register rA after the memory reference is complete (rA==0 is invalid for update forms).

Example load and store instructions are:

lwzx r3,r4,r5
 lwz r12,10(r30)
 stw r4,0x1000(0)
 stbu r6,4(r3)

The first instruction loads the word at address (r4+r5) into r3. The second loads the word at address (r30+10) into r12. The next instruction stores the contents of r4 to memory address 0x1000. And, the last instruction stores the least significant byte in r6 to address (4+r3) and then puts the value (4+r3) into register r3.

PROTOTYPE HARDWARE

The prototype schematic, shown in Figure 4, demonstrates a low-cost 403GA implementation. It isn't designed nor intended for production use.

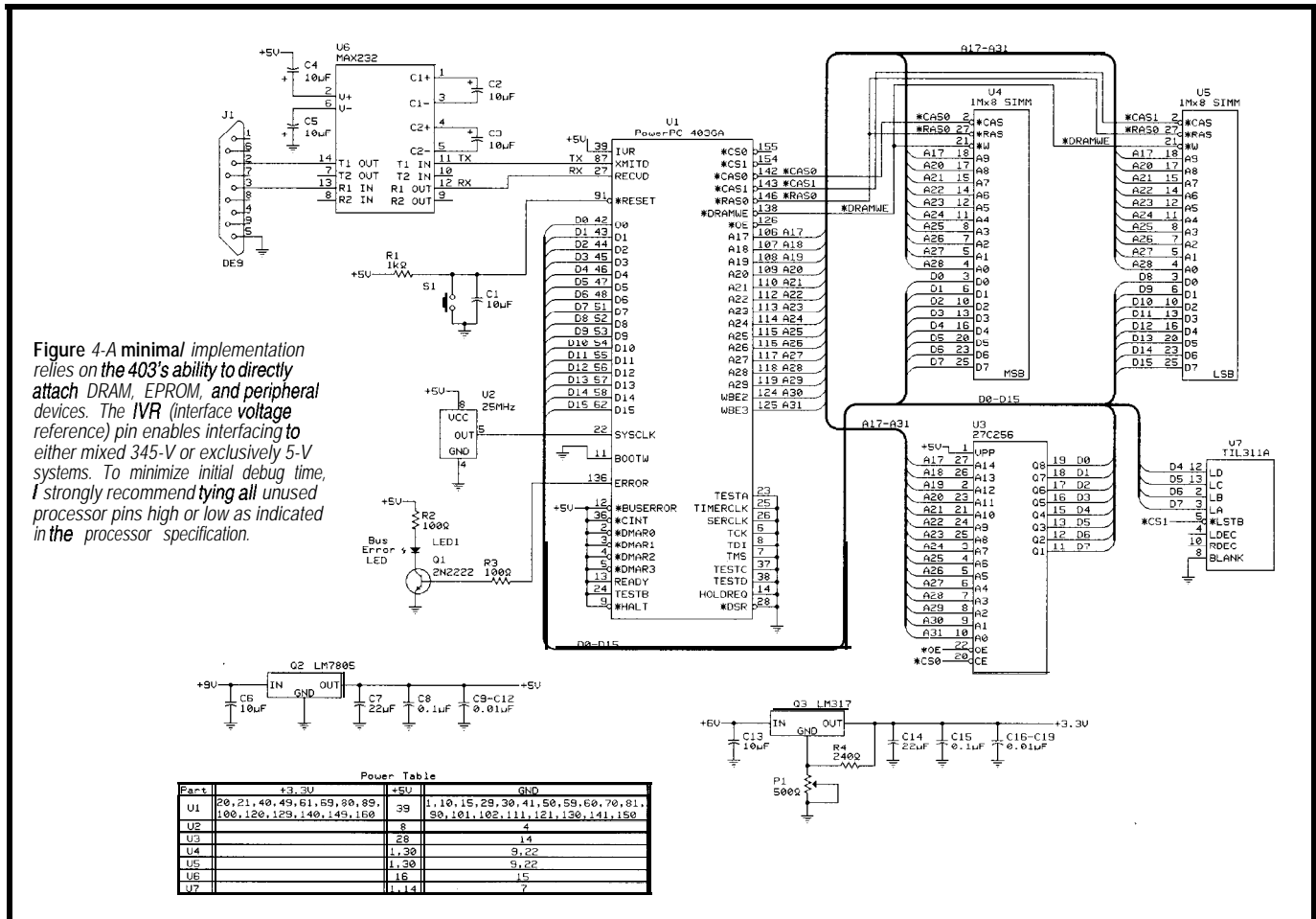
The prototype (see Photo 1) combines a byte-wide 32-KB (8K word) EPROM containing initialization code and a minidebugger with two 1-MB SIMM DRAMs, arranged as 1 M x 16 bits (5 12 K x 32-bit words), for program and data. It includes an RS-232 inter-

face and a simple single-digit hexadecimal display for initial debugging.

A minimal-cost system can also easily employ a single byte-wide SIMM by eliminating the second SIMM and reprogramming the DRAM bank register. Likewise, to maximize performance, a 32-bit ROM and/or DRAM configuration is a simple matter of bank-register programming (and a few extra bus wires).

The 403GA, available in a 160-pin plastic quad flat package (PQFP), is attached to the Aries QFP-to-PGA adapter and dropped into a surplus AMP 15 x 15 PGA ZIF socket. The cumulative adapter and (new) ZIF socket cost nearly approaches low-volume PC board NRE, but I chose the adapter/socket and wire-wrapping approach.

The 403 core operates at 3.3 V and may interface either a 3-V system exclusively or a mixed 3-5-V system (as programmed by IVR, pin 39). This implementation uses 5-V memory and I/O devices and thus requires IVR=5V.



I used an LM317 adjustable regulator to provide the 3.3-V core supply. All 14 VDD power pins (and 19 GND pins) must connect to this supply. All unused inputs should be tied inactive. In particular, the manufacturing test inputs, pins 23 and 24, must be tied low and high, respectively.

Since memory-device width is programmable, including the boot ROM, some method must initially indicate the boot ROM's width. This is achieved with the BootW input (pin 11). Tying this pin low selects an 8-bit boot ROM, whereas tying it high selects a 32-bit boot ROM. Tying it to RESET (pin 91) selects a 16-bit boot ROM.

The byte-wide boot/debug EPROM requires the two least-significant address bits (A30 and A31) to select the byte within the 32-bit instruction or data word. The LSBs are not normally required for 32-bit word references but are provided for byte and half-word accesses through signals WBE2 (pin 124) and WBE3 (pin 125).

BIU chip-select outputs *CS0-*CS7 are controlled by bank registers BR0-BR7, respectively. CS0 (pin 155) drives the EPROM chip enable, and the EPROM output enable pin is driven by the processor *OE signal (pin 126).

Contrary to what you expect, the high 8 (D0-D7) or high 16 (D0-D15) data bits interface byte or half-word wide devices. Data bits DO (MSB)-D7 (LSB) connect to EPROM data bits D7-D0, and data bits D0-D15 connect to the dual SIMM half-word DRAM.

DRAM multiplexed address bits connect to SIMM address bits AO-A9 in a less than straightforward manner. These processor address bits vary for different bus and DRAM memory sizes and are fully documented in the user manual.

For the 2-MB, 16-bit DRAM configuration implemented here, processor address bits A28-A20 and A17 connect to SIMM address lines AO-A9. For a 16-bit DRAM configuration, processor CS0 (pin 142) drives MSB SIMM0 CAS, and processor *CAS1 (pin 143) drives LSB SIMM1 CAS.

Processor *RAS0 (pin 146, the DRAM chip-select pin for memory

- 1 Initialize bank registers
- 2 Invalidate the icache
- 3 Enable EPROM cacheability
- 4 Test DRAM before data caching is enabled
- 5 Initialize exception (interrupt) vectors
- 6 Initialize serial port
- 7 Invalidate the dcache
- 8 Enable DRAM cacheability
- 9 Set exception vector base address (EVPR)
- 10 Enable serial-port interrupts
- 11 Enable write-protection bounds
- 12 Initialize MSR
- 13 Set SRRO and SRR1 to simulate debug exception
- 14 Jump to debugger

Table 1—Initialization may be somewhat less obvious, given the dual on-chip L1 caches, memory-bank register programming, and cacheability options. I used this sequence for my prototype implementation.

bank 7) drives *RAS on both SIMM DRAMs. When using DRAM devices, processor outputs *RAS3-*RAS0 are controlled by bank registers BR4-BR7, respectively (note the reverse numbering). The SIMM write signal is driven by processor *DRAMWE (pin 138).

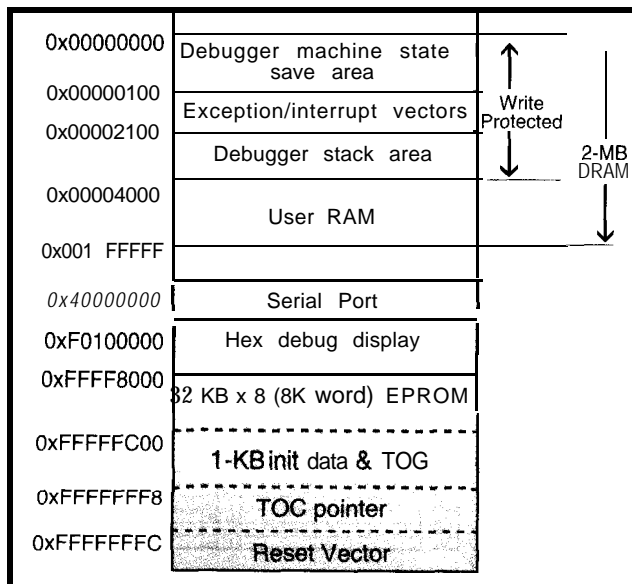
The single hexadecimal display (TIL-311A) is a poor man's alternative to an in-circuit emulator or debugger for initial system bringup. It's relatively simple to debug and verify itself and offers the capability of 16-point sequential bring-up status while debugging initialization code.

Finally, the 3-wire 9-pin D-shell RS-232 interface uses a single-supply MAX232 driver/receiver for processor serial-port signals XMITD (pin 87) and RECVD (pin 27). Note that processor input 'DSR (pin 28) is tied active.

INITIALIZATION SOFTWARE

Initialization software is somewhat complicated by the on-chip caches

Figure 5—For this implementation, the debugger shares low RAM with the interrupt/exception vectors, both of which are write protected from user-mode programs using the 403 memory-protection capabilities. User-mode programs are allocated the remaining 2M-0x4000 bytes for instruction and data memory.



and BIU direct-attach memory devices. Care must be taken to invalidate the caches before use and to properly initialize the memory-bank registers before attempting to access memory outside the boot ROM region. Table 1 shows the prototype initialization sequence.

First, I program the memory-bank registers to fully enable EPROM, DRAM, and I/O accesses and enable the device chip selects. Listing 1 shows the assembly code that sets up the bank registers.

The eight bank registers contain various fields for base address selection, bank size and usage, memory wait states, ready enable, and other static or DRAM specific settings.

For this implementation, bank register 0 (BRO) is configured to select the EPROM (*CS0), BR1 is configured for the hexadecimal display (CS1), and BR7 is set to control the 2-MB DRAM SIMMs, including refresh rate and cycle times, attached to the RAS0 memory control pin. Figure 5 shows the resulting memory map.

Once the bank registers are configured, I invalidate the unknown bootup contents of the instruction cache and enable EPROM icache cacheability to boost initialization performance for the remainder of the initialization sequence (see Listing 2).

At this point, and explicitly before enabling DRAM data cacheability, I verify DRAM integrity with several

memory test patterns. Once the memory is satisfactorily tested, I initialize the exception vector prefix register (EVPR) and exception/interrupt vectors and configure the serial port control registers. The serial-port initialization is shown in Listing 3.

The memory write-protection bounds registers are initialized to catch illegal user-mode program writes to low-vector memory or past the end of DRAM. The user mode MSR is initialized and loaded, along with the IAR (PC) of a dummy idle loop, into the save/restore registers SRRI/SRRO to simulate a debug exception. Control then transfers from the assembler initialization to a C-level debug routine.

CONTEXT SYNCHRONIZATION

Since the 403 is a pipelined, super-scalar processor, exercise due caution when attempting certain operations with CPU or memory context or sequence interdependencies.

For example, changing a memory protection-bounds register for a memory area before executing a store instruction to that same area usually causes a protection-bounds violation.

Ensure the protection-bounds update is complete before attempting a memory write. Several CPU and memory context synchronizing instructions establish a known context including ISYNC and EIEIO (enforce in order execution of I/O).

PERFORMANCE

The primary disadvantage of the reduced-cost, half-word DRAM configuration implemented here is that it requires two memory accesses per data or instruction word reference.

Using a debugger developed for this prototype, I downloaded and ran several performance tests for a 25-MHz implementation. Memory read access time was measured at 808.4 ns per word (or about a 20-cycle delay) for the given DRAM configuration and bank-register programming. The test read every other fourth word from a 128K-word array, thereby always missing the data cache.

This penalty can be halved via a 32-bit-wide DRAM configuration. Sequential memory reads, missing the dcache on one out of four loads, mea-

Listing 1—The 403 bank registers configure the BIU to the characteristics of each specific direct-attached memory device, establishing memory size and type, set-up and hold times, as well as DRAM interface and refresh timings.

```
.globl .entry
.entry:
# Set bank register 0 (CS0, EPROM) = 0xff180700
# BAS (base address select) = 0xff, BS (bank size) = 0 (1 MB),
# BU (bank usage) = 3 (R/W), BW (bank width) = 0 (8 bit),
# SLF (sequential line fill) = 0 (target word first),
# BME (burst mode enable) = 0 (burst disabled),
# RE (ready enable) = 0 (disabled), TWT (transfer wait states) = 7,
# CSN = 0, OBN = 0, WBN = 0, WBF = 0, TH (transfer hold cycles) = 0
    addi r3, r0, 0x0700
    addis r3, r3, 0xff18
    mtdcr BR0, r3

# Set bank register 1 (CS1, hex display) = 0x01100200
    addi r3, r0, 0x0200
    addis r3, r3, 0x0110
    mtdcr BR1, r3

    addi r31, r0, 0x0000
    addis r31, r31, 0xf010    # r31 points to hex display

    addi r30, r0, 0xf
    stb r30, 0(r31)    # Write to debug display

# Set bank register 7 (RAS0, DRAM) = 0x0038A264
# BAS (base address) = 0x00, BS = 2 MB, BU = RW
# BW (bus width) = 16 bit, RCT (RAS active to CAS active) = 1
# FAC (first access timing) = 1, BAC = 0, PCC (precharge cycles) = 1
# (2.5 sysclks), RAR = 1, RR (refresh rate) = 5 (7.68 μs at 25 MHz)
    addi r3, r0, -23964    # 0xa264
    addis r3, r3, 0x0038+1    # +1 because low 16 is negative
    mtdcr BR7, r3
```

sured at 18.9 MBps or about five cycles per load on average.

This 5-instruction icache bound loop verifies single-cycle instruction execution for given instructions with load and store always hitting in the cache:

```
loop:  1 r4,0(r3)
      a r5,r4,r4
      rli nm r6,r5,4,0xFFFF
      st r6,0(r3)
      bdnz loop
```

Average interrupt latency was measured for both best- and typical-case scenarios. The best response time of 17.5 cycles (700 ns) was measured for interrupt-handler instruction fetches and data references hitting the caches.

The typical-case latency measured at 125 cycles (5 μs) when the interrupt handler always missed in both caches. Both cases include six instructions of extra overhead to generate an external timing strobe.

Listing 2—The instruction cache contents are invalidated (using the icc instruction in a loop for each of the 64 IC cache congruence classes) before if is enabled for use.

```
# Invalidate the cache
    addi r3, r0, 64
    mtspr CTR, r3
    xor r3, r3, r3

ic_inv:
    iccci 0, r3
    ai r3, r3, 16 # next congruence class
    bdnz ic_inv

# Enable EPROM icache cacheability
    addi r3, r0, 0x0001
    mtspr ICCR, r3    # Make 0xf8000000-0xffffffff icacheable
```

Listing 3-1 used this serial-port initialization for the three-wire interface (Rx, Tx, and Gnd).

```
# Serial-port init
.set SERCLK,25000000
.set BAUD, 9600
.set BRD, (SERCLK/(16*BAUD))

init_serial:
    addi r4,r0,0          # Base address to on-chip peripheral (OPB)
    addis r4,r4,0x4000 # Serial port (0x40000000) and 0 to low byte


    stb r4,SPBRDH(r4) ii Baud rate div high = 0
    stb r4,SPBRDL(r4) # Baud rate div low = 0
    addi r3,r0,0x78
    stb r3,SPLS(r4) # Clear serial port line status bits (0x78)
    addi r3,r0,0xff
    stb r3,SPHS(r4) ii Clear serial port handshake status)
    stb r4,SPCTL(r4) # Zero serial port control register
    stb r4,SPRC(r4) # Zero serial port Rx command reg
    stb r4,SPTC(r4) # Zero serial port Tx command reg

    addi r3,r0,BRD/256
    stb r3,SPBRDH(r4)
    addi r3,r0,BRD&0xff
    stb r3,SPBRDL(r4) # Set serial port baud rate = 9600
    addi r3,r0,0x38
    stb r3,SPCTL(r4) # DTR = 1, RTS = 1, 8N1
    addi r3,r0,0xa0
    stb r3,SPRC(r4) # Enable receiver and RBR interrupts
    addi r3,r0,0x80
    stb r3,SPTC(r4) # Enable transmitter
```

If you plan to call a C routine from the interrupt handler or need to save the entire machine state, interrupt latency increases significantly (especially given a 20-cycle cache miss penalty). To maximize performance, tune your code for both loads and stores and instruction fetches to efficiently use the caches and avoid cache-miss penalties.

SIMPLY EMBEDDED SYSTEM

The 403GA provides a high-performance, low-power RISC processor with integrated on-chip caches and support facilities. Its direct memory attach bus unit makes it well-suited for embedded applications.

My prototype implementation demonstrates the simplicity and low parts count possible for a minimum embedded-controller system. IBM and select independent vendors provide a full line of software and hardware development tools, including debuggers, RTOs, and C/C++ compilers, supporting the PowerPC 400 series of embedded controllers. 

Many thanks to Kaivalya Dixit, Jeff Garelick, and IBM for their support

and encouragement of this "after-hours" project.

Randy Heisch is a senior engineer/programmer in the IBM RS/6000 Processor and System Performance Group. You may reach him at heisch@austin.ibm.com.

SOURCES

PowerPC 403GA
IBM
1000 NW 51 St.
Boca Raton, FL 33432
(407) 443-2000
Fax: (407) 443-4533
www.chips.ibm.com/products/embedded/index.html

Aries QFP-to-PGA adapter
Digi-Key Corp.
701 Brooks Ave. S
Thief Falls, MN 56701-0677
(218) 681-6674
Fax: (218) 681-3380

IRS

404 Very Useful
405 Moderately Useful
406 Not Useful

RELAY INTERFACE

PROVIDES SOFTWARE CONTROL OF RELAYS

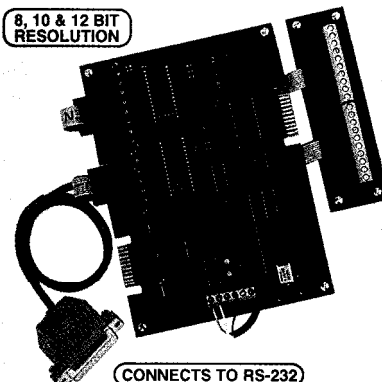


CONNECTS TO RS-232

AR-16 RELAY INTERFACE (16 channel).....\$89.95
Two 8 channel (TTL level) outputs are provided for connection to relay cards or other devices (expandable to 128 relays using EX-16 expansion cards). A variety of relays cards and relays are stocked. Call for more info.
AR-2 RELAY INTERFACE (2 relays, 10 amp)....\$44.95
RD-8 REED RELAY CARD (8 relays, 10 VA)....\$49.95
AH-8 RELAY CARD (IO amp SPDT, 277 VAC)....\$69.95

ANALOG TO DIGITAL

8, 10 & 12 BIT RESOLUTION



CONNECTS TO RS-232

ADC-16 A/D CONVERTER* (16 channel/8 bit)....\$99.95
ADC-8G A/D CONVERTER* (8 channel/IO bit)....\$124.90
Input voltage, amperage, pressure, energy usage, light, joysticks and a wide variety of other types of analog signals. RS-422/RS-485 available (lengths to 4,000'). Call for info on other A/D configurations and 12 bit converters (terminal block and cable sold separately). Includes Data Acquisition software for Windows 95 or 3.1
ADC-BE TEMPERATURE INTERFACE* (8 ch)....\$139.95
Includes tan. block & 8 temp. sensors (-40° to 146° F).
STA-8 DIGITAL INTERFACE* (8 channel).....\$%
Input on/off status of relays, switches, HVAC equipment, security devices, keypads, and other devices.
FS-4 PORT SELECTOR (4 channels **RS-422**)....\$79.95
Converts an RS-232 port into 4 selectable RS-422 ports.
CO-422 (RS-232 to RS-422 converter).....\$39.95

***EXPANDABLE**...expand your interface to control and monitor up to 512 relays; up to 576 digital inputs; up to 128 analog inputs or up to 126 temperature inputs using Ma PS-4, EX-18, ST-32 & AD-16 expansion cards.

***FULL TECHNICAL SUPPORT**...provided over the telephone by our staff. Technical reference & disk including test software & programming examples in QuickBasic, GW Basic, Visual Basic, Visual C++, Turbo C, Assembly and others are provided.

• **HIGH RELIABILITY**...engineered for continuous 24 hour industrial applications with 10 years of proven performance in the energy management field.
• **CONNECTS TO RS-232, RS-422 or RS-485**...use with IBM and compatibles, Mac and most computers. All standard baud rates and protocols (50 to 19,200 baud).

FREE INFORMATION PACKET...use our 800 number. Fax or E-mail to order, or visit our Internet on-line catalog.
URL: <http://www.eeci.com>
Technical Support (614) 464-4470

24 HOUR ORDER LINE (800) 842-7714
Visa-Mastercard-American Express-COD

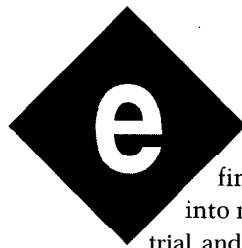
Internet E-mail: eeci1@ibm.net
International & Domestic FAX: (814) 464-9666
Use for information, technical support & orders.
ELECTRONIC ENERGY CONTROL, INC.
380 South Fih Street, Suite 604
Columbus, Ohio 43215.5491

FEATURE ARTICLE

Bill Houghton

Add an I/O Coprocessor to Your Embedded PC

When can't a 32-bit PC match the speed of an 8-bit PWM chip? According to Bill, when there's lots of I/O. Then, the PC's overhead and non-real-time response become problems. 8xC576 to the rescue!



Embedded PCs are finding their way into nondesktop industrial and scientific applications since there's no doubt the PC platform offers the most off-the-shelf hardware and software options.

However, when it comes to real-time data acquisition and control, there are some skeletons in the PC closet. All too often, it's only after you start rooting around that it becomes clear that "real time" has a different meaning on PCs than dedicated MPUs.

Depending on the exact collection of OS, BIOS, RTOS, DLLs, drivers, and compiler, it can take a long time for even the simplest I/O event to wend its way between CPU and the real world.

Even the most streamlined PC setups have a lot of overhead, imposing dozens if not hundreds of microseconds of overhead. Response time is poor and subject to an annoying amount of jitter (i.e., lack of determinism) due to everything from cache effects to rude-neighbor, cycle-hogging drivers.

If anything, the trend is getting worse. For instance, Windows NT is admired for achieving a degree of robustness required by embedded apps. But, it so insulates the hardware that it's a major exercise to toggle a bit.

One recent series ("Embedded PCs Go Industrial," *INK* 75 and 77) mea-

sured the average latency associated with NT's built-in DPC (Deferred Procedure Call) scheduling mechanism as 27 μ s for a 150-MHz Pentium setup. The worst-case response measured at greater than 1 ms (or in the words of the author, "essentially unbounded").

While a few dozen microseconds may not seem like much, it adds up quickly. To paraphrase current wisdom, "A few microseconds here, a few microseconds there, pretty soon you're talking *real* time."

Consider the simple task of generating PWM output. Typically, a periodic interrupt is handled by incrementing a counter, comparing it to a value corresponding with the desired duty cycle, and toggling the output bit for matches.

A little math shows a simple 8-bit (duty cycle = $\frac{0}{255} - \frac{255}{255}$) PWM is lucky to run at 100 Hz (i.e., $256 \times 27 \mu\text{s} = 6.9\text{-ms}$ cycle time) by the time a PC is through with it.

That a 32-bit PC can't match the I/O handling of an 8-bit chip foreshadows the solution. Use an MCU as an I/O processor (IOP). The Philips 8xC576 proves to be uniquely suited to the task.

'576 TO THE RESCUE

Two issues come to the forefront when choosing an IOP. Obviously, an IOP needs lots of I/O for processing.

Though packaged in the 805 1's familiar 40-pin DIP or 44-pin PLCC and QFP package, the '576 packs a lot more stuff onboard than the original. It's double the memory (8-KB[EP]ROM, 256-byte RAM) and has quite a few high-value I/O functions added as you see in Figure 1.

Along with a third 16-bit timer/counter (the original '51 has two), the '576 adds a complete five-channel Programmable Counter Array (PCA). As shown in Figure 2, it comprises a 16-bit timebase (driven at $\frac{1}{4}$ or $\frac{1}{2}$ the oscillator frequency from Timer 0 overflow or externally) feeding five compare/capture modules, each with its own pin (CEX_n).

Each module can be independently configured in compare or capture mode. In capture mode, the pin becomes an input which, on rising or falling edge (programmable), latches the main timer's value into the associated mod-

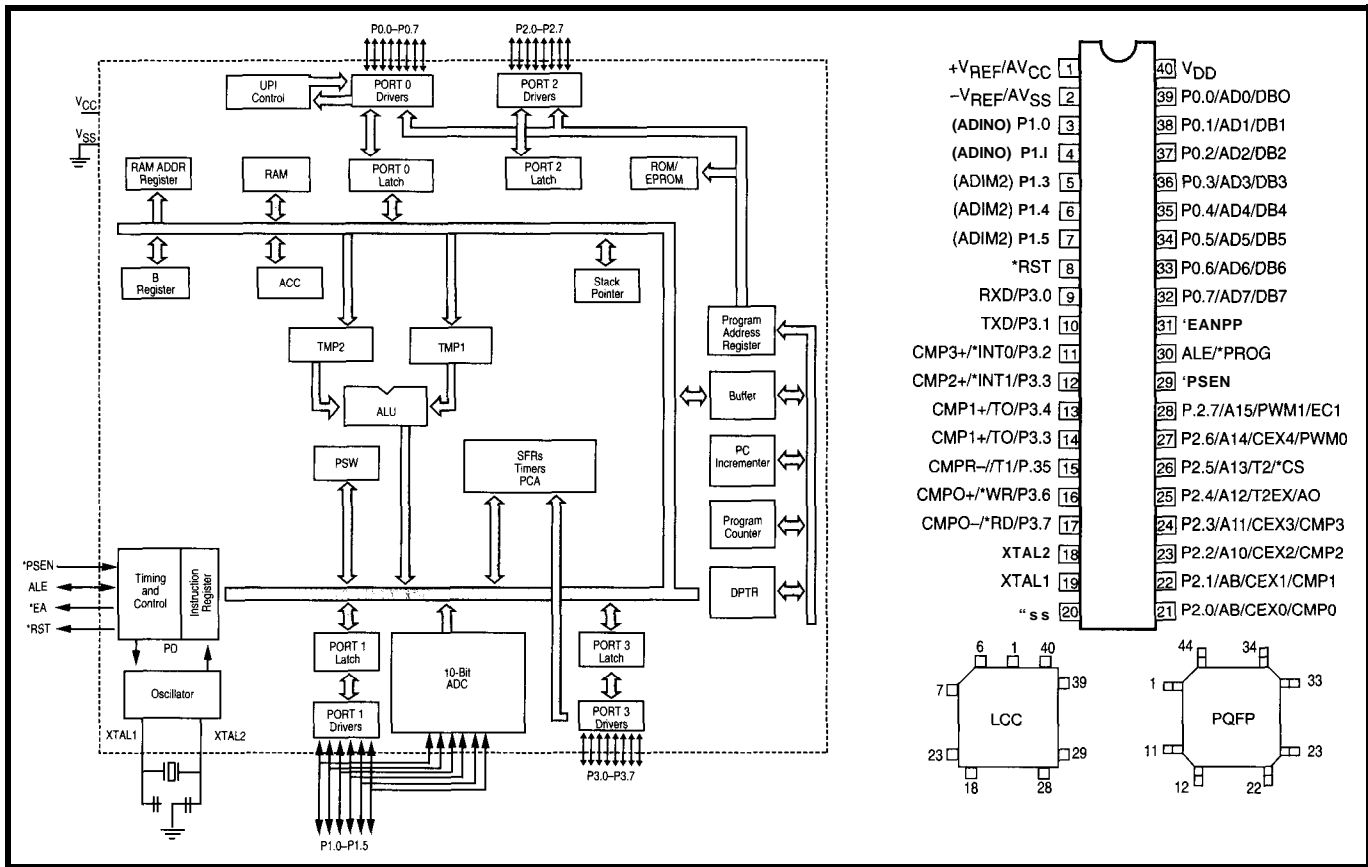


Figure 1—The 8x756 starts with an enhanced 8051 core-CPU and adds significant I/O functions, including 5-channel x 16-bit PCA (Programmable Counter Array), two dedicated 8-bit PWMs, 6-channel x 10-bit ADC, four analog comparators, and a UPI (Universal Peripheral Interface).

capture/compare reg (both 16-bits). In compare mode, the pin is an output that toggles when the main timer reaches the value programmed in a module compare/capture reg.

The PCA is a powerful and speedy (multimegahertz) unit, capable of many tasks including interval timing, periodic interrupt, PWM generation, and watchdog timing. Nevertheless, it's supplemented with a dedicated watchdog timer and dual 8-bit PWMs. Unlike a PC, which might struggle to hit 100 Hz with a software approach, the '576 PWMs run at up to 20+ kHz.

The '576 is equally well-armed on the analog front with a 6-channel 10-bit ADC supplemented with four analog comparators. The ADC is capable of relatively speedy conversion (48 instruction cycles, or 48 μs at 12 MHz), even though bandwidth is usually limited to a few kilohertz by slew rate and sample-hold timing constraints.

Though the various error specs (e.g., linearity, full scale, offset) range 1-3 LSB, slow-changing signals can be oversampled and averaged or the error thrown away along with the two LSBs by settling for 8-bit resolution.

The '51's UART gets enhancements as well, including framing error detection and automatic address recognition for the 9-bit data mode. This mode, offered on many vendors' CPUs, allows simple multidrop networking by relying on the ninth bit to signify that the remaining eight bits should be interpreted as a node address.

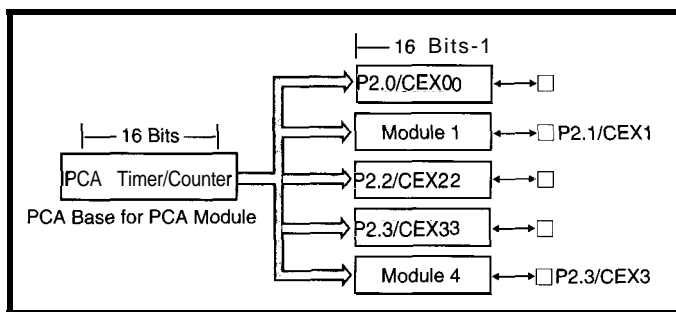


Figure 2—The PCA features high speed (up to 4 MHz) and resolution (16 bits). Each of the five modules can be independently programmed to operate in a number of different modes, some of which (e.g., PWM and edge capture) use an I/O line (CEXC-4).

However, nodes must check all addresses for a software match. The '576 includes an address register with automatic comparison, so it only interrupts the CPU when a match occurs.

The '576 incorporates a number of niceties addressing real-world design concerns. The reset circuitry (active low, as most chips are these days) incorporates the watchdog timer along with low-voltage and oscillator-failure detection (allowed range: 6-16 MHz).

There are also modes to reduce EMI, including cutting the clock drive (reduces max clock to 12 MHz) and

turning off ALE (i.e., single-chip mode). Low active power (a little over 1 mA/MHz typical at 5 V) is complemented with Idle (0.5 mA/MHz) and Power Down (5 μA) modes.

With plenty of I/O processing, all you need is a mechanism for communication between the PC and IOP. It's the final '576 add-on—the Universal Periph-

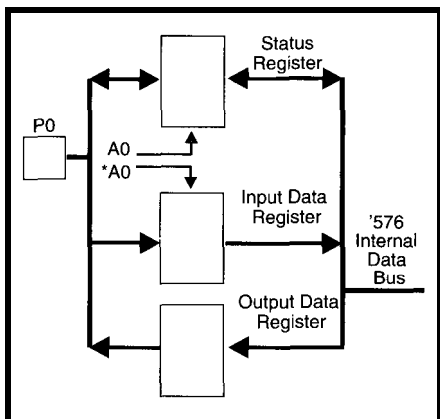


Figure 3—The UPI interface consists of one data register in each direction plus a status register.

eral Interface (UPI) port—that's makes the PC connection.

DEJÀ VU

It's ironic that many don't have a clue what a UPI is when almost everyone uses one on a daily basis. Everyone, that is, who uses a PC with a UPI-based micro on the receiving end of the keyboard cable.

Indeed, this is a classic example of an MCU offloading the PC CPU of mundane, time-consuming chores. No doubt a Pentium, even burdened with convoluted software, could manage a keyboard's clocked serial interface.

But, there's also no question that it's not a bright idea. The fact that an MCU with UPI port made sense as an IOP way back when reinforces the idea of using one today for all sorts of other I/O intensive and real-time tasks.

Fortunately, the UPI is quite simple to understand. Essentially, it's a pair of data registers [one in each direction] and a status register (see Figure 3). Together, they act as a communications mailbox between the '576 and PC.

It's easiest to understand the way it works by detailing port access from each processor's point of view.

The PC side accesses an 8-bit bidirectional data bus (Port 0 of the '576) and generates four control lines—CS, AO, RD, and WR. For the most part, operation is straightforward, with CS enabling the UP1 port and RD/*WR strobing the data to/from the host.

Note that port access is quite fast as shown in Figure 4. It is well able to handle the historically relaxed (e.g., 8 MHz) PC I/O expansion-bus timing.

For PC reads of the UPI, A0 works as expected, selecting either the data (AO=0) or status (AO=1) registers.

Writes are a little trickier. While a write with AO=0 accesses the data register, one with AO=1 does not directly write the status register.

Instead, it writes the data register (i.e., same as when AO=0) and sets the AF (Address Flag) bit in the status register. In other words, writes from the host always go to the data register, with AF in the status register as the only reflection of whether A0 was 0 or 1.

Although it sounds odd, this scheme makes sense. The decision not to allow the PC to write the status register directly is made purposefully to facilitate a robust protocol that minimizes confusion about who's doing what.

Simultaneously, A0 (accessible via the AF bit of the status register) can differentiate between subsystem commands and data at the application level.

From the '576's point of view, the UPI data bus is accessed as a byte (or bit) addressable SFR (Special Function Register address 80H) in the usual way. Meanwhile, the UP1 status register is also directly accessible via another SFR address (86H).

The final piece of the puzzle is the function of the various bits in the UP1 status register (see Figure 5). Besides AF [the already described A0 flag], there's UE (UPI Enable), IBF (Input Buffer Full), and OBE/OBF (Output Buffer Empty/Output Buffer Full).

Unlike the PC side, which can't write the status register (except the AF bit via AO), the '576 has more write access. The upper four bits (ST4–7) are always writable by the '576 as applica-

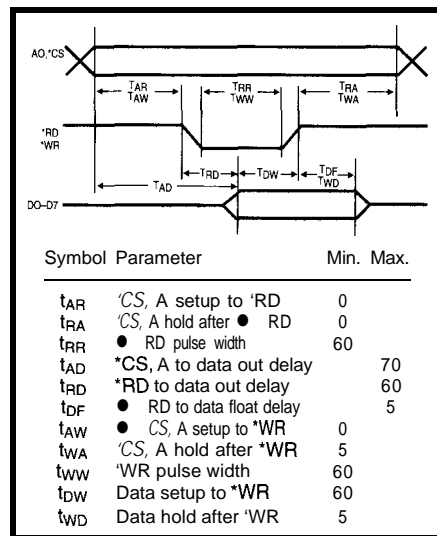


Figure 4—The UPI interface is quite conventional, relying on AO, CS, RD, and WR lines. Access time is fast enough to interface easily with most CPUs, and it ensures that the interface isn't a bottleneck. All times are in nanoseconds.

tion-specific flags. For example, the UP1 handling PC keyboards defines ST4–7 as parity and other error bits.

The UE bit is also always writable by the '576 to enable or disable the UP1 interface. However, when the UP1 is enabled, the key handshake flags AO, IBF, and OBE/OBF are only affected by hardware transactions.

Thus, during normal communications, the '576 doesn't write these flags since the hardware does it automatically. And, in the interest of robustness, it can't write them. Of course, during initialization and if things get really goofed up, the '576 can disable the UPI via UE to write protected flags.

Finally, IBF and OBE/OBF play the critical role of handshaking transactions across the UP1 data port. To keep things straight, remember that input and output are from the '576's point of view.

UCS.7	ST7	User-defined	status	bit	7	6	5	4	3	2	1	0
UCS.6	ST6	User-defined	status	bit	ST7	ST6	ST5	ST4	UE	AF	IBF	OBE
UCS.5	ST5	User-defined	status	bit								
UCS.4	ST4	User-defined	status	bit								
UCS.3	UE	UPI enable bit. 0 = Disabled, 1 = Enabled										
UCS.2	AF	Address Flag—contains the state of the A0 (Address) pin on the last write cycle. 0 = Write cycle with A0 cleared, 1 = Write cycle with A0 set										
UCS.1	IBF	Input Buffer Full Flag—set by hardware on the rising edge of a write command to the Input Data Register. Cleared by hardware on the completion of a read cycle of the Input Data Register by the '576.										
UCS.0	OBE	Output Buffer Empty Flag—cleared by hardware on the completion of a write cycle to the Output Data Register by the '576. Set by hardware on the rising edge of the read command of the Output Data Register by the host.										

*Note: This flag is OBE when read by the MCU but is inverted or OBF (Output Buffer Full) when read by an external host.

Figure 5—The UPI status register contains the critical flags enabling interprocessor communication protocols

Listing I--This code demonstrates the use of the UPI on the '576 written in Franklin C.

```
#include <reg51.h>
sfr UCS = 0x86; sbit UE = UCS^3; sbit AF = UCS^2;
sbit IBF = UCS^1; sbit OBF = UCS^0;
sfr ADC = 0xB1; sbit ADF = ADC^7; sbit ADCE = ADC^6;
sbit AD8M = ADC^5; sbit AMOD1 = ADC^4; sbit AMOD0 = ADC^3;
sbit ASCA2 = ADC^2; sbit ASCA1 = ADC^1; sbit ASCA0 = ADC^0;
sfr ADC0H = 0xAA; sfr ADC1H = 0xAB; sfr ADC2H = 0xAC;
sfr ADC3H = 0xAD; sfr ADC4H = 0xAE; sfr ADC5H = 0xAF;
sfr PWMO = 0xBE; sfr PWM1 = 0xBF; sfr PWMP = 0xBD;
sfr PWCON = 0xBC;

UC get_analog_channel(UC channel) {
    AMOD0 = 0; // Mode 0
    AMOD1 = 0;
    AD8M = 1; // 8-bit mode
    ASCAE = channel^2;
    ASCA1 = channel^1;
    ASCA0 = channel^0; ADCE = 1; // start conversion

    /* poll ADF flag. Loop breaks when ADF = 1 */
    while(!ADF); // wait for conversion to complete
    switch (channel) { /* return ADC value by selected channel */
        case 0: return(ADC0H); break;
        case 1: return(ADC1H); break;
        case 2: return(ADC2H); break;
        case 3: return(ADC3H); break;
        case 4: return(ADC4H); break;
        case 5: return(ADC5H); break;
        default: return(0xFF); break;
    }
}

UC get_digital_byte (UC dummy){
    UC byte; // declare temporary variable
    byte = P3 & 0x3f; // read P3 and strip the 2 M.S. bits
    // read P2, strip 6-ms bits, shift left 6 bits, OR with temporary var
    byte |= ((P2 & 0x03) << 6);
    return (byte); } // return result

UC put_digital_byte ( UC byte ){ // clear current port value
    P3 &= 0xc0; // AND parm with mask, OR with cleared port value
    P3 |= (byte && 0x3f);
    P2 &= 0x3f;
    P2 |= ((byte && 0xc0) >> 6);
    return(byte);

UC increase_pwm UC channel ){
    if (channel == 0) // test channel parameter
        PWMO++; // increase PWMO
    else
        PWM1++; // increase PWM1
    return(channel)

void main 0
{
    UC command; // declare temp vars for cmd and info bytes
    UC info;
    PWMP = 0x80; // initialize PWM controller
    PWCON = 0xFF;
    while(1) { // wait for PC cmd re. when IBF flag is set
        while (~IBF); // wait for the command
        command = P0; // read command byte. Clears IBF automatically
        while (~IBF); // wait for next byte from PC
        info = P0; // wait for information
        P0 = command; // echo the command
        while (!OBF); // wait for PC to read it
        switch (command) { // call approp funct, pass to parm
            case 1: P0 = get_analog_channel(info); break;
            case 2: P0 = get_digital_byte(info); break;
            case 3: P0 = put_digital_byte(info); break;
        }
    }
}
(continued)
```

99 MHz 8051 SBC!

You've heard about the Dallas 80C320, and 8051 compatible CPU. It can really get the job done fast, but you *need to get started right now*. The 320SBC features the '320 on a board that is ready to go -- NOW.

HTE's 320SBC features:

- High speed 8051 Instruction set, executes instructions up to 3X as fast as a standard 8051 at the same crystal speed.
- Cost effective: \$179 for the -50 development version, \$149 for the -10 OEM version in single qty.
- Two serial ports: RS-232 & 5V
- Optional 3rd RS-232 serial port can be used to free both the '320 serial ports for your application.
- On board monitor eliminates the need for an ICE in most cases.
- Development tools for C or ASM
- Production ready design - Now!
- Special configurations available that will reduce cost and meet your specific needs by including only the features you need.
- Compact size fits anywhere!



HiTech Equipment Corporation
 ,619) 566-1892 • Fax: (619) 530-1458
 nfo@hte.com http://www.hte.com

A PC write to the UP1 port (i.e., ● CS and * WR asserted) sets IBF=1 (and AF equals the state of A0 during the write). A subsequent UP1 port read by the '576 clears IBF but doesn't affect AF.

OBE/OBF is the only bit whose definition isn't straightforward. The dual naming convention reflects the fact that the sense of this bit is inverted between the '576 and host sides.

From the '576 point of view, OBE/OBF=1 means the output buffer is empty, so the '576 can write the next byte. From the host side, OBE/OBF=1 means the output buffer is full, so the host should read the byte.

For coding purposes, you may want to refer to OBE in the '576 software and OBF in the host code. In this case, a 1 in the bit is interpreted as a call for action (i.e., host should read or '576 can write). Some may be more comfortable just choosing one name for both host and '576 access and inverting the bit's polarity in software on one side accordingly.

Beyond these basics, little protocol is cast in stone. The A0/AF feature need not be used, and if it is, the interpretation of A0 is arbitrary. For example, instead of differentiating commands and data, the AF flag can simply transfer a ninth data bit or act as an auxiliary status bit like ST4-7.

Notice that the basic UP1 protocol is quite passive. That is, the PC and '576 aren't forced to rely on possibly disruptive means (e.g., interrupts, DMA) to communicate.

The '576 is the master in the sense that it can enable or disable the UP1 and directly set the flags. But, the UP1 port can't speak unless spoken to by the PC. Similarly, there's nothing the PC can do to the '576 except send data which the '576 is under no obligation to process.

Going beyond cooperative polling-type arrangements is completely up to the designer. On the '576 side, either or both of the IBF and OBE/OBF bits can be programmed to generate an internal interrupt (whose priority, high or low, is individually programmable as well). To get the PC's attention, just use a '576 port bit to output an interrupt request to the PC.

Listing 1-continued

```

case 4: P0 = increase_pwm(info); break;
default: P0 = 0xFF; break;
} // wait for PC to read information
while (!OBF);

```

UPI IN ACTION

A sample application shows how the UP1 facilitates the design of a PC plug-in IOP. The design in Figure 6 uses the CPU, some address decode logic, and little else to implement an ISA plug-in.

From the PC point of view, the UP1 is I/O addressed at ports 300H (data) and 301H (status), which the ISA spec allocates for prototype cards, so the addresses aren't likely already used. Should different addresses be required, simply choose another output from the decode logic.

The first decoder's outputs (U2) address the board between 300H and

3EOH on 20H boundaries (i.e., 300H, 320H.. .3EOH). Note that U2 is gated with AEN to prevent unintended access during DMA cycles.

Though the UP1 interface presents no timing problems for leisurely ISA I/O cycles, the '576 can only deliver 15 mA of the official 24-mA spec. As a practical matter, the drive is sufficient considering the historical progression of modern PC designs with the trend towards more logic on the motherboard and less in the slots.

For experimenting with interrupt-driven protocols, this design wires a '576 output bit (Port 2, bit 3) to generate an interrupt to the PC.

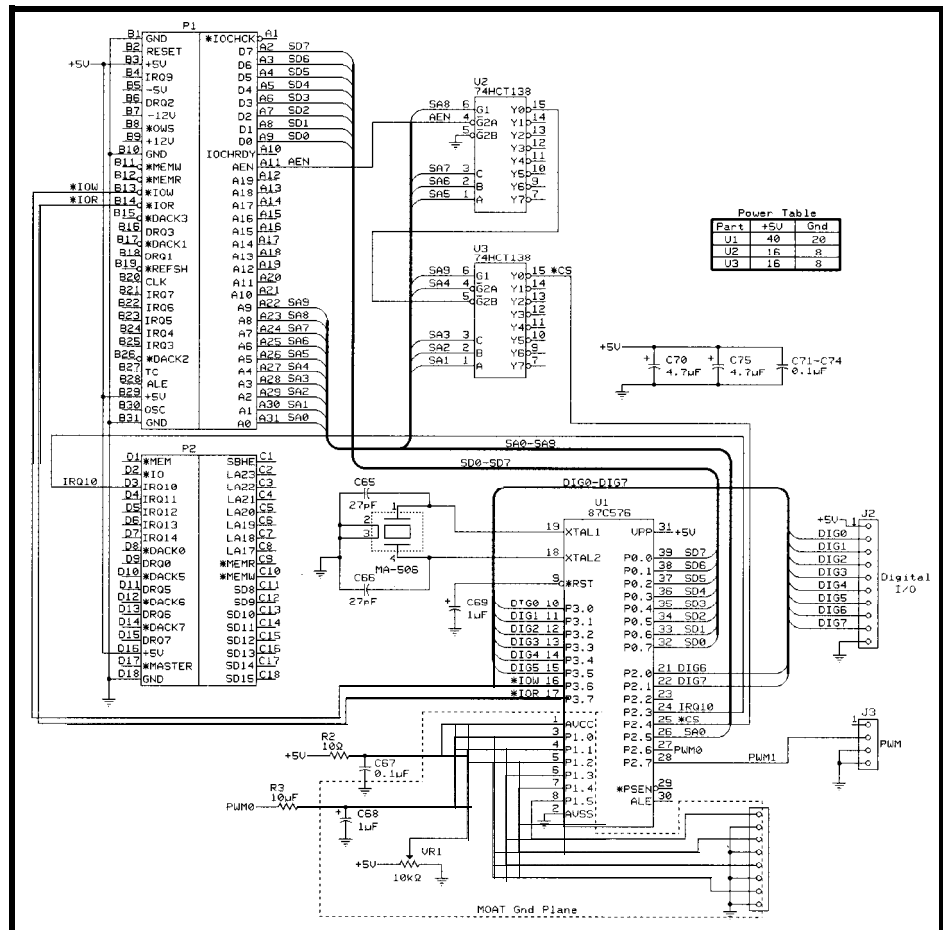


Figure 6—This application example connects the '576 to the ISA bus to create a PC plug-in I/O coprocessor.

20MHz PC with On-board LCD, Video & Flash

Only
\$225.00
OEM



PC/+Vs

FEATURES:

- V-40H Microprocessor
- 16MHz or 20MHz CPU
- PC® compatible BIOS & Architecture
- CMOS Floppy Disk controller
- VGA/LCD controller
- Up to 640KByte User DRAM
- Up to 2MByte programmable Flash™
- 2 16C550 RS-232C Serial Channels
- 1 V-40H Serial Port
- 5380 SCSI controller
- 8-bit general purpose parallel port
- Size: 4" x 4" (100mm x 100mm)
- Power consumption under 2 watts
- Requires only 5V operation

Megatel Computer Corp.
125 Wendell Ave.
Weston, Ont.
M9N 3K9 Canada

1-888-SMALL-PC

Phone 416-245-2953
Fax 416-245-6505
email smallpc@megatel.ca
website <http://www.megatel.ca>

"Embedded PC's for the OEM"

megatel®

Eight bits of digital I/O (six from Port 3 and two from Port 2), four of the A/D ports, and one of the PWMs are brought out to connectors. The second PWM is configured as a DAC using an RC filter and fed back to the fifth A/D port to gain a convenient analog-loop-back test capability.

The '576 code starts with four functions (`get_digital_byte`, `put_digital_byte`, `increase_pwm`, and `get_analog_channel`) (see Listing 1). As their names imply, they let the PC request that the '576 read or write eight bits of digital I/O, increase the PWM value, or read the ADC.

The main module simply waits for the PC to issue the command and performs the appropriate action. For the read commands (digital and analog), it returns the requested data.

On the PC side, the two functions (see Listing 2) `put_byte` and `get_byte` handle byte transfers to and from the UPI, respectively. Two other functions, `send_command` and `get_response`, assemble two-byte messages.

For `send_command`, the two bytes consist of the command and parameter. For `get_response`, the first byte is a dummy value (the echoed command), and the second contains returned re-

Listing 2—This code for the PC accesses the UPI pod and is written in Microsoft C.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dos.h>
#include <conio.h>

#define UPI_DATA_REG 0x0
#define UPI_STATUS_REG 0x1
#define UPI_BASE 0x300
#define IBF 0x02
#define OBF 0x01

unsigned char put_byte (unsigned short port, unsigned char data) {
    unsigned short dummy;
    if (inp(port + UPI_STATUS_REG) & IBF)
        return (-1);
    // 576 busy? dummy = outp(port + UPI_DATA_REG, data);
    // send a byte, wait until 576 has read it
    while (!(inp(port + UPI_STATUS_REG) & IBF));
    return(0);
}

unsigned char get_byte (unsigned short port) {
    while (!(inp(port + UPI_STATUS_REG) & OBF));
    // wait for byte return ((unsigned char)inp(port + UPI_DATA_REG));
}

void send_command(unsigned char command, unsigned char info)
{
    put_byte(UPI_BASE, command); put_byte(UPI_BASE, info);
}

unsigned char get_response (void) {
    unsigned char dummy;
    dummy = get_byte(UPI_BASE); // get the response
    return(get_byte(UPI_BASE)); // return the info byte
}

void main(void)
{
    printf("\n576 UPI Interface Utility\n");
    printf("\nGet Analog Channel 01 00 returned "); /* send command */
    send_command(1,0); /* display result on sdtio (screen) */
    printf("%02x", get_response());
    printf("\nGet Digital Byte 02 00 returned ");
    send_command(2,0);
    printf("%02x", get_response());
    printf("\nPut Digital Byte 03 55 returned ");
    send_command(3,0x55);
    printf("%02x", get_response());
    printf("\nIncrease PWM 04 01 returned .");
    send_command(4,1);
    printf("%02x", get_response());
}
```

sults for get_analog_channel and get_digital_byte.


main issues these four commands sequentially and displays the results onscreen. For testing digital I/O commands, the PC first directs the '576 to read the eight bits and then writes 55H to them. Probing the pins verifies the alternating bit pattern output (i.e., 55H = 01010101B), or pull bits high or low and see the results on the PC screen.

For the analog-loopback check, each run of the PC program increases PWM. Put the program in a loop and watch the A/D input (at 8-bit resolution) value cycle onscreen from 00 to FF repeatedly.

MORE IS POSSIBLE

This example illustrates one PC-UPI interface and protocol. Obviously, low-level I/O commands can also be handled with plain analog and digital I/O chips.

Full exploitation of the IOP calls for high-level commands that offload a large portion of the PC processing and real-time burden. Then, instead of just reading the ADC, it performs higher level tasks (e.g., averaging multiple readings or limit-checking the input).

Similarly, instead of just increasing the PWM output, the '576 can autonomously perform complex waveform generation. The goal is to fully offload the PC of high-speed and short-latency I/O tasks, freeing the main CPU for ever-more complex application processing. 

Bill Houghton is a senior applications engineer at Philips Semiconductors. He has over 12 years' experience specializing in the 80C51 microcontroller family. You may reach him at bill.houghton@sv.sc.philips.com.

SOURCES

8x576
Philips Semiconductors
811 E. Arques Ave.
Sunnyvale, CA 940883409
(408) 9915207
Fax: (408) 991-3773

IRS

407 Very Useful
408 Moderately Useful
409 Not Useful

EMBEDDED BIOS

BIOS ADAPTATION KIT

Listen to What our Customers Say

"We're impressed by the documentation and the readability of the code." - M. Ryan

"We are very pleased with the General Software BIOS and look forward to working with you to bring our product to market." - R. Levaro

Embedded BIOS is well-structured and documented, and technical support at General Software is excellent. - J. Toivanen

"I am sure we made the right decision to buy our BIOS from General Software." - R. Fillon

"Embedded BIOS is really the product for embedded PC designs. You were absolutely right." - J. Josse

The BIOS for Consumer Electronics

Why You Should Choose Embedded BIOS, Too

- BIOS, DOS, Flash Disk With One Low Royalty
- Instant Boot, Console Redirection, & Much More
- Expert Support with Guaranteed Response Time
- We Work Closely With Acer, AMO, Intel, & RadiSys to Deliver you a Proven, Tested, Feature-Packed BIOS
- Millions of Units Already Licensed

BIOS Adaptation Kit Includes:

- Complete Source Code
- Binary Configuration Program
- Quick Start + Over 600 Pages of Printed Documentation

For Details, Call 1-800-850-5700

General Software, Inc.
 320 108th Ave. N.E., Suite 400 • Bellevue, WA 98004
 Tel: 206.454.5755 • Fax: 206.454.5744 • Sales: 800.850.5755
 http://www.gensw.com/general • E-Mail: general@gensw.com
© 1997 General Software, Inc. General Software, the GS Logo, Embedded BIOS and Embedded DOS are trademarks of General Software. All rights reserved.


#116

Finally, Standard RS-485 Network Software

With Cimetrics' 9-Bit μ LAN you can link together up to 250 of the most popular 8- and 16-bit microcontrollers (8051, 80C196, 80C186EB/EC, 68HC11, 68HC16, 68332, PIC16C74).

The B-Bit μ LAN is:

- ▶ **Fast**- A high speed (62.5k baud) multidrop master/ slave RS-485 network
- ▶ **Flexible** — Compatible with your microcontrollers
- ▶ **Reliable**- Robust 16-bit CRC and sequence number error checking
- ▶ **Efficient**- Low microcontroller resource requirements (uses your chip's built-in serial port)
- ▶ **Friendly**- Simple-to-use C and assembly language software libraries, with demonstration programs
- ▶ **Complete**- Includes network software, network monitor, and RS-485 hardware
- ▶ **Standard** — The 9-Bit μ LAN is an asynchronous adaptation of IEEE 1118



Microcontroller or PC Master

Intel 8051 HVAC/R

Intel 80C186EB/EC

Intel 80C196 Transportation

Microchip PIC16C74

Motorola 68HC11 Process Control

Motorola 68HC16 Medical Equipment

Motorola 68332 Laboratory Data Acquisition

RS-485 MULTIDROP NETWORK

55 Temple Place • Boston, MA 02111-1300 • Ph 617.350.7550 • Fx 617.350.7552

117

FEATURE ARTICLE

Art Sobel

directly with minimal external components.

Acorn designed the ARM7500 into its low-cost RiscPC A7000, as well as the Digital TV Settop box. It is also present in Oracle's Network Computer. And, it's in Teknema's Easy Rider Internet Browser Computer and View-call America's Webster.

ARM7500 has been in production since July 1995. Figure 1 shows the ARM7500's main components.

CPU CORE

The ARM704 CPU core powers the ARM7500. It contains the ARM7 CPU as well as 4 KB of cache, MMU, and write buffer.

The cache is partitioned as a four-way set interleave design. For each set of upper address bits, four entries (each containing four words) can be stored in the cache. If all possible entries are filled, one is randomly discarded and overwritten.

MEMORY CONTROLLER

The memory controller interfaces with the three types of memory in the system—DRAM, ROM, and I/O. Most of the memory is DRAM, and the ARM7500 can directly control up to 256 MB, far more than is needed for most applications.

DRAM is the common fast-page variety with RAS before CAS refresh. It can be configured to be 16 bits wide for cost-sensitive applications, but that halves its bandwidth capability.

High DRAM bandwidth is of utmost importance when simultaneously servicing a fast CPU and high-density display. DRAM timing is directly controlled by the MEMCLK input, which can be asynchronous from the CPU cache clock.

The ARM7500 handles ROM in 16- and 32-bit widths. ROM timing has a programmable number of MEMCLK cycles. Flash memory can be accommodated as a peripheral on the I/O bus, but later versions have flash in ROM space.

Many I/O cycles are similar to ROM cycles, but some can be extended with the WAIT pin. Although the external I/O data bus is only 16 bits wide, it can extend to 32 bits with external latches.

Embedding the ARM7500

Part 1: The Chip and Development Board

Art brings us up close and personal to the ARM7500 and its development board. You'll soon see that this little baby has a lot of power and peripherals. It's almost an entire PC in a chip!



In this series, I introduce the highly integrated ARM7500 computer on a chip and the design of a simple ARM7500 application board. I also discuss the ARM version of the NetBSDOS—RiscBSD—that's been ported to the ARM7500.

The ARM7500 is a low-power, high-performance, single-chip computer centered around an ARM microprocessor core. This direct descendent of the Acorn Archimedes computer contains the original four-component custom chipset (i.e., CPU and memory, I/O, and video/sound controllers) as well as 4 KB of cache memory, LCD support, and PC peripherals.

In 1988, the original Acorn chipset was produced on a 3- μ m CMOS process. The current part is made with VLSI's 0.6- μ m CMOS process.

Processor speed also improved. The original chip ran at 8 MHz, producing 6,000 Dhrystones. The ARM7500 zips along at 40 MHz, producing 59,000 Dhrystones. The raw performance is similar to a 486DX4-100 but without the heatsink.

To maximize the potential of the ARM processor macrocell, the ARM7500 contains memory and I/O control on-chip, enabling external memory devices and peripherals to connect

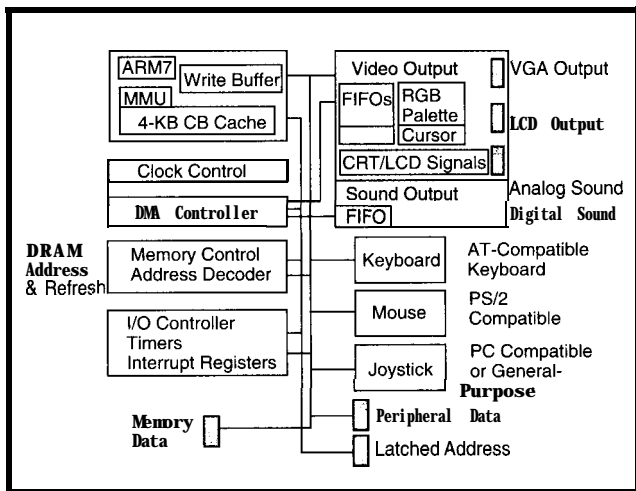


Figure 1—This diagram of the ARM7500 microcomputer shows the main operational blocks—CPU, memory controller, video controller, I/O controller, and embedded-PC peripherals.

Three DMA registers for the video set up a circular buffer that can be used for hardware scrolling. However, this feature is not so useful

with windowing software (e.g., X-Windows or a Web browser).

The cursor DMA is simpler and has a single DMA INIT register. Sound output requires dual DMA pointers that create a continuous datastream to the sound circuits.

VIDEO AND SOUND

The video and sound output logic is derived from VIDC (video controller) chips. The VIDC connects to the data bus with one chip-select signal and several DMA write signals.

The written control data contains the address of each programmable register and the controlled function's data field. With this configuration, the registers are just write only.

There are 34 dedicated registers for video functions and 10 sound-control registers. Also included are 256 individually addressed 28-bit video-palette registers. They are split into 8 bits for red, green, and blue entries for each logical color.

Separate FIFOs exist for regular video, cursor, and sound data. The large num-

Table 1—The memory map for the ARM7500 has assigned physical addresses. These are remapped by the *Demon* (Debugging Monitor). Both maps are shown.

Megabytes	From	To	Demon Map	Description
0	00000000	00ffffff	20000000	ROM Bank 0
16	01000000	01ffffff	21000000	ROM Bank0
32	02000000	02ffffff	disabled	Reserved
48	03000000	0300ffff	10000000	Module I/O Space
	03010000	0302bfff	10010000	16-MHz PC I/O (Combo)
	0302c000	0302ffff	1002c000	Reserved
	03030000	0303ffff	10030000	Module I/O Space
	03040000	031fffff	10040000	Reserved
	03200000	0320ffff	10200000	I/O Registers
	03210000	033fffff	10210000	Simple I/O
	03400000	034fffff	10400000	VIDC Registers
	03500000	03f0ffff	10500000	Reserved
64	04000000	07ffffff	disabled	Reserved
128	08000000	0ffffff	18000000	Extended I/O Space
256	10000000	13ffffff	00000000	DRAM Bank 0
320	14000000	17ffffff	disabled	DRAM Bank 1
384	18000000	1bffffff	disabled	DRAM Bank2
448	1c000000	1ffffff	disabled	DRAM Bank 3
512	20000000		memory map repeats	

ber of registers makes the video parameters very flexible.

The VIDC also supports LCD panels. The video controller supports a variety of single and dual panels, binary, greyscale, and color. Some panels require external circuitry for panel protection and interfaces.

The old Acorn computers had an 8-bit companded audio DAC built into the VIDC. Preserving that legacy, the VLSI ARM7500 adds support for external 16-bit CD-ROM serial DACs.

MEMORY MAP

The addresses in Table 1 are physical addresses encompassed in 29 of the 32 address bits. Their position in memory may be moved to other virtual locations.

Thus, even in the debugging monitor (*Demon*), the DRAM is moved to lower addresses to allow for exception vector programmability.

DEVELOPMENT BOARD

When I contemplated the design of an ARM7500 development board, I tried to include support for all possible uses customers might have for the part. Any one application might have features removed or modified. The ultimate use was a Unix workstation. Other applications were subsets.

One important consideration was the mechanical configuration of the board. I decided that the best configuration was a direct fit into a flat PC workstation chassis.

The positions of the connector and mounting holes were adjusted to fit a case from Leadman Electronics. The

I/O

In the ARM7500's full complement of onchip peripherals, there are about 70 programmable registers in the I/O and memory controller (52 in the internal I/O controller and 18 DMA controller registers).

Of the 11 open-drain I/O pins, 8 can be external interrupt pins. The keyboard and mouse interfaces have two open-drain pins for data and clock which support the AT and PS/2 mouse interface protocols.

Four pins are used for the joystick interface, which supports the PC-type variable resistor joysticks with capacitors on the motherboard. Each pin connects to an open-drain clamp transistor and a comparator.

When the open-drain transistor is turned off, a 16-bit counter operating at 2 MHz starts. It is stopped by the comparator being triggered as the variable resistor charges the attached capacitor. An interrupt is generated when the programmed combination of comparators is triggered.

This interface can also be used as four general-purpose timers measuring rising-edge timing from external sensors (e.g., a sonar detector).

DMA

Three DMA channels supply the internal video, video cursor, and sound controller with data from the DRAM.

Fast-page DRAM has a limited total bandwidth, so the display robs the CPU of memory cycles. Beyond a certain point of display resolution, either the display or CPU is starved for memory cycles.

case has a 100-W power supply, holders for 3.5" floppy and hard disks, and a wide, open, back panel with room for many I/O connectors.

A board extender inserts into an ISA slot and offers three other slots at right angles. The motherboard only includes that ISA slot. The space usually allocated to slots is taken up by peripherals that used to fit into I/O cards.

Two important chips complement the CPU. For the I/O combo chip, I chose the SMC FDC37C665, but similar chips from other suppliers can be used. The '665 contains two 550-type serial ports as well as a bidirectional parallel port, high-density floppy controller, and IDE chip select.

The second chip—the SMC92C94—is a self-contained Ethernet controller that directly handles 10BaseT with a transformer filter.

THEORY OF OPERATION

Figure 2 shows the first page of the RC7500 schematic. The board clocks are generated by a Chronitel CH9294. It provides three clocks, one being the

14.318MHz 4x color-burst frequency used on PC motherboards to generate the OSC signal for the expansion slot.

The three 7500 clocks connect to the MCLK output of the CH9294. This output is programmed by the jumpers JP1–3 to be nominally at 65 MHz. Therefore, the processor cache clock will be 32.5 MHz after being divided by 2 internally.

The CH9294's VCLK output is programmed by a 74HCT377 register. This clock provides a video clock for VGA and SVGA pixel rates and goes back into the VIDC section of the ARM7500. Standard 72-pin, 32-bit SIMMs are used for main memory.

The PCF8583 became the real-time clock chip because the ARM7500 has two pins dedicated for I²C operation. It offers additional battery-backed RAM for saving system hardware data. The RC7500 has room for a 3-V lithium battery or a standard external PC motherboard battery.

The PCF8583's alarm signal connects to the *NEVENT1 pin. The panic button connects to NEVENT2. Either

interrupt can wake the ARM7500 from power-down modes.

The RC7500 is equipped with four ROM/EPROM sockets. Since the ARM7500 can support 16- and 32-bit memories, as little as 64 KB (2 x 27C256) or as much as 4 MB (4 x 27C080) of ROM can be used.

So, anything from a boot loader to a complete OS with many utilities can be placed in ROM. Addressing of various types of EPROM is provided by jumpers JP4–8.

The ARM7500 generates the low-going *ROMOE signal, which is connected to the *ROMOE pins. When an external ROM emulator is hooked up to the logic analyzer port, the onboard ROM can be disabled by pulling the DISAROM signal high.

The ARM7500 has two electrically identical keyboard ports assigned to keyboard and PS/2 mouse connectors as shown in Figure 3. These are AT-compatible bidirectional open-drain serial ports.

Dual diodes and capacitors are used for static protection and noise reduc-

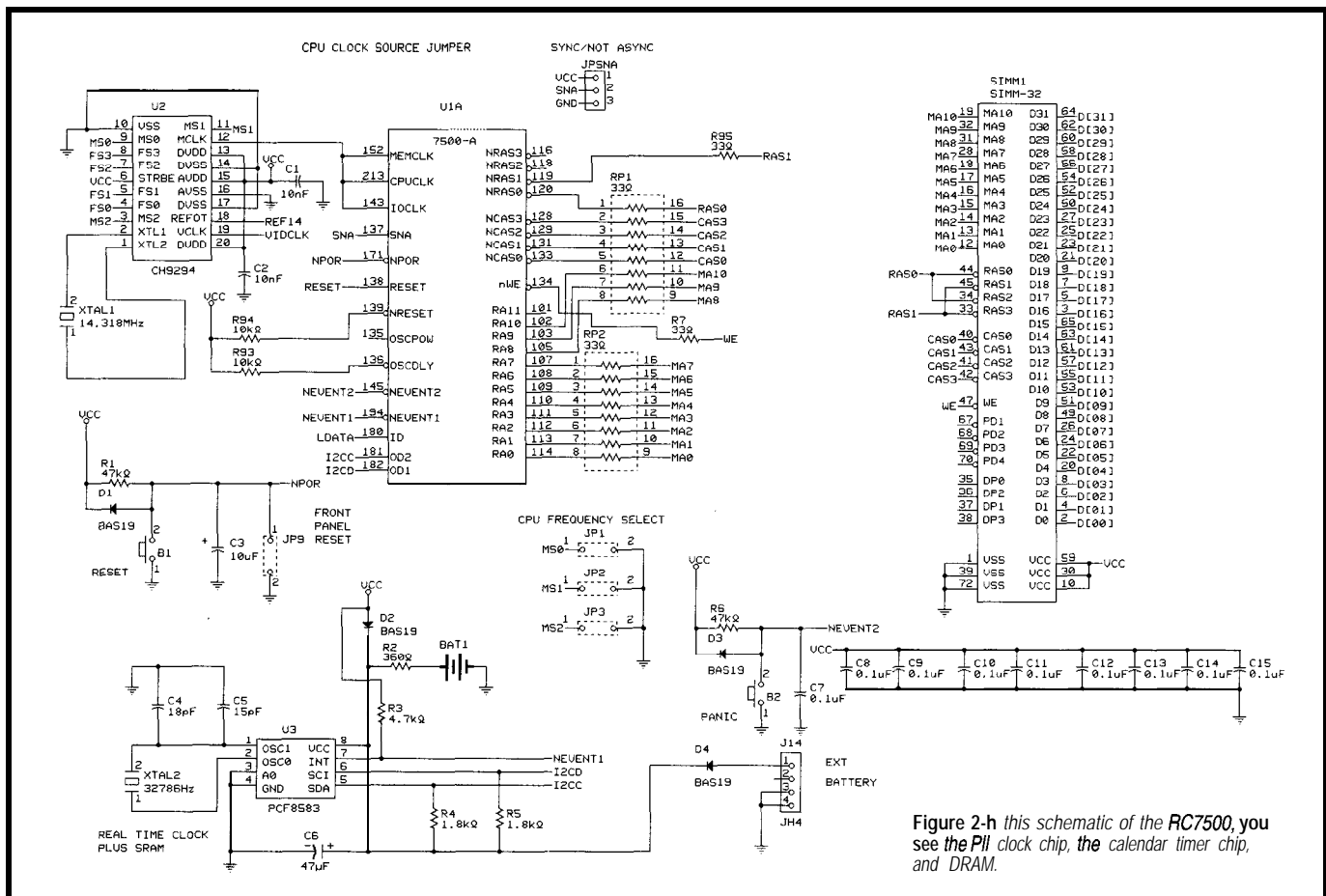


Figure 2-h this schematic of the RC7500, you see the Pll clock chip, the calendar timer chip, and DRAM.

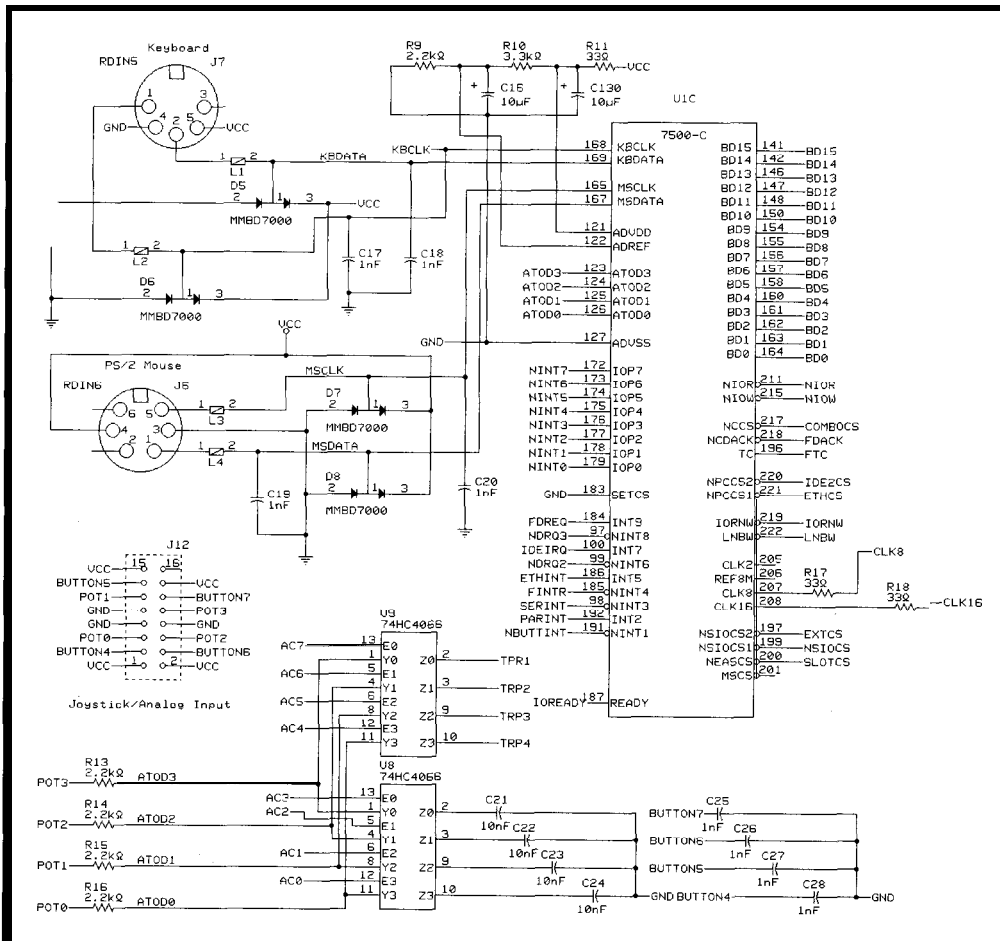


Figure 3—Here are the keyboard, mouse, and joystick connections. The joystick can be programmed for use as four general-purpose timers.

tion. The bit rate on these pins is fairly slow, so 0.001- μ F caps do not affect the signals.

The ARM7500 has four single-slope ADCs. Each pin has an open-drain transistor and attached voltage comparator controlling a 16-bit ripple counter. When the transistor is off, the voltage on the pin rises as current from a resistor attached to the +5-V supply charges a capacitor connected to ground.

When voltage rises to the comparator threshold at 2 V, an interrupt is signaled and the 16-bit ripple counter stops. The counter is run from the IOCLK divided by 32 (ideally 2 MHz), giving a 32,768- μ s maximum time period.

The 100-k Ω joystick pots provide the resistor to +5 V. In a PC, they are routed to an SE555 timer circuit timed by the CPU with interrupts off for as long as several milliseconds. The self-timed ARM7500 A/D circuits prevent having to turn the CPU interrupts off for such a long period.

These unique inputs are more versatile than joystick-only ports. If an external rising signal is substituted for the resistor- and capacitor-formed slope, the pins can be used for simple timers.

Distance to nearby objects can be measured for four inputs simultaneously (up to 16' at 2 ms per foot). To be used as a timer, the capacitors are switched out and pull-up resistors are switched in to each A/D input with 74HC4066 analog switches.

The audio and video connections are shown in Figure 4. I opted for the higher fidelity of an inexpensive 16-bit CD DAC (Philips TDA1543).

The ARM7500 supports direct-drive RGB outputs into doubly terminated 75- Ω lines. I added protection diodes to these outputs because CRTs can collect static charges very easily.

LCD outputs for straight and split-screen displays connect to a 16-pin connector (P3) for convenience. Since there is no standard LCD connector, an adapter must be made for each LCD

panel connected to the interface. This connector also has the signals for a genlock add-on circuit.

The vertical sync resets when the SINK pin is held high. However, there's no such circuit for horizontal sync. It must be achieved by line locking with a dot-clock PLL and phase detector on the horizontal sync pulses.

The sound output generated from the TDA1543 could have been connected to a 3.5mm stereo connector. I chose a more complicated circuit, using the National LMC1982 sound controller IC, plus the LM2878 amplifier SIP, which generates 5 W per channel.

The LMC1982 is controlled with a serial datastream, using a minimum pin count from an added 74HC377 data latch. The only problem left is where to put the speakers!

Since this board is for unique applications (or porting old ones) for the ARM, I provided an ISA-like plug (see

Figure 5). All signals to the slot are buffered to protect the 7500. Most existing boards can be made to work.

This board is not completely ISA compatible, which would have required a redesigned ARM7500. Principally, the board only handles slave devices.

Only two pseudoDMA channels are handled by FIQ interrupts. No byte packing is done, so the designer must know where the bytes are going in any port. Software does byte packing and unpacking. Slot addressing is carved out of the expansion memory in several segments for activating MEMR/W, IOR/W, and DMA signals.

A 96-pin connector has a selection of the bus signals for expansion into a logic analyzer pod. It could also be connected to a ROM emulator or writable control store. ROM, DRAM, and many I/O operations can be decoded.

PORTS AND INTERFACES

The typical I/O devices found in a PC are all compacted into Super IO or

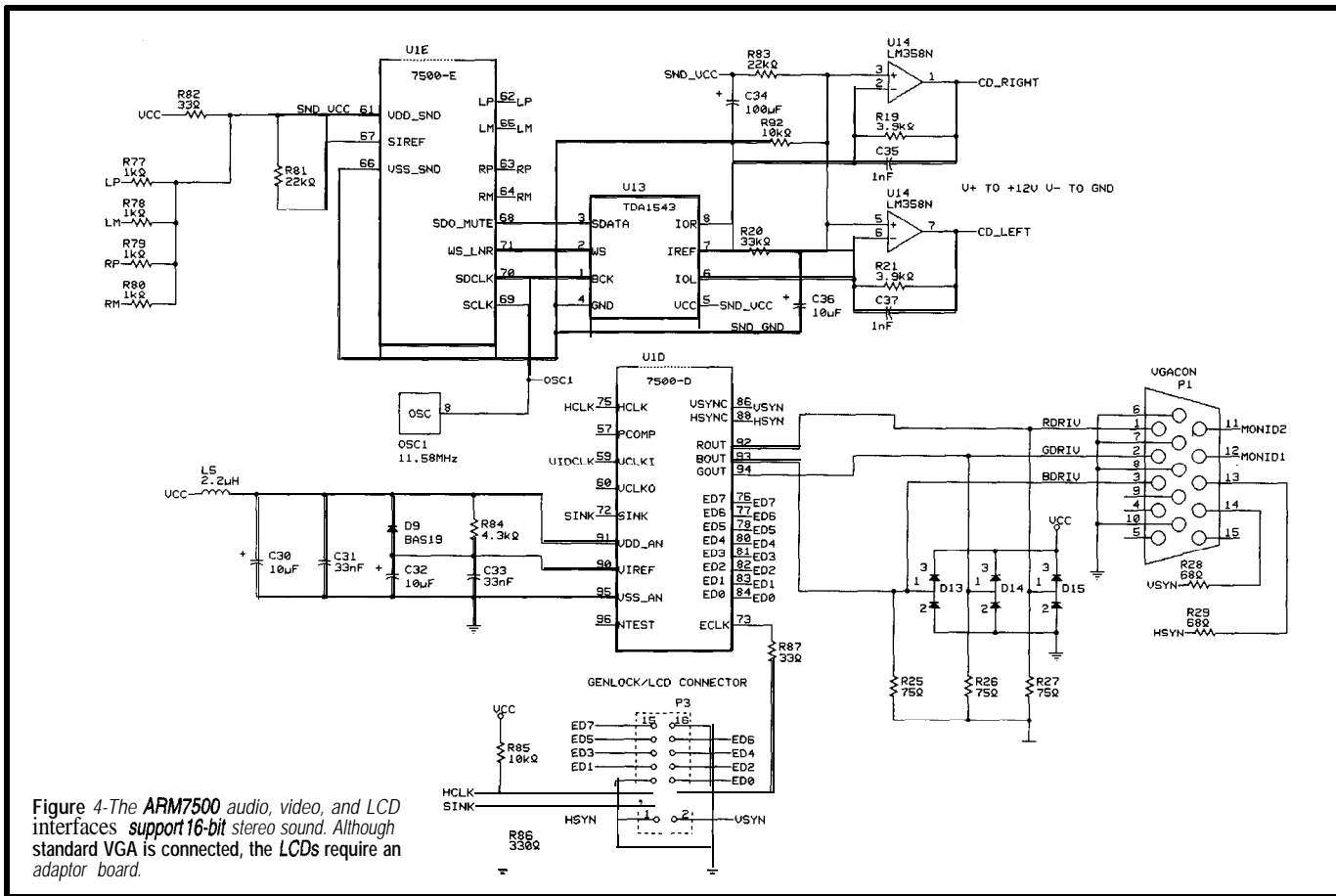


Figure 4-The ARM7500 audio, video, and LCD interfaces support 16-bit stereo sound. Although standard VGA is connected, the LCDs require an adaptor board.

combo chips. I couldn't resist these, and neither could Acorn. We both use the SMC37C665, so software like RiscBSD requires fewer rewrites.

There are two serial ports. Port 1 connects to a 9-pin subminiature D connector (J2) that's PC compatible and also found in all other ARM development boards. The same serial hookup for VLSI and ARM PID and PIE boards can be used.

The second serial channel is terminated by a 10-pin header (P14) that can be connected to a 9- or 25-pin D connector with mass-terminated wire as found in many PC I/O boards.

The parallel port supported by the SMC37C665 is bidirectional and ECP compatible. The proper driver supports these IEEE-1284 interfaces. The parallel port is terminated with a 26-pin header that translates to a 25-pin D connector with a short cable.

The SMC37C665 has a PC-compatible enhanced floppy controller with digital data separator. It directly drives all normal floppy types up to 2.88 MB. Unfortunately, all these floppy-drive

manufacturers shut down at the same time. The RC7500 chassis mounts a single 1.44-MB 3.5" floppy drive.

IDE1 is mapped in the combo's address space, and the chip selects are produced by the 37C665 combo chip. IDE2 is mapped in the simple expansion space, so two different master drives can operate simultaneously for faster operation than a master/slave configuration (although two masters and two slaves can be supported). Usually, this is one hard drive and one ATAPI IDE CD-ROM drive.

Ethernet is all the rage and fairly inexpensive. The SMC92C92 supports both twisted-pair and coaxial versions. The 10BaseT version uses shielded twisted-pair wiring into star hubs.

With the proper software OS and TCP/IP protocols, the RC7500 can be made into a workstation that can coexist in the same network with Unix boxes and PCs-and onto the Internet!

The Ethernet local address is stored in an 93C46 EEPROM that also has room for the Host ID-if one is ever needed. The 10BaseT is terminated in

an RJ-45 8-pin telco connector (J5), of which only 4 pins are used.

Power is provided by a standard PC power connector, supplying the ± 5 and ± 12 V needed by the motherboard and expansion slot. A 4-pin disk-drive connector can be used that supplies the +5 and +12 V needed on the motherboard.

Two 74HC377s control the LED bar and the joystick input load programming.

ARM7500 APPLICATIONS

The ARM7500's range of features makes it extremely flexible. It can be programmed to optimize for high performance, low power, or both.

The ARM has an eight-year history in Acorn PCs. Its low cost and integration offers places like schools a lower price computer. With NetBSD, the ARM offers display nodes for campus networks.

Power-management circuitry and the power-efficient characteristics makes the chip particularly suitable for low-power portable applications. Since 32- and 16-bit wide memory

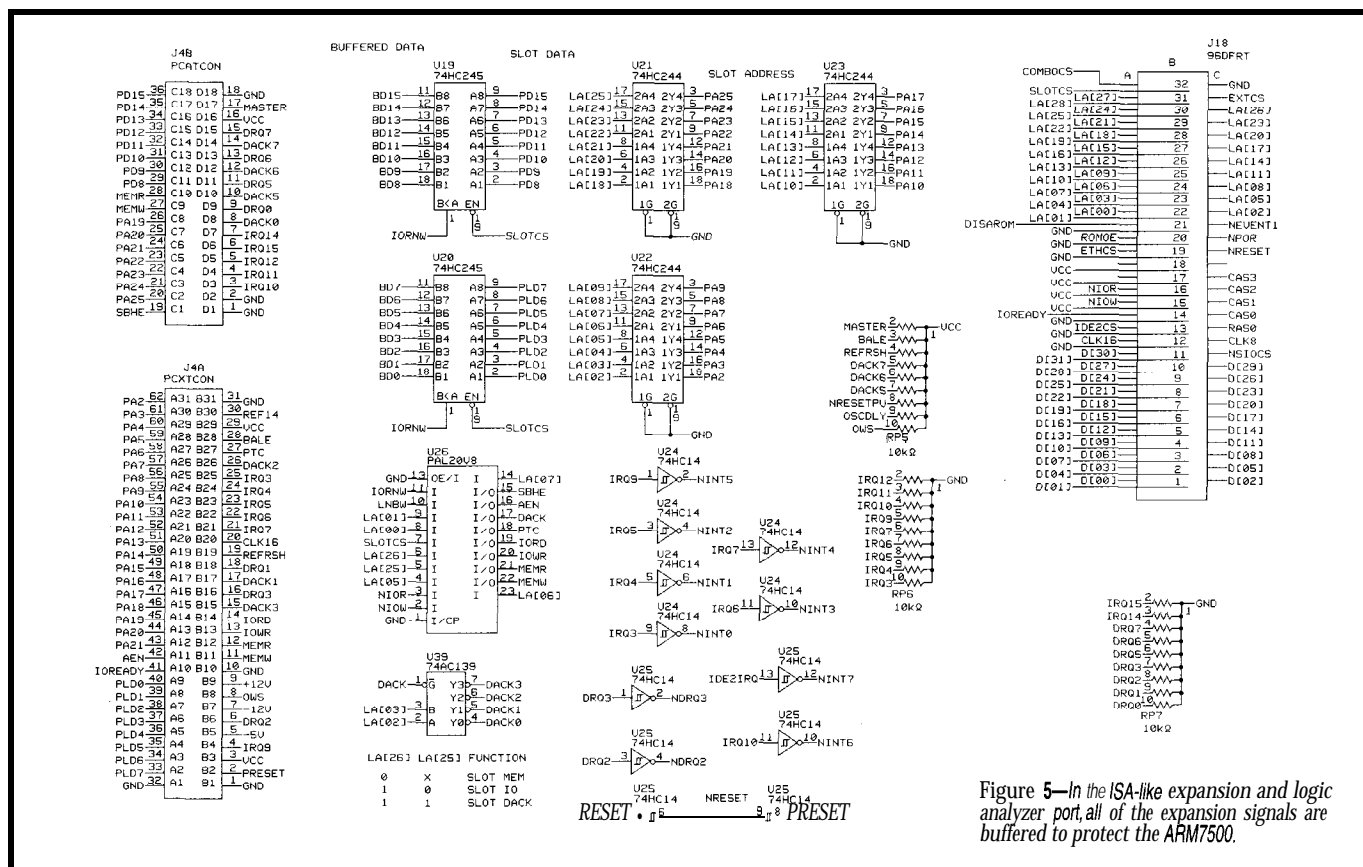


Figure 5—In the ISA-like expansion and logic analyzer port, all of the expansion signals are buffered to protect the ARM7500.

systems are supported, high-performance 32-bit or lower-cost 16-bit systems can be designed.

The ARM7500 drives monochrome (single or dual panel) and color LCDs, which is useful for electronic instrument displays. The ARM7500 has been designed into marine and aircraft GPS and is presently being used in a cockpit display system that makes a small-aircraft panel resemble a 747's.

The combination of ARM macro-cell processing power and peripheral macrocells enable the chip to fit into any application requiring high-quality video, sound, and general data I/O (e.g., laser karaoke and DVD players).

Its good performance, low cost, and integration make it an economical solution for the sub \$500 Web terminals. It can also be embedded in TVs, replacing the simple 8-bit micro that handles the IR receiver to switch channels and adjust volume.

As a display device, TV currently suffers from low-bandwidth video amplifiers and interlaced scan. It loses display detail and has an annoying flicker in high-contrast fonts and line drawings. To compensate, high-detail

areas must be antialiased and antitwitered (i.e., the frames must blend into each other to reduce contrast).

In Part 2, I'll discuss programming the ARM7500 in the RC7500 board. I'll also take a look at the Demon ROM and console test program. □

Art Sobel is the hardware applications manager for embedded products at VLSI Technology. He has spent 24 years in Silicon Valley designing disk-drive electronics and controllers, laser interferometers, laser-printer controllers, many controller chips, and speech synthesizers. You may reach Art at sobel@sanjose.vlsi.com.

SOFTWARE

RiscBSD and GNU cross-development software is downloadable from <www.ph.kcl.ac.uk/~amb/riscbsd/docs.html> or <ftp.netbsd.org>. A Helios/ARM RTOS package is available from Perihelion Distributed Software at <www.perihelion.co.uk>. More information on ARM7500 and boards can be found at <www.arm.com/Pro+Peripherals/ASSPs/7500> and <members.aol.com/sobellini>.

SOURCES

ARM7500 chips and RC7500 development board

VLSI Technology
1109 McKay Dr.
San Jose, CA 95 13 1
(602) 752-8574
dale.roark@tempe.vlsi.com

ARM7500

Cirrus Logic, Inc.
3100 W. Warren Ave.
Fremont, CA 94538
(510) 623-8300
Fax: (510) 226-2180
charley@corp.cirrus.com
www.cirrus.com/prodtech/
ov.netmobile/ps7500.html

ARM7500 case

Leadman Electronics
2980 Gordon Ave.
Santa Clara, CA 95051
(408) 738-1751

IRS

410 Very Useful
411 Moderately Useful
412 Not Useful

Four Bits Unleashed

FEATURE ARTICLE

Brad Stewart

Our bits rule! Billions of 4-bit microprocessors are in use today in all kinds of applications.

According to market research, the number of 4-banger MCUs (over a billion) shipped worldwide in 1996 accounts for nearly 32% of all units sold. That's over four times the number of 16-bit MCU units sold. (For the record, 54% of MCU units sold are 8 bit.)

You may think that a 4-bit micro is outdated, old fashioned, and even retro. So, why are they still used in such large numbers? And, why should you use a nibble nosher when there are so many PICs, '05s, or Z8s to choose from?

There are many reasons, of course, but consider these. These little guys have been around a long time—over 25 years. Remember the 4004?

They're cheap. These 4-biters are often around \$1 in high quantity. And, they consume very little power, often only microamps. Some versions can even run on 1.5 V.

Lower circuit complexity allows for larger chip geometries which results in an integrated circuit that is rugged and less susceptible to static discharge and high-voltage electromagnetic fields.

Lower power means lower levels of EMI emissions making them easier to FCC certify.

These devices are feature rich and include useful peripheral circuits such as LCD and vacuum fluorescent display drivers, timers, counters, ADCs and DACs, power management, real-time clocks, LED drivers, I/O, serial ports, sound generators, and interrupts.

Fast. Yes, you read it right! It's not uncommon to find turbo fours with a

1- μ s instruction time. Adding to the computational efficiency is that most instructions are executed in one cycle. Hey, that's around a MIPS!

So where do these little jewels pop up? More places than you might think.

Here's a short list: toasters, cameras, watches, exercise equipment, smart batteries, calculators (including graphing and scientific), battery chargers, bread makers, coffee makers, washing machines, and weather gauges.

Oh, and then there are vacuum cleaners, garage-door openers, ovens, thermostats, hand-held games, sprinkler controls, water softeners, fishing lures, automotive systems, telephones, organizers, portable stereos, and remote controllers, not to mention multi-meters and test instruments.

By far, the largest suppliers of 4-bit MCUs are Japanese and Korean companies like NEC, Sharp, Samsung, Toshiba, and S-MOS. Sharp has dozens of models and variants of their 4-bit micros, each targeted for specific markets.

For example, the Sharp SM3511 has a 3 136-dot LCD graphics driver, 8 KB of SRAM, half a megabit of ROM, wide supply voltage, and 1- μ A standby current. In addition, the LCD driver signals are multiplexed to scan a keyboard and access up to 16 MB of ROM.

This part is designed specifically for electronic portable hand-held organizers, language translators, dictionaries, and Bibles.

Four-bit MCUs come in many package sizes and form factors, including bare die which can be bonded directly to a circuit board or a thin flex film.

UNDER THE HOOD

Let's look at one of these nibble crushers in more detail. Figure 1 shows a block diagram of the SM5K6 manufactured by Sharp Electronics.

This 4-bit parallel processor has 52 instructions, 4 KB of ROM, 256 nibbles of RAM, and an 8-channel 10-bit ADC. It has 8 direct-drive LED outputs, low-power standby mode, two counters with 15-bit prescaler (use one as a watchdog timer), synchronous serial port, 3 internal and 2 external interrupts, 4 inputs, and 20 input/outputs.

It runs on a 2.0-5.5-V supply and is available in a 36-pin QFP or 30-pin

Even today when many companies boast of new-fangled megachips, the 4-bit processor holds a healthy third of the market. But, as Brad points out, these new 4-biters have a lot more oomph than their predecessors.

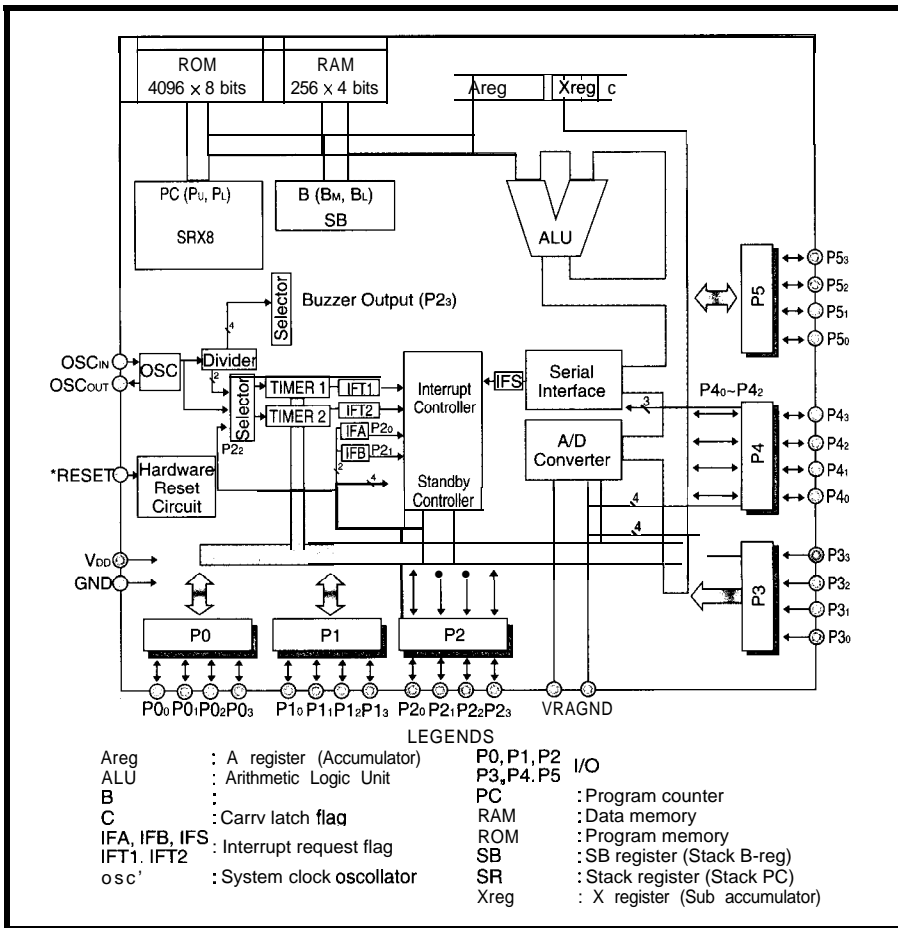


Figure 1—The Sharp SM5K6 4-bit microcontroller is a complete system on a chip featuring a 10-bit A/D converter, dual timer/counters, a serial interface, and 20 programmable I/O pins.

SDIP package. For prototyping and small production runs, the part is available as an OTP (one-time programmable) version.

Code development is done using a PC-based assembler/linker and an in-circuit simulator. One of the more “modern” enhancements to 4-bit development is the introduction of a C-like structured assembler, which I discuss in more detail later on.

PROGRAMMING MODEL

My first exposure to a 4-bit micro was at college. It was briefly mentioned in our digital logic class but was dismissed as obsolete because, by then, Intel had announced the 8008.

Then, just this year, I looked at 4-bit micros a bit more seriously and with renewed interest and curiosity while attempting to dismiss the “wimp” label some were trying to attach to 4 bits (or was it me?).

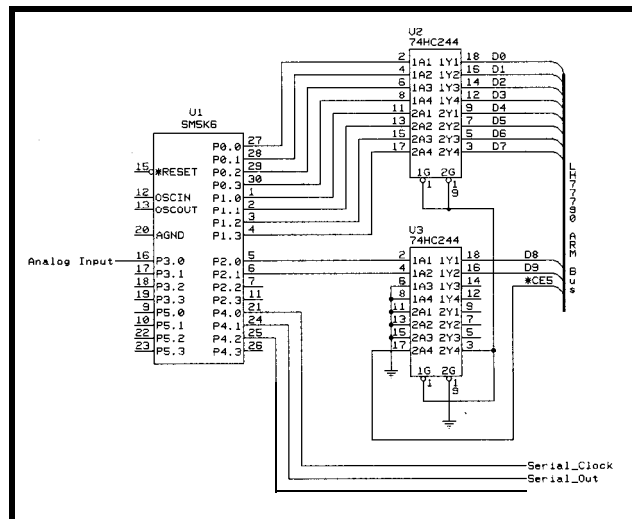
“How is it possible?” I asked myself. “How do you get 52 instructions

and 4 KB of address space with only 4 bits? How can you do anything with 4 bits?”

Well, it works a lot like an 8-bit micro—but different. And, you’ll soon see that the ‘5K6 is anything but a wimp. Let me start by explaining the register set of the ‘5K6.

The A register, or accumulator, is a 4-bit general-purpose register. It is used with the ALU, C (carry) bit, RAM, and transfers data between input and output terminals.

Figure 2—The SM5K6 is used as a 10-bit A/D converter for the ARM RISC microprocessor. The optional S/I/O provides extended capabilities. The bus controller provides a chip-select signal to strobe the A/D data on the data bus.



The X register is a 4-bit register that holds temporary data. The A and X registers can combine to form an 8-bit value.

A 4-bit ALU performs parallel operations. For example, the ADC instruction does a binary addition with a nibble value stored in RAM, the A register, and the carry bit. The result is placed in the A register, and the carry bit is generated if a carry occurs during ALU operation.

The B register is an 8-bit register that specifies the RAM address. The upper 4 bits is called the BM register, and the lower bits is the BL register.

In addition, the B register can be used as a general-purpose register. It can also hold the address of internal control registers during the execution of input, output, and test instructions.

There is also an 8-bit SB register that’s used as a save register for the B register. The contents of B and SB can be exchanged using the EX instruction.

The PC (program counter) is 12 bits wide and specifies the ROM address. The upper 6 bits [one nibble plus one crumb] represents one of 64 pages, and the lower 6 bits denotes one of 64 steps in a page. There is also a set of eight 12-bit SRs (stack registers).

During a subroutine call, the contents of the PC are pushed onto the stack, allowing for up to eight levels of subroutine nesting. It’s a slight limitation, I agree, but it’s better than many 8-bit micros.

In the ‘5K6, there are 4 KB of ROM for program and data storage. All instructions are one byte and operate in

one cycle, with the exception of CALL, DR (clears the clock dividers), T L (transfer over page boundaries), and PAT (table look-up), which require two bytes and two cycles.

Therefore, 90% of the instructions are one byte and are executed in one cycle, or 1 μ s with a 4-MHz clock. (The '5K6 divides the input clock by four.)

Now that you know the basics, here's some code. Listing 1a sets the first 16 nibbles of RAM to 0.

The code in Listing 1a requires 5 bytes of ROM and executes in about 34 μ s. Notice the EXC D X instruction (Exchange Contents of RAM with accumulator and decrement BL). It does many things at once.

First, it performs an exclusive OR of the contents of the BM register with the crumb (2 bit) value x, leaving the result in the lower crumb of the BM register. After the instruction is executed, the instruction that follows E X C D is skipped if the contents of BL register becomes 0xF as a result of a carry.

Using the x value makes it easy to set up the B register, say, for example, when you want to copy or move the contents of RAM to another location across page boundaries. In this case, I set x to 0 so that BM remains 0.

The assembler supports some macro instructions. For example, you can replace the first two instructions with:

```
LBL OFH
```

which loads the entire B (all 8 bits) register with the value 0x0F. It still assembles into two 1-byte instructions.

C-LIKE STRUCTURED ASSEMBLER

One of the more innovative enhancements to 4-bit tool design is the introduction of a C-like structured assembler. It is not a true C compiler, but it does look and feel like C code with the added benefit of making it easier to write and read your code.

It also takes care of the overhead of program transfers across page boundaries. For example, Listing 1a could be written as Listing 1b.

The C-like compiler generates a source listing (see Listing 2) where the C-like code is commented out and it

Listing 1—Here are examples of how an assembler routine (a) and a C-like routine (b) clear the first 16 nibbles of RAM. As you can see, the C-like routine is easier to follow.

```
a)
MCLR:
  LBMX OH   ;load BM to zero
  LBLX OFH  ;load BL to zero
  LAX OH    ;load accumulator with zero
  EXCD OH   ;store contents of A into RAM address B
           ;Then decrement BL and skip next instruction if BL = 0
  TR  .-2   ;jump to PC-2

b)
for (b = 0;){
  *b = 0;    //set contents of address in b to 0
  incb;     //increment BL and skip next instruction if 0
  continue;
  break;
```

Listing 2-A C-like compiler generates an assembly listing equivalent to the routine in Listing 1b.

```
0000 .SM5K6
0000 - ORG 0000h

           //clear first 16 nibbles of RAM
; for(b=0;);
0000 LBL 00h
0002 1020000:
; {
*b=0;
0002 LAX 00h
      incb; //increment bl and skip next instruction if 0
6 0 0 3 EXCI 00h
; continue;
0004 TR L020002
break:
0005 TR L020001
0006 L020002
0006 TR L020000
0007 1020001
; }
0007 END
```

Listing 3—This C-like structured assembler initializes RAM to zero, reads all ports, saves the values in RAM, and sets up a timer interrupt. The timer interrupt polls memory and the ports and lights an LED if any of the ports or memory has changed.

```
Initialize:
org 0; // clear all 128 nibbles of RAM
for(b=0;bm<8 bm++){ // b is 8-bit register, bm is 4 bit
  for(;;){
    *b=0;
    incb; // increments bl reg, skips next inst
          if bl=0xf

    continue
    break;

// read all ports and store in first five memory locations
bm=0;
inp 1; // read P1
*b=a; // store P1 data to RAM
bl++; // increment RAM address
inp 2; // read P2
```

(continued)

generates the appropriate assembly instructions. At this point, you can hand optimize the code then assemble it as you would a “normal” assembly source listing.

Note that the C-like program requires 7 bytes instead of 5. However, as code increases in size and complexity, the difference in the program size is minimal, especially after some hand optimization.

In some cases, the C-like compiler can produce more efficient code, especially when working with pointers and switch-case statements. The big plus, though, is that C-like code is much easier to read and debug, as the next example illustrates.

REAL-WORLD TEST EXAMPLE

When you use a microcontroller in many HVAC applications, the device is susceptible to some pretty harsh environments. Before a part can be used in a gas ignitor, for example, it must be subjected to a number of brutal tests including the “shower of sparks.”

This test entails placing a Frankenstein-like spark-gap generator in close proximity to the micro while it runs a test program. Your job: hope and pray it doesn't fry.

Listing 3 is C-like test program written for a '5K5 microcontroller. (It's basically the same as a '5K6 but has 128 nibbles of RAM and 2 KB of ROM.)

It sets up an interrupt that occurs every 10 ms to read the I/O ports and scans RAM for any changes that might have occurred between interrupts. If anything has changed, an appropriate LED is activated.

Originally, this code is written entirely in assembly. When converted to structured C-like assembler, the code size differed by only two bytes.

OTHER APPLICATIONS

The ADC on the '5K6 has 10 bits of resolution with a 30.5- μ s conversion time (using a 4-MHz clock). When you consider that this part costs close to a buck (for a masked version in large quantities), you have an economical replacement for many standard lo-bit ADCs.

But since it's a microcontroller, you could add some enhancements such as

Listing 3-continued

```

*b=a; // store P2 data to RAM
bl++; // increment RAM address
inp 3; // read P3
*b=a; // store P3 data to RAM
bl++; // increment RAM address
inp 4; // read P4
*b=a; // store P4 data to RAM
bl++; // increment RAM address
inp 5; // read P5
*b=a; // store P5 data to RAM

outp 0,0x00; // turn LEDs off
; // timer initialize routine ;
dr; // divider clear
outp 0x0b,0x0b1; // modulo register set
bl=0x0a; // timer counter set
out;
outp 0x0c 0x9; // timer mode control register set
// divide input clock by 128
outp 0x0e 0x4; // interrupts every 10 ms with 2-MHz clock
ime=1; // interrupt enable

LOOP
for b=0x10;bm<8;bm++){ // write bit pattern to locations 16-128
for(;){
*b=5; // use 0101 bit pattern
incb; // increments bl register, skips next inst
if bl=0xf

continue
break;

for (b=0x10;bm<8;bm++){ // read RAM and check that it did not change
for(;){
if (*b!=5) goto ERROR1;
incb; // increments bl reg, skips next inst if bl=0xf
continue
break;

goto LOOP;

org 0x0204; // TSR from timer interrupt
bm=0
inp 1; // read P1
if(a==*b) goto ERROR
bl++ // increment RAM address
inp 2; // read P2
if(a!=*b) goto ERROR:
bl++; // increment RAM address
inp 3; // read P3
if(a!=*b) goto ERROR:
bl++; // increment RAM address
inp 4; // read P4
if(a!=*b) goto ERROR:
bl++; // i n c r e m e n t R A M a d d r e s s
inp 5; // read P5
if(a!=*b) goto ERROR:

//set ok LED
outp 0,0x0;
outp 0,0x8; // pulse LED measure it with oscilloscope
rtni;

ERROR: // this is port read error
outp 0,0x01; // light port error LED
goto E.LOOP; ERROR1: // memory error
outp 0 0x2; // light RAM error LED

ELOOP:
got0 ELOOP:

```

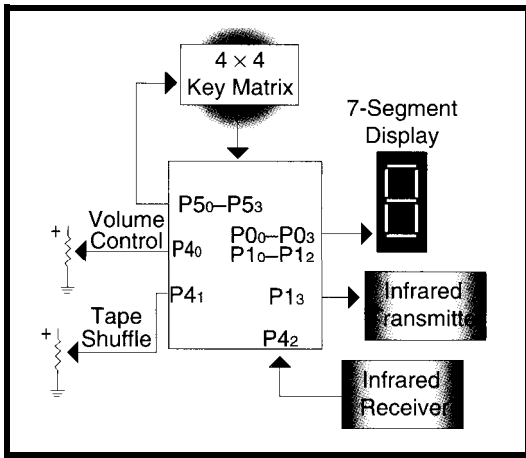


Figure 3—This VCR remote control uses the '5K6.

automatic gain control, a VOX switch, compression, or some simple digital filtering.

Figure 2 shows a schematic of a '5K6 used as a speech-sampling circuit that interfaces to a Sharp LH77790 ARM-based RISC microcontroller. In this case, the ARM memory-management unit uses one of its six chip selects to read the A/D values via a couple of 74HC244 buffers.

The '5K6 continuously runs the ADC at full speed and places the result on 10 output pins. The ARM processor reads the port at the desired sample rate, which for speech is typically 125 μ s, well under the '5K6's 30.5- μ s conversion time.

The bidirectional synchronous serial port of the '5K6 could also be used as a means to transfer commands and data to and from the host microcomputer.

Being CMOS, the '5K6 can be shut down entirely or run at a slow clock speed. For example, the '5K6 can be part of a hand-held instrument that operates as a smart peripheral for a more advanced microcontroller, such as an ARM or H Series RISC processor.

In this example, the '5K6 performs several functions. It can serve as a lithium-ion battery charger, power management, a real-time clock, supervisory control, a low-speed ADC, and nonvolatile storage.

Communication to the host microcontroller is via the serial port that is clocked by the host and interrupts the '5K6 during serial transmission. When running from a 32.768-kHz crystal, the part consumes about 20 μ A. The ADC is set to compare mode so that if the voltage drops below a programmed

threshold, some immediate action can take place.

And finally, Figure 3 shows how you might use the part as an infrared transmitter with an LED display and keyboard.

In this case, the '5K6 is placed in the Stop mode until a key is pressed that wakes up the processor.

STILL IN BUSINESS

Yes, the 4-bit microcontroller is here to stay. Although the market share of the 4-biters is expected to steadily decline over the years, it will remain with us well into the 21st century.

Sharp and other manufacturers are committed to supporting 4-bit microcontrollers until nobody wants them anymore. But, that scenario seems unlikely—a least for a while. In fact, Sharp is planning new parts that are even faster, cheaper, and better.

Hmmm... Maybe if I connect eight of them in parallel, I could run Unix in my watch. ☹

Brad Stewart is a senior applications engineer at Sharp Electronics. He has over 20 years' experience in analog and digital design engineering and application support, as well as in marketing and sales. You may reach Brad at (360) 834-8930 or bstewart@sharpsec.com.

SOURCE

SM5K6

Sharp Electronics Corp.
5700 NW Pacific Rim Blvd., Ste. 20
Camas, WA 98607
(360) 834-2500
Fax: (360) 8348903
www.sharpmeg.com

IRS

413 Very Useful
414 Moderately Useful
415 Not Useful

BIG THINGS COME





Starting at \$99

Micromint's Domino-52 microcontroller is a "supercomputer" in less than 0.75 cubic inches. We've packed the most essential elements into one tiny package. Domino is a plug-and-go module, just attach +5V and a terminal or network. A simple keyed sequence saves an autostarting program in nonvolatile memory.

SPECIAL FEATURES

- 80C52 with ROM-resident, full floating-point BASIC
- 32K bytes SRAM and 32K bytes EEPROM
- Two PWM outputs, I²C bus
- Serial I/O: (up to 19,200 bps) RS-422, RS-485 & RS-232A
- Two interrupts and three timers
- Parallel I/O: 12 bits, 3 shared with ADC and I²C
- Power: +5V @ 15 mA;
- Size: 1.75" x 1.062" x 0.4" potted
- A/D-converter: 2 channels, 12 bits, 10k samples/sec.
- Connections: via 2 x 10, 0.1" dual-row header
- -20°C to 75°C operating temperature
- Industrial temperature available



4 Park Street • Vernon, CT 06066

Call 1-800-635-3355

(860) 871-6170 • Fax (860) 872-2204

FEATURE ARTICLE

Daniel Patten & Michael Miller

A Universal IR Remote-Control Receiver

Daniel and Michael built a universal remote that offers manufacturer and component types in a learned set using an 8051-based decoder. Find out how flexible and inexpensive their solution is.



Have you ever finished the ultimate project and after giving the big demo to your neighbor, he asks, "Where's the remote control?" You soon realize that part of you agrees. Remote control would be nice. So, you shift all other projects to the bottom of the to-do pile.

Unfortunately, not long after starting down that road of IR remote-control research, you find that there are no IR standard receiver chipsets/decoders. You're not alone. A lot of people have had this experience.

In this article, we explain IR remote-control basics and define a system in

which most projects can be controlled via IR remote control. Our solution, the URCR (Universal IR Remote-Control Receiver), easily decodes many standard remote-control signals.

The URCR is based on the Intel 8051 microprocessor. Although it has a microprocessor, you can successfully integrate the URCR into most designs without any microprocessor or programming experience.

STANDARDS

Our first projects destined for remote control were an electronic crossover and preamplifier. But, we wanted to use remote control in many different projects. The time had come for some serious research.

We disassembled every available remote control (-20 total) and found that the majority used an NEC transmitter/keyboard encoder chip. The most widely used chips came from the D6xxx series (e.g., the D6124, D6120, D6122, and D6600).

NEC makes several different series of remote-control encoders but no decoders. Apparently, most manufacturers have to write code for a microprocessor in each product they make.

While it may not be a standard, NEC recommends a particular format for the serial datastream. NEC registers companies and allocates manufacturers a unique ID, known as the customer code, which is included in the serial datastream.

Most of the tested remotes supported the NEC standard. And, since

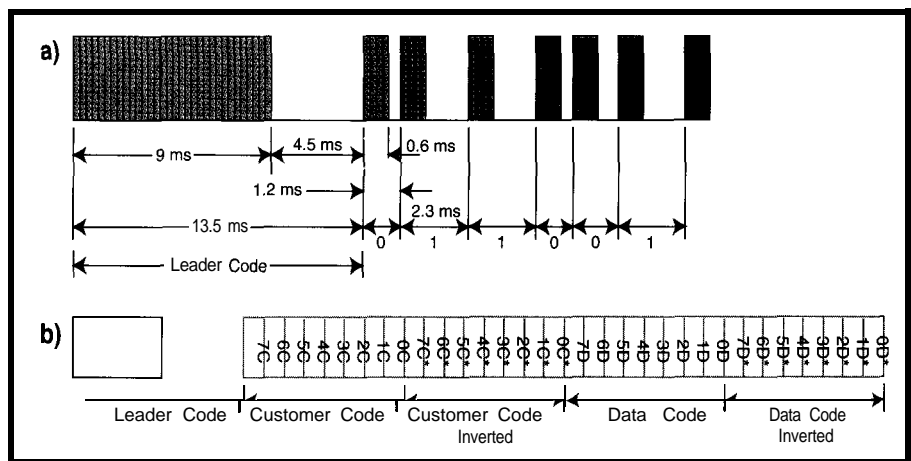


Figure 1 --The NEC-style data format uses a pulse-position-modulation scheme to encode the data. **a**—The interpretation of a PPM code is based on the time between pulses. **b**—The data bits marked with *denote bits that are used to calculate check sums. This error checking helps ensure that the data is decoded correctly.

this standard uses a fixed number of bits in the serial stream and we had the databook, we decided to use it.

We found that the measured data correlated very closely to NEC's databook figures. Figure 1a illustrates the header (i.e., beginning) of the IR data-stream.

The NEC standard uses pulse position modulation (PPM) with a data carrier frequency of 38 kHz. The header is composed of a 9-ms high pulse followed by a 4.5-ms low pulse. In addition, a 32-bit datastream follows the header with no end-of-header or stop bits.

PPM might appear confusing at first, but it's relatively straightforward. Look at the gaps between data pulses. A long gap represents a data value of 1, and a short gap represents 0.

Figure 1b illustrates the signal composition. Of the 32 data bits, only 16 carry command information. The other 16 are for error checking.

This high level of data redundancy, if the decoder takes advantage of it, can result in a high degree of accuracy in decoding transmitted commands. We knew what the IR transmitter was sending. We just had to decode it!

MICROPROCESSOR IMPLEMENTATION

Our first decision: Which microprocessor to use? We were familiar with the Intel 8051, so we went with that. How much horsepower? Not very much, as a few quick calculations show.

A standard clock frequency for an 8051 is 11.0592 MHz, with the microprocessor taking 12 clock cycles to complete a standard instruction. Therefore, an 8051 executes an instruction about every microsecond (nearly 1 MIPS!):

$$\left(\frac{1}{11.0592\text{MHz}}\right)(12) = 1.085\mu\text{s}$$

If the gap between pulses forming a 0 in the IR serial datastream is 0.6 ms, the 8051 can run 553 instructions while waiting for the next pulse:

$$\frac{0.0006}{1.08 \times 10^{-6}} = 553$$

Even with the most bloated code, an 8051 should have enough horsepower to decode the serial datastream.

in gated mode and the interrupt in edge mode.

We used this method because a timer will run (count) as long as the interrupt pin is deasserted. That is, the timer counts the gaps between pulses, and the code is interrupted at the start of every high pulse.

This way, on each interrupt, an ISR (Interrupt Service Routine) can determine whether a long or short gap has occurred, reset the counter, and wait for the next interrupt.

To accommodate different types of remotes, we created a one cell and a zero cell. A cell is a time measurement the 8051 uses to determine if data is a 1 or a 0. If a pulse is of a width that falls in the one cell, the data is interpreted as a 1.

The 8051's timer increments every 1.085 μs . The databook indicates that the zero gap should ideally be 0.6 ms. A 0.1-ms margin accommodates timing errors, resulting in a cell of 0.5-0.7 ms. For a one gap, the cell margin is 1.2 ms.

For an ideal zero gap, the counter increments 553 times:

$$\frac{0.6\text{ms}}{1.085\mu\text{s}} = 553$$

The count margin is 461-645 times. For a one gap, the ideal count is 1014, and the margin ranges from 922 to 1106.

These count results are divided by 256. Why? Because the 8051 timer has 16-bit resolution (i.e., it can increment 65,536 times before wrapping back to zero). This timer/counter is

implemented with a low and high byte. The low byte of the timer can count to 256, and the high byte of the timer is only incremented every time the low-byte timer reaches 256.

In trying to keep the code easy, we ignore what the low byte of the timer is doing. This way, we only have to

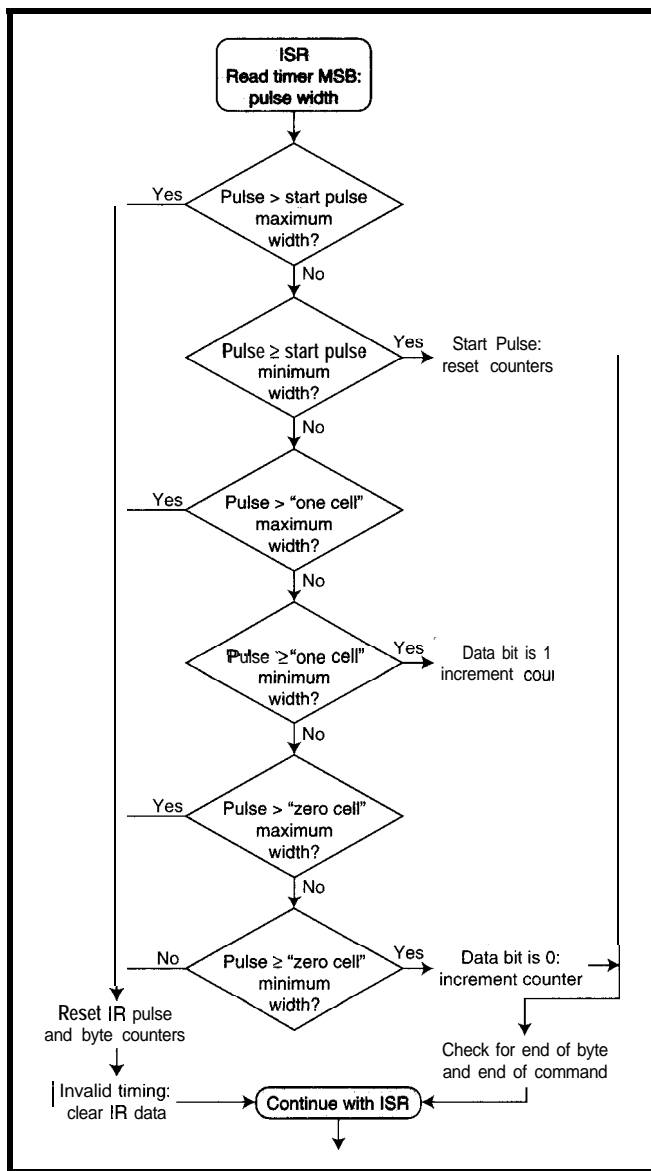


Figure 2—The basic pulse-position-modulation decoding algorithm uses time cells to determine the value of each data bit.

HARDWARE

Our receiver of choice—the Sharp IR module—is part of the GP1Uxx and IS1Uxx families. It's the perfect match for the 8051.

To set up the hardware, we connected the IR module to an 8051 interrupt pin and put the associated timer

compare one byte (the high byte) to see if a gap is a zero or a one.

The high byte provides enough resolution ($256 \times 1.085 \mu\text{s} = 277.8 \mu\text{s}$) for this application. In practice, this technique works quite well.

A zero gap is represented by the high-byte timer being either a 1, 2, or 3. A one gap is represented by either a 4, 5, or 6.

FIRMWARE

We wrote the code for the 8051 in assembly. Figure 2 is a flowchart for the cell-detection algorithm we implemented in an ISR for the 8051.

Each time an IR pulse is presented at the interrupt pin of the 8051, the processor executes this routine to determine what serial data is being transmitted.

A corresponding code fragment of the zero- or one-cell selection is provided in Listing 1.

URCR

By adding a small nonvolatile memory and writing some more involved code, we believe we came up with a novel solution. The URCR is an inexpensive and easy way to add IR remote control to an existing design.

We designed it with a high degree of flexibility for operation in systems with or without a microprocessor. In fact, you can easily use it without knowing that the URCR is really a microprocessor.

Our goals were to:

- design a stand-alone microprocessor for use as a generic IR remote-control decoder
- make the decoder easy to paste into any design
- make the decoder inexpensive
- design multiple capabilities so the decoder can match any target system's capabilities
- make the decoder learning, so it can accommodate most remotes

To achieve the flexibility for implementation in a variety of systems, the URCR has six operation modes. They are pin programmed on the microprocessor, which puts itself into one of the modes on powerup (see Table 1).

Listing 1—A few simple checks of the timer are all that is needed to determine info which cell the data bit should be categorized. Erroneous data is detected when the bit does not match the definition of a "one cell" or a "zero cell."

```
getPulseISR:
    push PSW
    push ACC
    clr EA ;disable interrupts

;Timer is incremented every 1.085 μs. TH1 resolution is 256 *
;1.085 μs (i.e., 277.8 μs). TH1 value should be close enough to
;determine start pulse width. Start pulse width is 4.5 ms, which
;is 4147 timer ticks. TH1 would have been hit 16 times.
;start pulse = 4.5 ms
;long pulse = 1.58 ms
;short pulse = 0.46 ms
;window = 3.889 < start pulse < 5.0 ms (i.e., 14 < TH1 < 18)
;window is 1.1 < long pulse < 1.667 ms (i.e., 4 < TH1 < 6)
;window is 0.277 < short pulse < 0.833 ms (i.e., 1 < TH1 < 3)

;detect start pulse
clr C ;clear carry bit for comparison
mov A,#12h ;look at timer high byte
subb A,TH1 ;if TH1 > 18, set CF
;this could be the end pulse! check for it!
jc checkEndPulse ;pulse width > threshold

clr C ;clear carry bit for comparison
mov A,#0dh ;look at timer high byte
subb A,TH1 ;if TH1 > 14, set CF
jc startPulse ;pulse is a start pulse

;detect long pulse
clr C ;clear carry bit for comparison
mov A,#6h ;look at timer high byte
subb A,TH1 ;if TH1 > 6, set CF
jc badPulseWidth ;pulse width > threshold

clr C ;clear carry bit for comparison
mov A,#3h ;look at timer high byte
subb A,TH1 ;if TH1 > 4, set CF
jc longPulse ;pulse is a long pulse

;detect short pulse
clr C ;clear carry bit for comparison
mov A,#2h ;look at timer high byte
subb A,TH1 ;if TH1 > 3, set CF
jc badPulseWidth ;pulse width > threshold

clr C ;clear carry bit for comparison
mov A,#00h ;look at timer high byte
subb A,TH1 ;if TH1 > 1, set CF
jc shortPulse ;pulse is a short pulse

;detect glitch pulse
jmp badPulseWidth
```

MORE HARDWARE

As Figure 3 shows, the URCR doesn't need many extra components to run. Basic hardware is sufficient for Modes 0, 1, and 2.

The five required component sections are customer-code input, reset circuit, mode selection, IR receiver, and oscillator circuit. Other optional connections can be added for increased functionality.

Recall that the NEC standard specifies an 8-bit customer code. This code is fed into the URCR by attaching eight switches (a DIP switch can be used) to pins CCO-CC7. The customer

code can also be in the form of an 8-bit latch loaded by another microprocessor.

Connecting three switches to pins MODE0-MODE2 provides a way to select the URCR's operating mode. The switch positions are only valid at powerup. The URCR system ignores subsequent changes in their values. The switch positions are shown in Table 2.

For the 8051 microprocessor to start properly, the Reset pin must be held high at powerup. This task is accomplished by the 10-kΩ resistor and 10-μF capacitor.

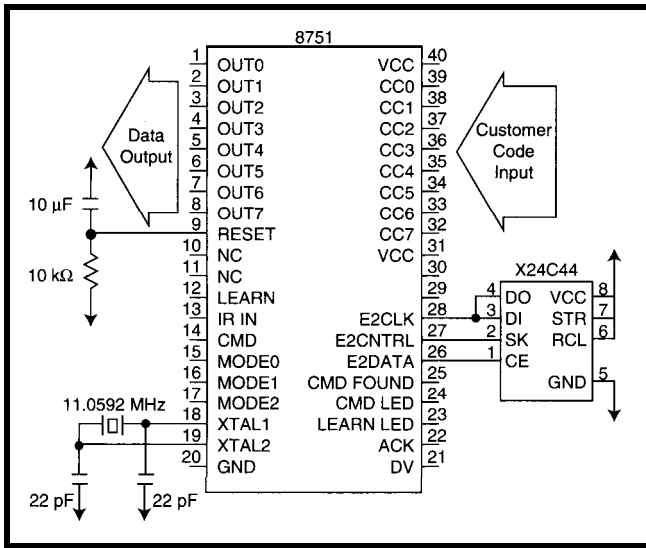


Figure 3—This simple schematic shows how easily the URCR system can be implemented. It requires only six external components.

On powerup, the capacitor charges and acts as a low-resistance connection to the 5-V supply. When the capacitor is done charging, it acts as an open circuit to the 5-V supply.

The IR receiver's output is connected to pin 13 of the microprocessor. The receiver can be somewhat susceptible to power-supply noise, so a 1-µF tantalum capacitor connected directly across its power pins is advised.

The 11.0592MHz crystal and two 22-pF capacitors form an oscillator for microprocessor operation.

To support Modes 3, 4, and 5, you need an EEPROM. We used a Xicor X24C44, which can store 256 bits of nonvolatile information.

The X24C44's memory is organized as a 16 x 16 matrix. Since 16 commands can be stored, there can be 8 bits of command and 8 bits of customer code for each command. The X24C44 has a very simple interface and only requires three connections to the microprocessor.

CODE FUNCTIONALITY

The URCR code comprises five major components—initialization, the pulse ISR, the operating-mode loops, the learning routines, and the EEPROM search routine.

After powerup, the processor initializes itself and determines its current operating mode. The operating mode is read from three switches at bootup, and changes in switch settings do not affect the processor until power is cycled.

Once the switch is read, the processor calls the appropriate run-time routine. The processor remains in the operating-mode routine until power is turned off.

In the most basic operating modes, the processor may only poll status bits to see if a new command was received and export the command data. In more complex modes, the microprocessor polls for new commands, searches the EEPROM to match incoming commands to learned ones, monitors the Learn and Command buttons to arbitrate the command-learning process, stores new commands to EEPROM, and exports valid command data with a Data Valid/Acknowledge protocol.

The processor enters the ISR when the interrupt line is asserted by an IR pulse. The timer is gated off during ISR execution, and it is reset and gated on

at the end of the ISR. This way, the 8051 can measure the time between pulses and determine the data value encoded via a PPM scheme.

In addition to the PPM decoding, the ISR is responsible for grouping the pulses into bytes, calculating checksums to verify data integrity, and updating the new command flag to alert the operating loop that a new command was received.

In the operating-mode loop, the processor monitors the new command flag indicating that a new IR command was successfully received and processed by the ISR. Once the processor knows it has new data to act on, it exports the data as required for each operating mode. For learning modes, arbitrating the command-learning process is also performed in this loop.

The URCR's learning capabilities provide a powerful tool for implementing custom IR designs. The flexibility of using any universal remote control to dynamically program the URCR makes its programming simple. Figure 4 outlines the learning process.

You enter the learning mode by pressing the Learn button, and the Learn LED is activated. Once in the loop, a second Learn button press causes the processor to leave this mode. Therefore, pressing the Learn button twice in a row erases the command memory.

In the learning loop, a Command-button press puts the processor in the command-button loop. The Command LED lights to identify this mode.

Mode	Description	Notes
0	Basic	Commands passed through, Customer code from switches, Commands are static and replaced by next command
1	Microprocessor Interface Single Byte	Commands available, DV/ACK handshaking for the microprocessor interface
2	Microprocessor interface Double Byte	Commands available, Customer code available, DV/ACK handshaking for the microprocessor interface
3	Basic Learning	Learning mode supported with addition of serial interface EEPROM. Commands available. DV/ACK handshaking for the microprocessor interface
4	Microprocessor Interface Learning Single Byte	Learning mode supported with addition of serial interface EEPROM, Commands available, DV/ACK handshaking for the microprocessor interface
5	Microprocessor Interface Learning Double Byte	Learning mode supported with addition of serial interface EEPROM, Commands available, Customer code available, DV/ACK handshaking for the microprocessor interface

Table 1 --The URCR can operate in one of these six modes. Mode 0 is the most basic implementation and Mode 5 is the most involved.

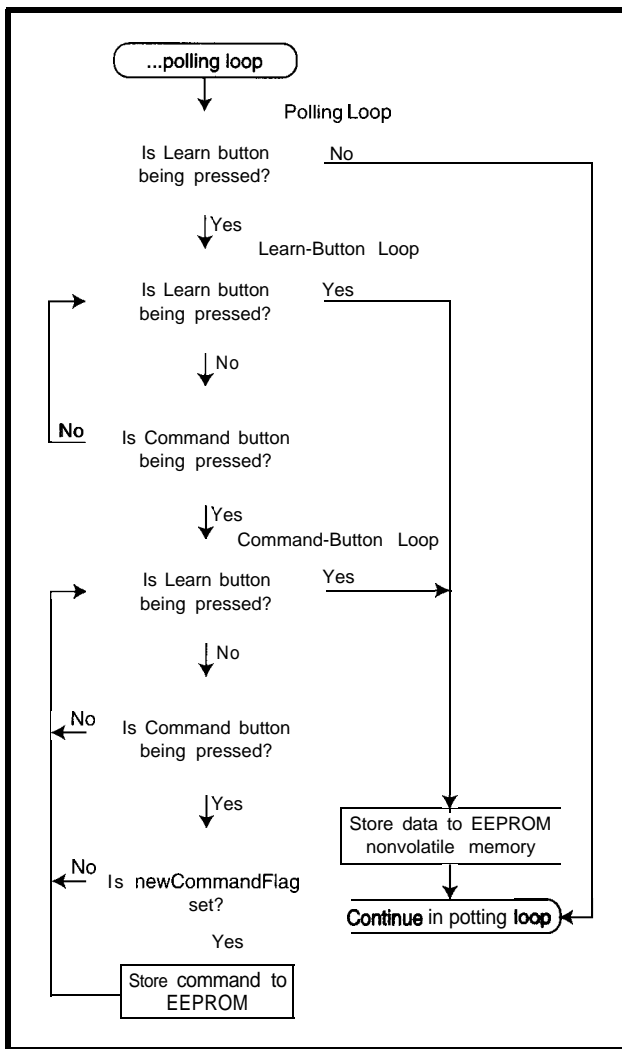


Figure 4—The learning loop is controlled by two buttons: Learn and Cmd. Each button press steps the URCCR through this algorithm.

The URCCR's learning scheme enables the user to mix manufacturers and component types in the learned set. For example, if you use the URCCR in an audio system, a power amplifier can be controlled by the Sony TV codes and an electronic crossover can be controlled by the Pioneer CD codes.

MODES OF OPERATION

In Mode 0, the user sets the desired customer code via switches on microprocessor pins CCO-CC7. Any command that matches the customer code and passes error checking is output on pins OUTO-OUT7 (see Figure 5a).

The command stays on these pins until another valid command is received. In a simple system, devices (e.g., relays) can be activated on a specific command and stay active until another command deselected it.

Mode 1 supports a microprocessor interface, so a system already with a microprocessor can easily interface to the URCCR. Customer codes are selected via switches on microprocessor pins CCO-CC7. Any command that matches the customer code and passes error checking is output on pins OUTO-OUT7 (see Figure 5b).

The command stays on these pins until the ACK (Acknowledge) pin on the URCCR is asserted low. The URCCR notifies an external microprocessor that a valid command was received by asserting the DV (Data Valid) pin low.

A single command stays on pins OUTO-OUT7, and the URCCR ignores further commands until the external microprocessor asserts ACK, notifying the URCCR that the command byte was read.

Mode 2 is nearly the same as Mode 1, except the customer code is also output on pins OUTO-OUT7. The customer code is output first, followed by the command. Figure 5c illustrates

The URCCR then waits for either a Learn-button press to exit the learning mode or a new command to be received and flagged by the ISR. Additional Command-button presses have no effect while the URCCR waits for a new command.

On entering the learning mode, the EEPROM address pointer is reset to 0x0000 and incremented with each new command. When the ISR receives a valid command, the Command LED turns off and the command is assigned to the next available EEPROM memory location.

The URCCR exits the learning mode if all available EEPROM memory locations have been programmed. The Learn LED turns off to alert the user that the URCCR has reached full learning capacity.

The operating modes that use command learning rely on the EEPROM search routine to determine if the

current command is already learned. If it is, the URCCR executes the appropriate data movement. Otherwise, it ignores the command.

The URCCR's X24C44 serial EEPROM stores the learned-command set. The X24C44 is divided into volatile (SRAM) and nonvolatile (EEPROM) sections.

The SRAM provides unlimited reads or writes and is used for all X24C44 accesses except for storing data to non-

volatile memory. The EEPROM can withstand a minimum of 1,000,000 store operations.

The EEPROM is accessed to load the SRAM with valid data at powerup and to store the learned set of commands. When command data is stored in the EEPROM, it's stored with the most significant byte (i.e., the customer code) first.

When a command matches a learned command, the Command Found LED flashes to indicate a successfully completed memory search.

	Mode	Mode2	Model	Mode0	Function
I	0	off	Off	Off	Basic
I	1	Off	Off	On	Microprocessor interface, single byte
	2	Off	On	Off	Microprocessor interface, double byte
	3	Off	On	On	Basic, learning
	4	On	Off	Off	Microprocessor interface, learning, single byte
	5	On	Off	On	Microprocessor interface, learning, double byte

Table 2—The URCCR can be used in one of these six modes. To select a mode, simply set the switches. The URCCR will enter the selected mode when power is applied.

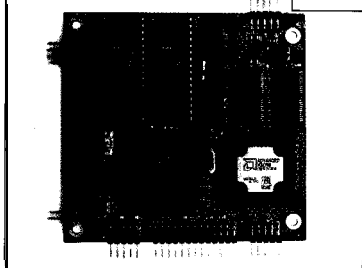


Sets The Pace

Am5x86™ 133 MHz
PC/104 cpuModules Exceed
Pentium-75 Performance



\$945

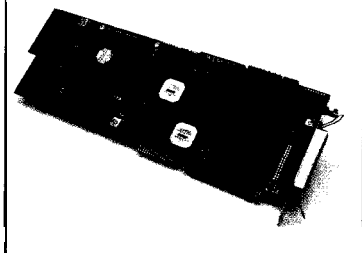


The CMV586DX133 offers versatile embedded functionality

Embedded PC and DSP Drive RTD's Intelligent Data Acquisition and Control Board



\$3985



The IDAC5210 with PC1104 extension bus, 500 KHz analog and digital front-end is powered by on-board Am5x86 133MHz CPU and TMS320C50 DSP

Our PC/104 and ISA Bus product lines feature Analog and Digital I/O, CPU, DSP, Shared Memory, SVGA, PCMCIA, CAN Bus and Intelligent GPS



Real Time Devices USA

200 Innovation Boulevard . P.O. Box 906
State College, PA 16804-0906 USA

Tel: 1 (814) 234-8087 . Fax: 1 (814) 234-5218
FaxBack®: 1 (814) 235-1260

www.rtdusa.com . E-Mail: sales@rtdusa.com

RTD Europa

Budapest, Hungary
Fax: 36-1-326-6737

RTD Scandinavia

Helsinki, Finland
Fax: 356-9-346-4539

RTD is a founder of the PC/104 Consortium and the world's leading supplier of intelligent ISA DAS interfaces.

#123

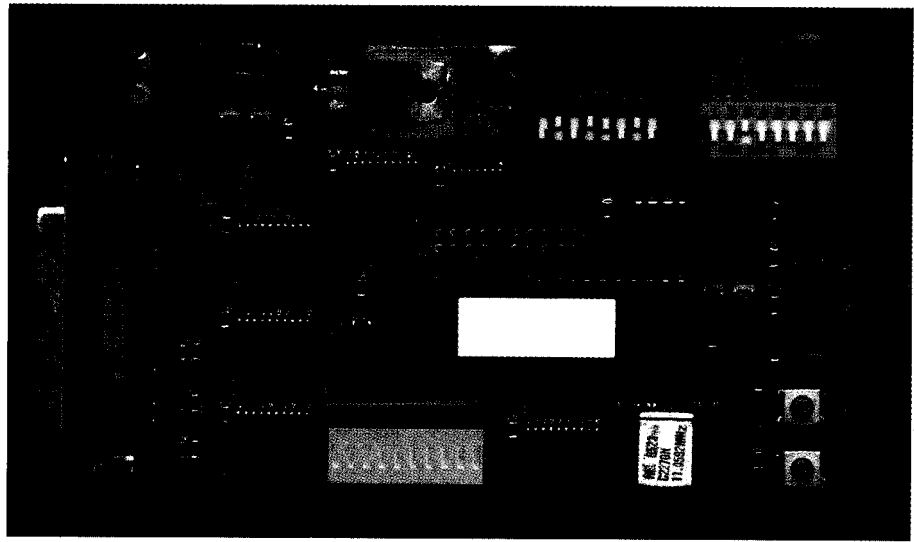


Photo 1 --The URCCR Evaluation Board makes it easy to experiment with the IR remote control. The evaluation board is capable of running in all six operating modes. It also has a PC-compatible parallel port interface.

the DV/ACK procedure for reading the two bytes of information.

Mode 3 incorporates the URCCR's learning capabilities. It has the same data-movement scheme as mode 0, but only commands matching the learned commands generate output data.

In addition, the output data is not the IR command. Rather, it's the EEPROM address of the learned command that matches the current command.

Mode 4 incorporates the microprocessor interface of Mode 1 with the learning capabilities of Mode 3. Like Mode 3, the output data is the EEPROM address of the learned command.

Mode 5 combines the data-output scheme of Mode 2 with the learning capabilities of Mode 3. As with Mode 2, the output data is the learned command's customer and command codes.

URCCR EVALUATION BOARD

We developed the multipurpose evaluation board shown in Photo 1 to explore the URCCR's capabilities. All functions of the URCCR can be tested, including the microprocessor interface, via connection to a personal computer through the parallel port.

As you see in Figure 6, the URCCR eval board supports all six modes of operation. The board is based around U1, an Intel 87C51 microprocessor, which contains its own program memory onboard.

On bootup, the microprocessor reads the status of pins P3.5, P3.6, and P3.7, which are driven by S2.1, S2.2, and S2.3. Turning these three switches on selects Mode 0.

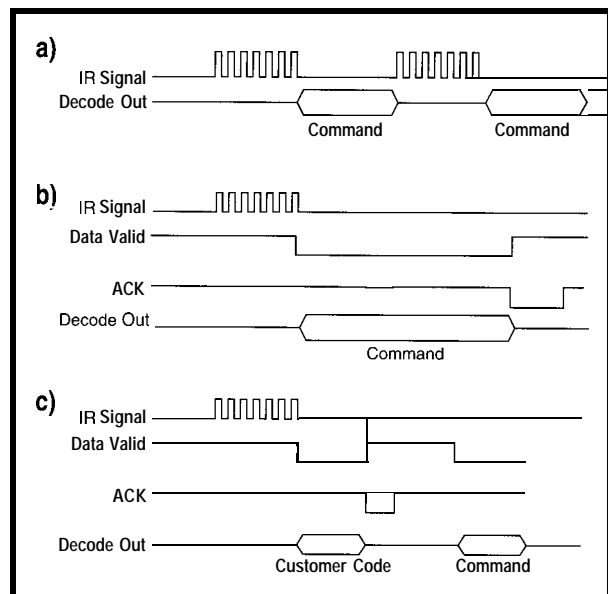


Figure 5--The URCCR has three methods of exporting data. a--The simplest method sends data out as soon as the entire byte is received. b--The Data Valid/ACK scheme can send out one byte or, as you can see in (c), two bytes.

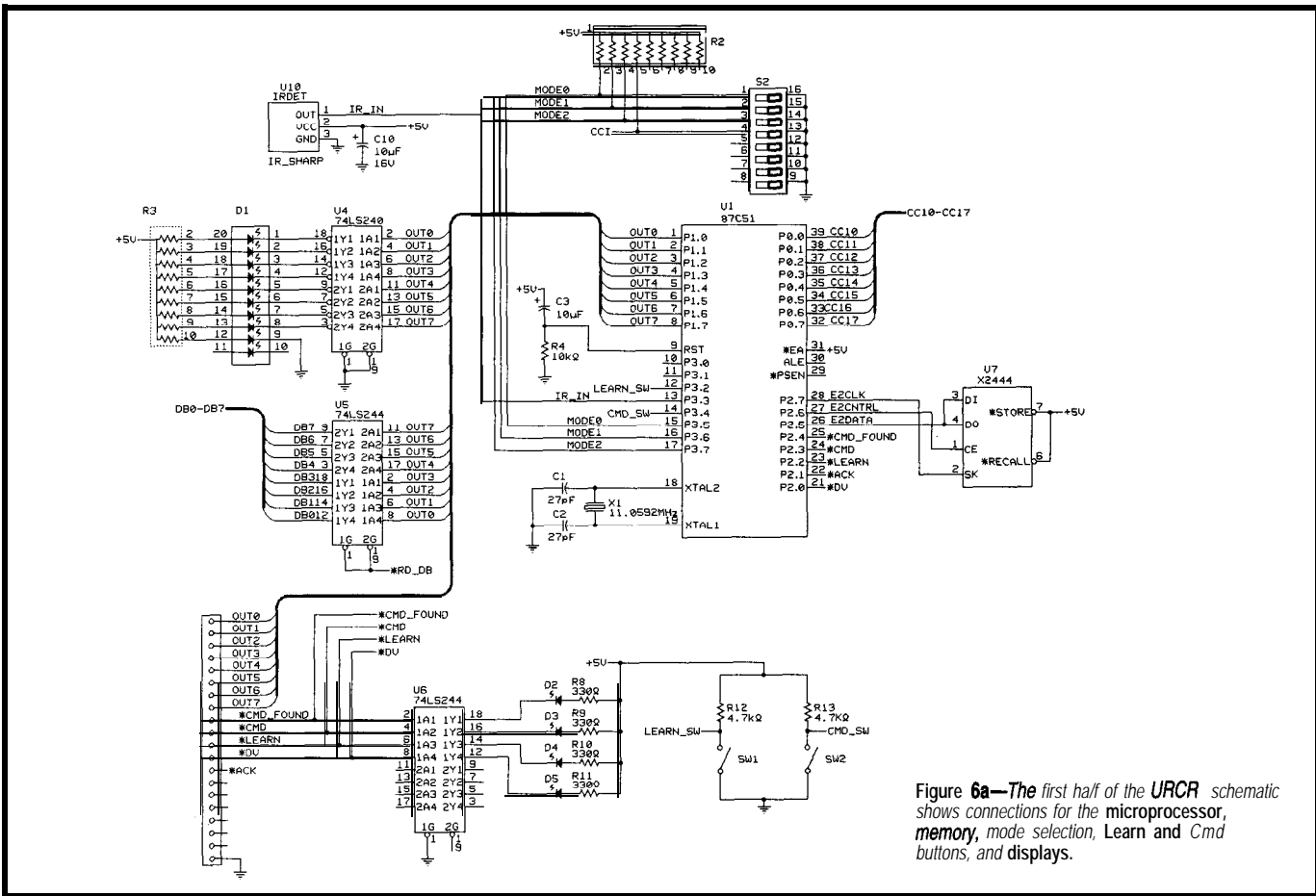
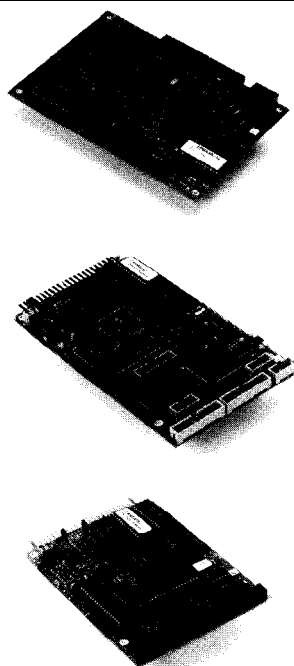


Figure 6a—The first half of the URCR schematic shows connections for the microprocessor, memory, mode selection, Learn and Cmd buttons, and displays.



It All Adds Up!

- + A/D Inputs with 12-bit accuracy
- + Analog outputs
- + Intel and Zilog CPUs
- + Quadrature encoder inputs
- + Buffered RS-232/485 serial ports
- + Keypad & LCD display interface
- + Program using a PC
- + 512K program, 512K data memory
- + 5V only
- + Program in C, BASIC, or Assembly
- + Floating point math
- + Starting from \$195

- ✓ Easier, faster development
- ✓ More integrated design
- ✓ More \$\$ savings

With our line of embedded controllers and accessories, everything adds up to make your job a success!

For more information **303-690-1588** & FREE catalog, call, email or fax: **303-690-1875**
REMTE " **PROCESSING**
 Web Site: www.remotep.com info@remotep.com The embedded control company

The customer code is input on Port 0 and can be set by the PC or switch S 1. If you want to set customer codes by the switches, S2.4 is turned on.

The decoded IR commands are output on P1.0–P1.7 and can be viewed on the LED pack D1 or read by the PC. RD_DB must be taken low to read the data. These signals are inverted in the PC. SW1 and SW2 are used to programming the URCR in Modes 3, 4, and 5. LEDs D2, D3, and D4 display current status of the programming cycle.

The URCR eval board interfaces to the PC via the parallel port. You need a bidirectional parallel port, but most new PCs are bidirectional capable. ECP (Extended Capabilities Port) and EPP (Enhanced Parallel Port) ports are also compatible.

The parallel port interface can set the customer code and read the data bus (decoded IR commands and customer codes). It also monitors the programming cycle and DV, and it controls ACK.

Sample PC parallel port interface code is available for the evaluation

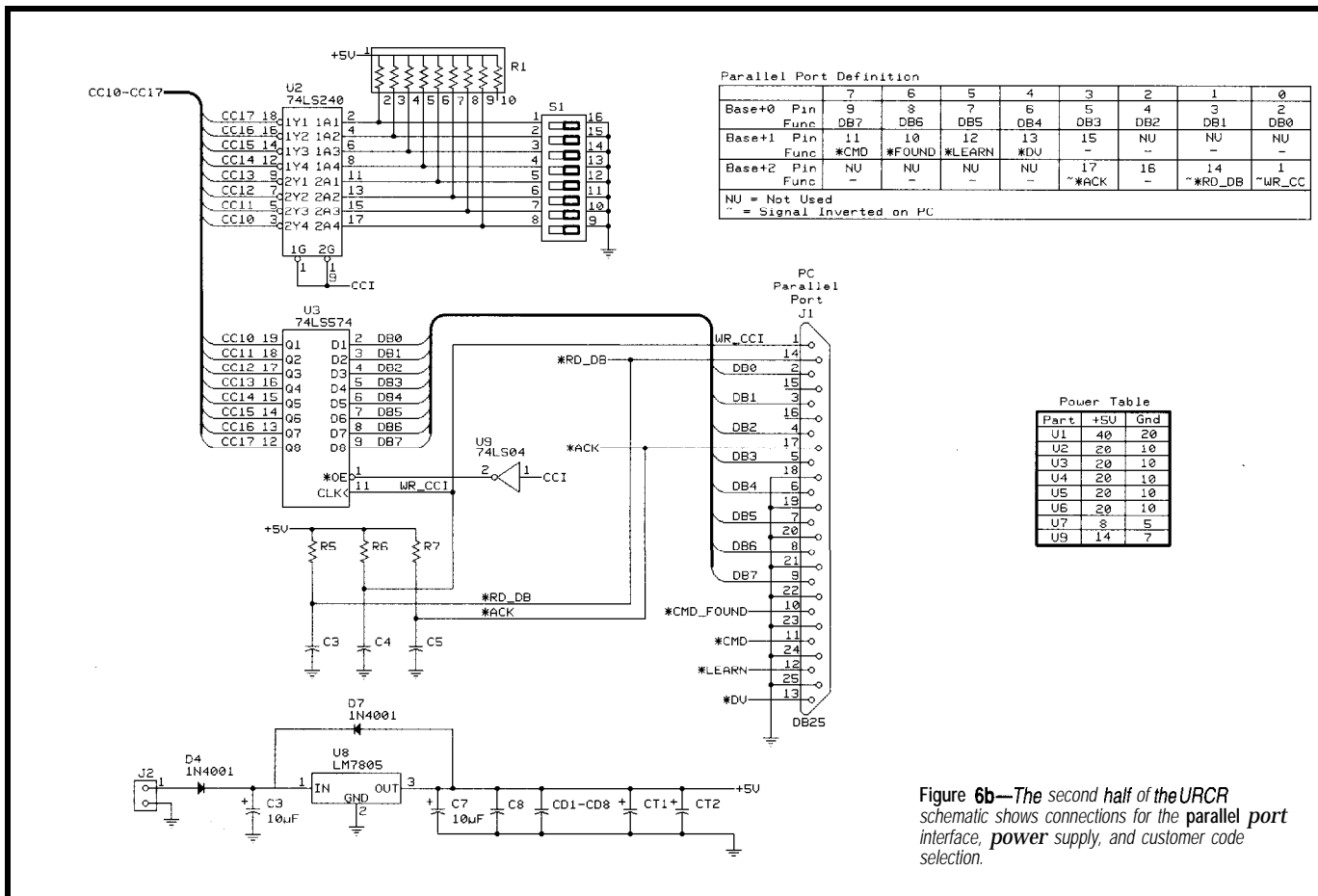


Figure 6b—The second half of the URCR schematic shows connections for the parallel port interface, power supply, and customer code selection.

board. A fully functional menu-driven program is supplied with the URCR eval board, and the PC code is DOS compatible.

FINAL THOUGHTS

We hope this article solves some of the mysteries of IR remote control. With the basics of an IR remote-control standard, a few parts, and some code, you can realize a system that easily decodes many different types of remote controls. □

Daniel Patten, an electrical engineer at DAS, designs data-acquisition systems and custom computer hardware. As well, he designs high-performance audio equipment and speakers, recently completing a.25W single-ended class A amplifier. You may reach him at dpatten@dasengr.com.

Michael Miller is an electrical engineer at DAS. While he primarily develops firmware and software for custom hardware, he also collaborates on custom hardware design. You may reach him at mmiller@dasengr.com.

SOURCES

X24C44

Xicor
15 11 Buckeye Dr.
Milpitas, CA 95035
(408) 432-8888
Fax: (408) 432-0640
www.xicor.com

87C51

Philips Semiconductors
811 E. Arques Ave.
Sunnyvale, CA 94088-3409
(801) 264-8050
Fax: (708) 296-8556
www.semiconductors.philips.com

IR Module

Sharp
5700 NW Pacific Rim Blvd., Ste. 20
Camas, WA 98607
(360) 834-8700
Fax: (360) 834-8611
www.sharp-usa.com

NEC Electronics, Inc.
2880 Scott Blvd.
Santa Clara, CA 95050-8062
(408) 588-6000

Fax: (408) 588-6130
www.nec.com

87C51, Switches, LED, Crystal
JDR Microdevices
1850 S. 10th St.
San Jose, CA 95112-4108
(408) 494-1400
www.jdr.com

87C51, IR Module, X24C44
Marshall Industries
9320 Telstar Ave.
El Monte, CA 91731-2895
(818) 307-6000
Fax: (818) 307-6187
www.marshall.com

URCR Components, URCR Evaluation Board, Programmed 87C51
Dan Patten
1768 N. 980 W
Orem, UT 84057
(801) 434-7226

IRS

416 Very Useful
417 Moderately Useful
418 Not Useful

68 MicroSeries

74 From the Bench

78 Silicon Update

MICRO SERIES

Joe DiBartolomeo

Standards for Electromagnetic Compliance Testing Testing Houses

Part
4
of
4

There's
more to
EMI
approval

than just understand-
ing the tests—there's
an entire testing
house procedure.
Getting that right
translates into
immediate savings.

a

basic under-
standing of EM1
standards and tests is
important to all designers

of electronic equipment. After all, it doesn't matter how well your equipment performs its task. If it fails EM1 testing, you can't ship.

Most electronics designers know that EM1 tests are de facto design specifications and that knowledge of these tests is essential. Parts 1-3 present the most common emission and immunity standards and tests required by the FCC and European Community.

Understanding EM1 tests and standards is not only necessary for the design of electronic equipment, but it's also important when it comes to taking your equipment to be tested. So, this article deals with the process of taking electronic equipment to EM1 test labs.

Many designers spend hours reviewing the EM1 tests and designing their products to meet them. However, they spend little time selecting a lab or preparing for the tests. But, the selection of the test lab and what's done before, during, and after greatly affect your product's chances of passing.

When I look back on my first lab test, I realize I did many things wrong and was only lucky to get away with it. The first mistake was in how I selected the test lab.

Since I live in metro Toronto, I have some choice about which lab to use. To select a test lab, I started by listing important characteristics—experience, quality of test equipment, flexibility of scheduling, price, recommendations of others, personnel, and so on. I then categorized the factors by their primary and secondary importance.

Next, I determined the most important characteristic in each group. This gave me what I considered the number-one, most important factor in selecting an EM1 testing laboratory—geographical proximity to our manufacturing facility.

At the time, it seemed reasonable. With 4 million people in Toronto, traffic is always bad. And of course, there's always the weather. Toronto has some of the best weather in Canada, but that's not saying much.

All and all, I thought that choosing a lab based on how close it was to the office was a good idea. I got away with this error solely because that lab happens to be excellent and fit my needs perfectly.

On my first trip to the lab, I met a fellow from Pennsylvania. Of course, I asked him why he drove all the way to Canada to perform FCC tests. He said it was because of the quality of the lab. I thought he was nuts.

But, now I understand that finding a lab that fits your needs is crucial and can have a huge impact on the length of time it takes you to get your equipment passed. Here are some thoughts on selecting a testing lab.

RECOMMENDATION OF COLLEAGUES

Let's say that you're in the middle of designing a new product and you plan to get the prototype tested for EMI. Ask your colleagues to recommend a lab.

When you call the lab, they'll give you a test date in about three months. Two months later, you may realize your prototype won't be ready on time and that you need an extra two weeks.

At that point, you may call the test lab and find that they can't fit you in

because they're on a tight schedule. They may not be able to test your product for another three months.

The key point is that your product's stage in its life cycle has a great bearing on the lab you use. In the early stages of a product's life span, there are more likely to be design changes. Then, you want a flexible lab that tests you when you're ready.

- Soldering iron, especially if your product has surface-mount components
- Magnifying Glass
- Various values and sizes of capacitors, resistors, and inductors
- Ferrites, both individual and clamp-on-cable type
- Voltmeter
- EMI probes
- Filter topology and Laplace transform table
- Conductive tape and aluminum foil
- All peripheral cables needed to run your instrument in normal and worst-case modes
- List of possible failure and remedial strategies
- Notebook (Remember to take good notes!)
- Test plan

Figure 1—Here are some things you should be sure to take along to the test lab. Many of these items may seem obvious, but they're often forgotten. Supplement this list as necessary for your particular instrument.

Later, when the product has passed EM1 testing and you're testing for QC purposes only, you need a lab that has fixed schedules. At that point, any delay in your QC testing delays shipping the whole run.

FLEXIBILITY

The flexibility of lab schedules is more important to equipment early in the product's life cycle. But, it could be important at later stages as well.

Let's say you have a mature product you gave a minor retrofit to. When you take it to the lab, you find that the minor retrofit causes major EM1 problems. You then realize that you need an extra day to complete the tests.

Will the lab accommodate you by delaying someone else's tests? Can you work on the instrument at the lab?

Almost every lab understands that you may have to add a capacitor here and there, but what if you have major modifications that take several hours? Can you do them and then continue testing? Will you be charged for the

time that you are at the lab working on your product?

Of course, this begs the question—why not always go to a lab with a flexible schedule? Well, to accommodate you, the lab had to delay someone else. So, don't be surprised if on occasion that same lab delays you.

Whether or not you can live with the odd delay is up to you. I work for a small R&D company, and having a flexible lab is important to me.

EXPERTISE

The expertise of the lab is also very important. Good, experienced test technicians are worth their weight in gold. They've seen a lot of problems and can usually offer good suggestions.

Some labs have several EM1 engineers on staff to help you with every aspect of the EM1 process from design to testing. All these experienced people are usually more expensive. You must decide whether you need that expertise or not.

Once you've chosen a test lab, select the standards and tests you'll seek compliance under. This task should be done in conjunction with the test lab.

Prior to meeting with the lab, make a list of the tests you think your equipment needs to pass. In making the list, anticipate most of the questions the test lab will ask you. This approach enables you to communicate intelligently with the people at the test lab and to determine which tests are appropriate for your equipment.

Keep in mind that the test lab has no idea what your product does. They see hundreds of products a year. If you can explain your product's function with respect to EMI, you're more likely to get the proper test set the first time.

It's fairly common for test labs to add tests once they get a feel for the equipment. Also, prior listing of the tests enables you to gauge your understanding of the EM1 issues and feel comfortable about the final test list.

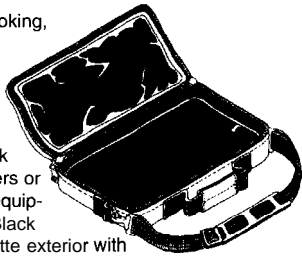
Good communication avoids a situation like I experienced when a

ALL ELECTRONICS

C O R P O R A T I O N

NOTEBOOK COMPUTER CARRYING CASE

Great looking, padded carrying case, suitable for most notebook computers or similar equipment. Black leatherette exterior with separate zippered compartment for papers or accessories. Detachable nylon web shoulder strap. Interior space is 13.5" X 9" X 2.5"



CAT # CSE-12

\$9⁷⁵
each

470 UF, 450 VOLT SNAP-IN CAPACITOR

Nichicon
LGQ2W471 MHSC
1.375" diameter X 2" high.
0.4" lead spacing.



\$4⁵⁰
each

10 for \$40.00

CAT# EC-4745

Low Low Price! 25' BNC-BNC CABLE



25 foot RG58-A/U, 50 ohm co-ax cable with male BNC connectors molded with strain relief on both ends. Ideal for studio, lab or communications use. Inquire for quantity pricing.

\$4⁷⁵
each

CAT# CBL-25

10 for \$45.00

ORDER TOLL FREE

1-800-826-5432

CHARGE ORDERS to Visa, Mastercard,
American Express or Discover

TERMS: NO MINIMUM ORDER. Shipping and handling for the 48 continental U.S.A. \$5.00 per order. All others including AK, HI, PR or Canada must pay full shipping. All orders delivered in CALIFORNIA must include local state sales tax. Quantities Limited NO COD. Prices subject to change without notice.

CALL, WRITE
FAX or E-MAIL
for our FREE

96 Page
CATALOG
Outside the U.S.A.
send \$2.00 postage.

MAIL ORDERS TO:
ALL ELECTRONICS
CORPORATION
P.O. Box 567
Van Nuys, CA 9140
FAX (818)781-2653

E-Mail allcorp@allcorp.com
Internet - <http://www.allcorp.com>

test lab wanted to prescribe AC testing for my battery-powered instrument. I immediately labeled them a bunch of crooks and dismissed them from my lab search. In looking back, however, I had to question whether or not I told them that my instrument was always battery powered.

The next big mistake I got away with was not being prepared. During the testing of my equipment, it was discovered that 81 MHz was being radiated. I had no idea from where, and what was worse, I had very little in my tool box to solve the problem. Because I was close enough to my office, I was able to get the components I needed.

Being prepared and anticipating the problems is crucial. With EMI, you make your own luck. The better prepared you are, the "luckier" you'll be.

In preparing to go to the test lab, there are several things to do, regardless of what stage your product is at.

First, pack a full tool kit that includes the items listed in Figure 1. The lab may have a good selection of tools, but don't count on it.

A good solder iron is critical. Bring a full set of capacitors, resistors, ferrites, and inductors. I use the designer's kits I get from the manufacturers.

With the ferrites, bring both the beads and the cable-clamp type. A drawing of filter topologies is often very useful, as is a Laplace transform table.

Another nice thing to have in the toolbox is conductive tape or aluminum foil for shielding or plugging up RF "holes." Spare boards are also a good idea, especially if you're doing immunity testing.

For each board and/or subassembly, a list of all frequencies that can be emitted helps you identify where the radiated emission is coming from.

For example, if your microprocessor board has a 20-MHz clock, it's reasonable to expect that you may see the fundamental and odd harmonics of this frequency.

Of course, you must keep in mind that the frequencies emitted will not be the exact 1st, 3rd, 5th, etc. harmonics because digital signals are trapezoidal, not squares. But, the frequency values should be close. A list of all the technologies and their rise times gives you an idea of possible frequencies that can be emitted.

Bring both internal and external cables of different lengths. Although any wire radiates or receives any frequency, cable length has a great deal to do with the amount of radiation emitted or received.

Also, bring some means of making cables. And, if your cables are not shielded or twisted, you should bring some. If your cables are shielded, bring double-shield cables.

If you're doing immunity testing, you'll most likely do Electrostatic

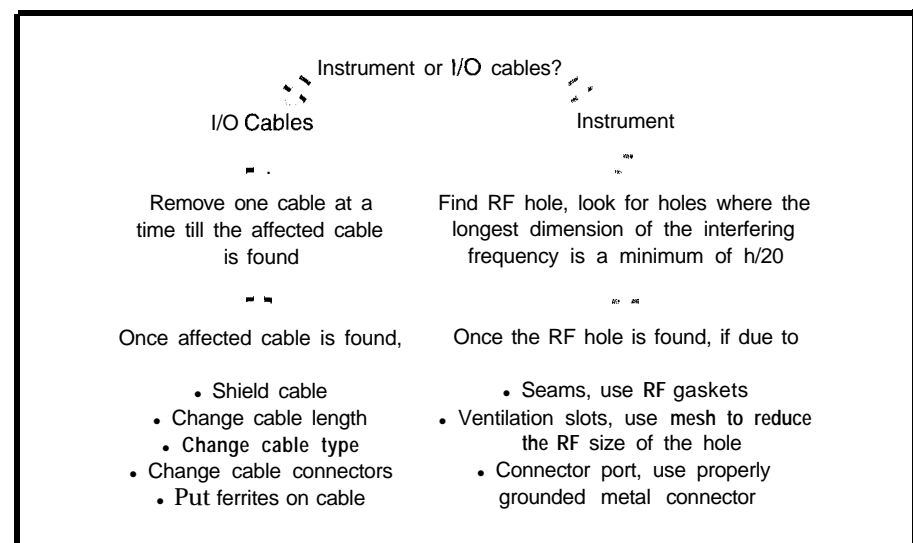


Figure 2—This simplified flowchart shows a possible plan of action if equipment fails radiated immunity tests. Of course, you'll expand this chart to reflect your particular instrument. Keep in mind that any solution will be a permanent part of your instrument, so keep the cost of the solution in mind!

Discharge (ESD) and Electrical Fast Transients/Bursts (EFT/B) tests. So, you'll want surge suppression, MOV, and the like.

Also, prepare boards that have surge suppression on all the I/O and power lines. If you have a transient problem, these help identify the location quickly.

Research. What you find may surprise you. Here's a couple of examples.

In school, we all learned that a signal traveling in coax cable is completely shielded. The signal travels down the center conductor, and the outer shield prevents any signals from entering or escaping from the cable.

Unfortunately, the shield is never perfect. If it's a solid conductor, it breaks. If it's a braid, it's leaky.

As the frequencies increase, the inductance of wiring increases. At ~10kHz, a wire has more inductance than resistance. So, the concept of a good RF ground comes into play. Many engineers use an ohmmeter to check the impedance of ground connection. Unfortunately, they're only testing the ground at DC.

Next, make a flowchart of possible failures and the actions you can take. Let's look at a couple of tests. Imagine a failure. What would you do?

TESTING EXAMPLES

Imagine that your microprocessor-based system radiates 152 MHz at a level that's unacceptable. First, as you see in Figure 2, you must determine whether it's from your I/O cables or from the unit. Disconnect the I/O cables, and test again.

If the problem is a cable radiating, try different lengths of cable, shield the cable with conductive tape or aluminum foil, use ferrite clamps, or place passive filters on the line. Only you can determine the best order to try these solutions since you know your equipment.

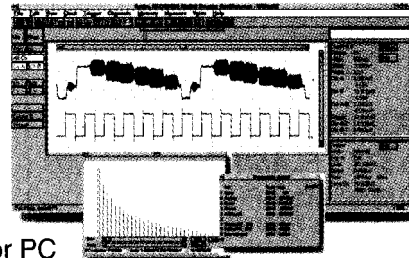
If the unit is causing the radiation, first decide whether it's the processor clock or some other signal. Try reducing the clock rate of your processor.

Some microprocessors have this facility built in. The Motorola 68333, my personal favorite, makes it easy to change the clock rate. With onboard flash, it can be updated via RS-232.

2 Channel 100 MSa/s Digital Oscilloscope



- 2 Channel Digital Oscilloscope
 - 100 MSa/s max single shot rate
 - 32K samples per channel
 - Advanced Triggering
 - Only 9 oz and 6.3" x 3.75" x 1.25"!
 - Small, Lightweight, and Portable
 - **Parallel Port** interface to Laptop or PC
 - Advanced Math, TV Line Trigger, and FFT Spectrum Analyzer options
- For \$499 you get the model DSO-2102 **Oscilloscope**, Probes, Interface Cable, Power Adapter, and Windows and DOS Software.



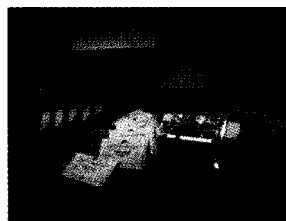
200 MSa/s Digital Oscilloscope

- 200 MSa/s max single shot rate
- 2 oscilloscope channels
- 8 Logic Analyzer channels
- 10 channels simultaneously
- 125 MHz Single Shot Bandwidth
- up to 128K Samples/Channel
- FFT and Spectrum Analyzer



DSO-28264 (10Ch, 200MS, 64k) \$1999
 DSO-28464 (20Ch, 200MS, 64k) \$3299
 All prices include Probes and Software

500 MHz Logic Analyzers



- 40 to 160 channels
- Variable Threshold
- 16 Level Triggering
- up to 500 MHz
- 8 External Clocks
- up to 512K samples/ch

LA4240-32K (200MHz, 40CH) \$1350
 LA4280-32K (200MHz, 80CH) \$2000
 LA4540-128K (500MHz, 40CH) \$1900
 LA4580-128K (500MHz, 80CH) \$2800
 LA451 60-1 28K (500MHz, 160CH) \$7000
 All prices include Pods and Software

Call for information on our
100 MSa/s Pattern Generator option

 **Link Instruments (201) 808-8990**

369 Passaic Ave • Suite 100 • Fairfield, NJ 07004 • Fax (201) 808-8786

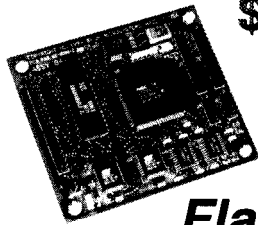
Web: <http://www.LinkInstruments.com> Email: Sales@LinkInstruments.com

EMBEDDED DOS

CONTROLLERS AT 8051 PRICES

Use Your PC Development Tools

NO MORE CRASH & BURN EPROM
Technology



\$195
Qty 1 Price

Flashlite

DOS Single Board Computer

with 572 k FLASH Memory disk drive

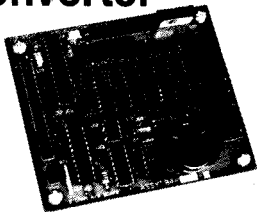
- ✓ 10Mhz/8 Mhz CPU
- ✓ 2 Timers
- ✓ 512 k bytes RAM
- ✓ 4 Interrupt Lines
- ✓ 512 k/256 k FLASH
- ✓ 8 Analog Inputs
- ✓ 2 Serial Ports
- ✓ X-Modem File Transfer
- ✓ 24 Parallel I/O Lines

INCLUDES DOS & Utilities

USE YOUR TURBO C COMPILER OR
QUICKBASIC COMPILER
SAVE TIME, MONEY AND HEADACHES

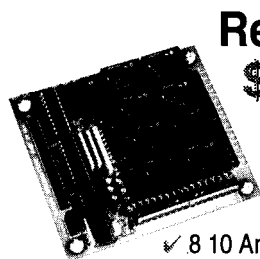
A/D Converter

\$95
Qty 1 Price



- ✓ 8 Channels, 12 Bits
- ✓ 6 us. Conversion Time
- ✓ Clock/Calendar Option
- ✓ Includes Drivers & Apps.

GET YOUR EMBEDDED CONTROLLER
PROJECT RUNNING FAST!
WITH THESE ACCESSORIES



Relay I/O
\$139
Qty 1 Price

- ✓ 8 10 Amp Relays
- ✓ 8 Opto-Isolated Inputs

JK microsystems
Cost Effective Controllers for Industry

TO ORDER (510) 236-1151

FAX (510) 236-2999—email: jkmicro@dsp.com
Visit our WEB site-www.dsp.com/jkmicro
1275 Yuba Ave., San Pablo, CA 94806

Other processors may require the crystal to be changed.

If your problem is gone after you reduce the clock rate, you probably have an antenna (cable or trace) tuned to 152 MHz on a line affected by the processor frequency. If the radiation is not affected by the microprocessor frequency change, then look to other clocks or oscillator sources.

Consider another possible scenario. Your equipment hangs up during transient testing.

First, install the boards you made with surge suppression on every line. If that solves the problem, then isolate the board by returning the original boards one by one to the unit. Once the board is located, find the individual line and put surge suppression on it.

Some designers may decide that surge protection on every line is a good design practice, but it depends on the equipment. If it's a high-end product, a few extra dollars don't matter. For cost-sensitive equipment, a few extra dollars are the profit margin.

Remember how you pass the tests determines how you build the equipment. If you pass the test with every line surge protected, that's how the equipment must be built and sold.

PRECAUTIONS

This may seem like overkill, but keep a few things in mind. The labs are very busy, so use time wisely. If you have one day booked, don't spend half a day making cables.

When you arrive at the lab, make a test plan for each test to be efficient. You don't want to wait while the test technician decides what test to do next and then finds they're missing the proper antenna.

Time is money. Know the order of the test, and make sure that the lab and you have everything necessary. If you need something, get it while another test is being run.

During the test, don't just sit and watch. Take careful notes of the test setup and equipment. Ask questions. It may slow down the works, but your notes may be of great value.

If you're failing, experiment at the lab as much as possible. They have the test equipment. Here, your flowchart

is crucial. Under pressure, you don't have time to think about what to try next.

Document well what you tried that didn't work. Not only is this information very valuable, but you'd be surprised how many times the same thing is tried.

COMING HOME

When you get back to your office, whether you pass or fail, document what happened. If you passed, the information will be useful for your next trip. If you failed, the notes will help you discover a solution to the problem.

Look at EM1 tests as sudden-death exams. You could be passing every test, but if you fail one, you can't ship.

Of course, there are plenty of other things to learn about EM1 standards and testing. However, a basic understanding is the first priority since the standards and tests are design specifications. And, by far, the best place to handle EM1 is at the design stage. □

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

REFERENCES

- FCC, *Code of Federal Regulations*, Title 47, Parts 15 and 18, 1995.
- IEC Standard 1000-4-1, *Electromagnetic Compatibility, Testing and Measurement Techniques, Overview of Immunity Tests, Basic EMC Publication*, 1992.
- C. Marshman, *The Guide to the EMC Directive 89/336/EEC*, EPA Press, Ambo, UK, 1992.
- T. Williams, *EMC for Product Designers*, Butterworth and Heineemann, Oxford, UK, 1996.

IRS

- 419 Very Useful
- 420 Moderately Useful
- 421 Not Useful

Infrared Remotes are Everywhere... If You Can Find 'Em

FROM THE BENCH

Jeff Bachiochi

duces a beeping sound to help you locate it. But what happens when you lose the remote's remote?

I enjoy a good gadget as much as anyone, but initially, I had my doubts about a recent gift I got. A watch is a noble gift to give or receive. One of my favorites has a calculator built in, but the keypad didn't last.

This time around, I wasn't going to calculate. This watch has an IR transmitter built in. At first, I thought, "Gimmick," but then, I found it replaced the remotes I could never find.

This month's project takes advantage of this transmitting timepiece.

CASIO WRIST REMOTE

This watch, shown in Photo 1, is like having an all-for-one IR remote with you at all times. It's compatible with most TV, cable, and VCR equipment made by today's leading manufacturers-GE, Goldstar, Hitachi, Jerrold, JVC, Magnavox, Mitsubishi, Panasonic, Philips, Pioneer, RCA, Samsung, Scientific-Atlanta, Sharp, Sony, and Toshiba.

Although there's no full-function keypad, you can still access most important functions-power on/off, channel up/down, volume up/down, play, fast forward, rewind, and stop.

Unlike many Windows applications with their pop-up help menus, this Casio requires a small instruction booklet. Once the manufacturer codes are set, the reference is not necessary until you get a new piece of equipment.

The first thing you notice is that most manufacturers still haven't agreed on a universal IR scheme.

are you constantly playing hide and seek with your TV/VCR/audio/CD/cable/satellite/toaster-oven remotes? At our house, trying to find the remote is like hunting down the mates to those odd socks in the dresser drawer.

We own a piece of furniture especially designed for remotes (along with the latest TV and cable guides). But when I sit down to view a bit of mind-numbing entertainment, there's no listing guide in sight, never mind a remote. It's quicker to find the daily listing in the newspaper and walk over to the TV to change channels than it is to locate the surfing tool.

Not long ago, I saw a commercial for a device that attaches permanently to the back of the clicker and pro-

Far too frequently, dogs, kids, couches, or Mrs. Doubtfire find unique and hard-to-find locations for that all-important channel changer. But Jeff's new watch has IR, so it's time to review just how IR receivers and transmitters communicate.

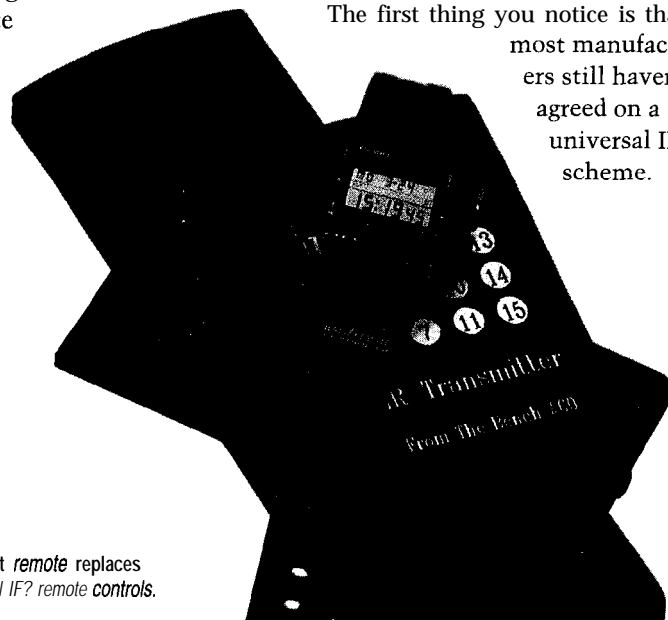


Photo 1--The Casio wrist remote replaces many of the standard AN IR remote controls.

That's why you have to set specific codes.

To be able to recognize IR codes, you must know a little about them. If you don't have a scope handy for experimenting, you can gain a bunch of insight from Ed Nisley's article on the MCIR-Link ("Extending Your Control: The HCS II MCIR-Link," *INK 29*). Although I have a scope, it isn't a storage scope, so it doesn't lend itself to capturing pulse trains.

I decided to create a tool to analyze various signals' bursts. It served as the front end of the project and reduced the programming.

The first objective was to look at various signals and find simple ways of identifying them. I wanted a real-time display of just what was happening.

Most IR transmissions are actually bursts of a 38.4-kHz carrier. A single cycle of 40-kHz carrier lasts 25 μ s. The demodulator needs a minimum burst of a few cycles. I picked 250 μ s as a sample time. Most minimum bursts seem to be around 500 μ s.

One of Microchip's newest little chips is an 8-pinned processor—certainly far from what most would consider the minimum computer. After all, what can you do with a computer that has only 4-6 I/Os?

Actually, quite a lot! With an internal RC oscillator, all the pins except power and ground can be used as I/Os. (The internal RC oscillator is factory trimmed to 4 MHz.)

This month's project is designed for the 12C508. Keep in mind that the full project requires two programmed devices—the first to gain information and the second to use that stored info in a permanent fashion. The device is an OTP, but windowed parts are available.

To give real-time data analysis, I chose to send characters out a port pin serially. At 9600 bps, that works out to about 2.5 bits of a character. Hmm... I could quickly see that I needed to do at least 57.6 kbps, if not 115 kbps, just

to send a single character and have some time left to do something else.

Because I like using a 4-MHz crystal, which gives 1- μ s instruction times, and the 12C508 was fixed at 4 MHz, I knew I was going to have some baud-rate timing errors. Calculations showed roughly 2% for 57.6 kbps and 4% for 115 kbps (rounding to the nearest 1 μ s).

I went with the 2% error. RS-232 communications can easily handle 2%. This choice enables me to send a single character (10 bits at 17 μ s per bit) for each sample and still have 80 μ s left.

Since what I was wanted to see was how the IR was going to be interpreted by the micro, I went for simple. Send an "H" if the sample was logic high and "L" if the sample was logic low.

In addition to sampling the demodulated IR and sending out a reflection of the sample's logic level, I tracked the number of consecutive times the same logic state was sampled. Using a single-byte counter allowed for a steady logic state of 64 ms (250 μ s x 256). A state that lasted that long was illegal.

An illegal state causes the serial output bit to be redefined as an input bit and therefore halts any further output to whatever serial device I had it connected to. The serial device may be a printer or a smart/dumb terminal. Either way, it prevents the data from spooling endlessly once the IR is removed (i.e., it stopped changing state).

I'm a crash'n'burn guy. The chances of my program running the first time out of the gate are nil, even with a bit of time spent with the simulator.

My objective was to create code that could be used with the little PIC12C508 processor. However, I'd probably have to go through a bunch of OTP devices or else spend all my time in front of the EPROM eraser. So..

I CHEATED

I used a PicStic1. Its code space is EEPROM and electrically reprogrammable. I downloaded compatible assembly code (hex file) to it and

used it for development.

Once the code ran, I reassembled for the 12C508. Since the 12C508 doesn't have a UART or interrupts, the serial needed to be bit banded.

In addition, the rest of the code had to be cycle counted such that each possible code path always took exactly the same number of instruction cycles. If the serial and sample loop was correctly timed, samples could be accurately taken each 250 μ s.

I also cheated on the RS-232 connection. To eliminate the MAX232 (used in good designs) for conversion of TTL to \pm 10-V RS-232 signals, I produced an inverted TTL serial output. Although not good design practice, I'd only use it for analysis and it wouldn't be necessary for the final product's operation.

Note in Figure 1 how the serial output line is pulled to ground. This feature keeps glitches from looking like start bits to the serial device whenever the output pin is reconfigured as an input pin.

Using ProComm, I captured my first visual glimpses of the (demodulated) IR produced by the Casio. As I input various manufacturers' codes and saw evidence of different kinds of coding, I thought I may have bitten off more than I could chew.

I soon realized, however, that I ultimately wanted to simply turn on or off one of the micro's outputs by recognizing an IR code. Therefore, I didn't necessarily have to completely understand every transmission scheme used by all the manufacturers.

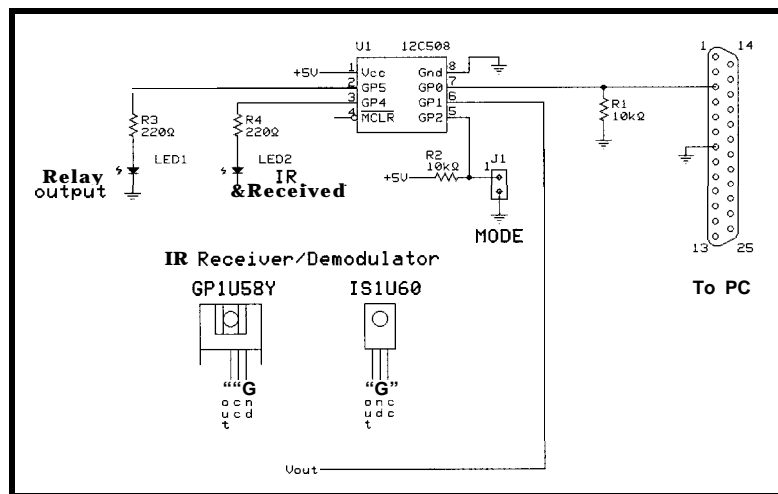


Figure 1—The circuit shows the test setup for this project. The finished circuit only requires an IR input (GP1), the control output (GP5), and an optional IR received LED output (GP4).

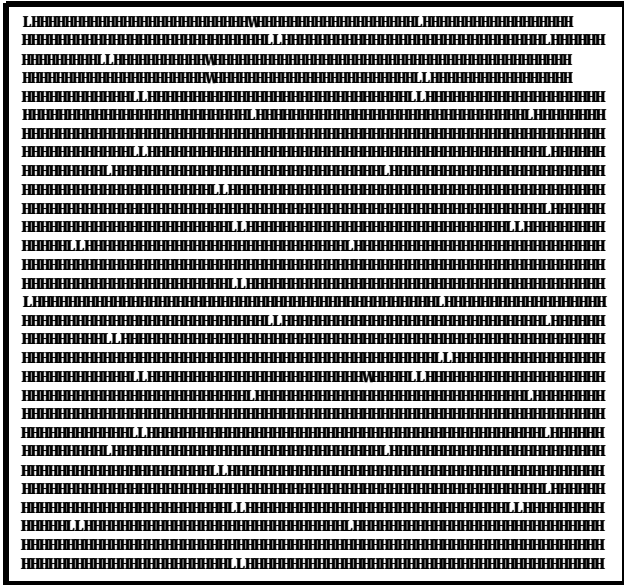


Figure 2—ASCII serial output shows logic levels received from an IR transmission.

For instance, one scheme (shown in Figure 2) produced a pattern of 34 logic-high duration cycles before repeating the sequence. Comparing the patterns produced by four different functions [push buttons], I quickly saw that after sampling the first three logic-high durations, I could easily tell which of the buttons was pressed. Was that enough!

ENOUGH ALREADY!

Fact is, the same three-value combination can easily come up again in a particular sequence. It also depends on the tolerance of what's considered a match. Must values match exactly? What about a deviation of ± 1 count?

My experiments show that the minimum duration of a logic high is about 16 cycles (4 ms), while the logic-low duration is always -500 μ s for this particular manufacturer's code.

The logic-high durations also seem to be in multiples of 16. It's then easy to see that a single compare with a built-in tolerance of $+7/-8$ can be achieved by adding 8 to each sample and making a compare, masking out the low nibble shown in Table 1.

The 12C508 parts have no EEPROM data storage like the 16C84, so the final table of numbers as sampled by the program needs to be placed in a table along with the code that's running.

You can also fit a number of tables into this 8-pin package. Each table, when matched to an incoming sample, performs a different operation (e.g., set, clear, toggle, or pulse a particular bit).

But, that's getting a little too far ahead. First, I need to decide how many durations long the table will be. I chose to use 16 table entries because the chances of that number reoccurring seemed highly improbable.

I stuffed them into RAM while in a programming mode. Then, I switched to a compare mode to quickly test new sequences without physically programming a table with them.

To make debugging easier, I programmed the sample counts to be dumped at the end of either a program or compare cycle. And, I added a 1-s pause after each compare cycle. This way, I could see what the micro was seeing and a new compare wouldn't begin immediately.

STOP, LOOK, AND LISTEN

When the program executes, it looks at the mode input pin to determine which mode to run in. Programming mode samples the IR every 250 μ s.

Once a logic state remains constant for 256 samples, it's considered idle. An idle state clears the table pointer preparing to start a compare.

As an IR stream is received, each high and low logic-level duration is counted by an 8-bit counter. The adjusted high-level counts (8 added and the lower nibble masked off) are stored into sequential RAM registers.

Table 1--Eliminate the need for multiple compares by reducing resolution.

Decimal	Binary	Binary+8	High Nibble
7	00000111	00001111	0000xxxx
8	00001001	00010001	0001xxxx
9	00001010	00010010	0001xxxx
22	00010110	00011110	0001xxxx
23	00010111	00011111	0001xxxx
24	00011000	00100000	0010xxxx

For some manufacturers, you may need to save the low-level counts or even both.

When the appropriate number of duration counts are stored, an output LED is enabled, indicating a successful program cycle. In this mode, execution stops until the programming mode bit is grounded (compare mode).

Execution then continues in compare mode. To resample (reprogram) an IR sequence, leave it in programming mode (input bit high) and reset the processor.

Different programming practices are used depending on whether the table will be soft coded (programming mode) into RAM or hard coded as a table. I don't envision this device being programmed by the user at the time of installation. Instead, I think it should come programmed for a particular code (e.g., my Casio).

Using the PicStic offers more options and flexibility, but this project is intended to end up as an exercise in minimum parts cost and size.

Independent of the storage type (i.e., RAM or code table), the compare process is the same. At idle, the table counter is cleared. In addition, a "good" flag is cleared.

As the IR stream is detected, the consecutive high-duration samples are counted as before. Instead of storing an adjusted count, it's compared to the corresponding table entry. If *not* equal, the "good" flag is incremented.

The samples continue until the end of the table is reached. The output LED is enabled, signaling the end of a compare. If the "good" flag is still at 0, all entries matched and a good compare function is performed. In this case, an exclusive OR toggles the relay logic output to the opposite state.

In both the program and compare modes, once 16 samples are taken, the sample counts are spit out of the serial

output bit in ASCII. Now that I've programmed the device from the selected Casio function key and verified that it compares correctly, I can use these counts in the compare code table used by the 12C508 microprocessor.

The final code placed in the 12C508 doesn't require the programming mode. Its compare table is hard coded and cannot be changed. The serial output bit isn't needed anymore.

However, I don't have to change the code. I don't need those functions. Just three bits are necessary-the IR input, the compare status LED output, and the RELAY output. I'll bet you can find useful functions for those other bits.

This complete circuit can be built into almost anything-a lamp, radio, coffee maker, or door lock. The circuit current is small enough to be powered from an unisolated AC/DC circuit. No bulky transformer is needed.

A 5-V relay can be directly driven from the micro's port pin for a high-sensitivity model that doesn't exceed the micro's source current. However, a solid-state relay is a good alternative.

So, don't be nervous about experimenting with new things. Use the tools you already have to develop the tools you need to explore new areas.

And, if you have any sure-fire ways to keep the remote from getting lost, please share them with me.

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.

SOFTWARE

All the software tools I used to develop this project's code are available from Microchip's Web site.

SOURCES

LT1060-ND
Digi-Key Corp.
701 Brooks Ave. S
Thief Falls, MN 56701-0677
(218) 681-6674
Fax: (218) 681-3380

12C508
Microchip Technology, Inc.
2355 W. Chandler Blvd.
Chandler, AZ 85224-6199
(602) 786-7200
Fax: (602) 786-7277
www.microchip.com

PicStic1
Micromint, Inc.
4 Park St.
Vernon, CT 06066
(860) 871-6170
Fax: (860) 872-2204
www.micromint.com

ISIU60, GP1U58Y
Sharp Electronics Corp.
Microelectronics Group
5700 NW Pacific Rim Blvd., Ste. 20
Camas, WA 98607
(206) 834-2500
Fax: (206) 834-8903

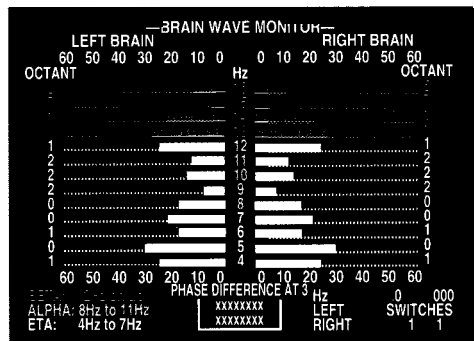
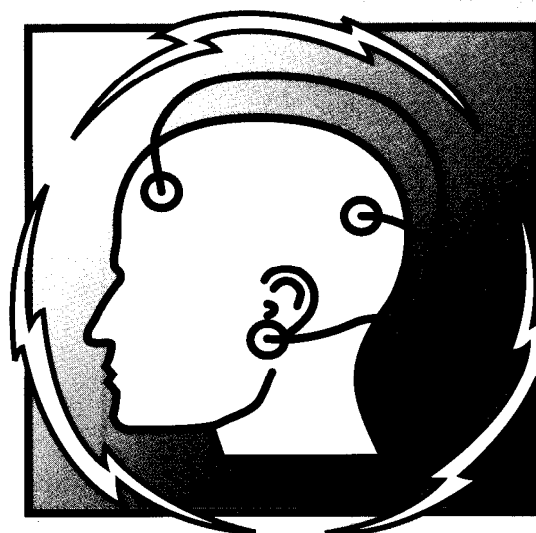
IRS

422 Very Useful
423 Moderately Useful
424 Not Useful

HAL-4

EEG Biofeedback Brainwave Analyzer

The HAL-4 kit is a complete battery-operated 4-channel electroencephalograph (EEG) which measures a mere 6" x 7". HAL is sensitive enough to even distinguish different conscious states-between concentrated mental activity and pleasant daydreaming. HAL gathers all relevant alpha, beta, and theta brainwave signals within the range of 4-20 Hz and presents it in a serial digitized format that can be easily recorded or analyzed. HAL's operation is straightforward. It samples four channels of analog brainwave data 64 times **per second and transmits this digitized data serially to a PC** at 4800 bps. There, using a Fast Fourier Transform to determine frequency, amplitude, and phase components, the results are graphically displayed in real time for each side of the brain.



PACKAGE PRICE - \$279⁹⁵ SHIP
Contains HAL-4 PCB and all circuit components, source code on PC diskette, serial connection cable, and four extra sets of disposable electrodes.

to order the HAL-4 Kit or to receive a catalog,
CALL: (860) 871-6170 OR FAX: (860) 872-2204

CIRCUIT CELLAR KITS • 4 PARK STREET
SUITE 12 • VERNON • CT 06066 • www.circellar.com

• The Circuit Cellar Hemispheric Activation Level detector IS presented as an engineering example of the design techniques used in acquiring brainwave signals. This Hemispheric Activation Level detector is not a medically approved device, no medical claims are made for this device, and it should not be used for medical diagnostic purposes. Furthermore, safe use requires HAL be battery operated only!

Modem Déjà Vu

SILICON UPDATE

Tom Cantrell



our refrigerator will be on the Internet."

"There he goes again" is probably what you (and my long-suffering editor) are thinking. It's true I get a little wild when it comes to prognosticating about technology. It's also true I usually end up being right.

That's because no special intellectual or psychic abilities are needed to predict a science-fiction future for ICs. Just don't specify a date.

"In our lifetime" works especially well since neither would-be debunkers nor the debunker care much after that. If you simply wait long enough, you can rely on ever-improving chip price and performance to prove you right.

Actually, it's not hard to imagine a WebFridge scenario. Who hasn't cruised into the kitchen late at night only to find "Arrgh! No milk! "?

Already, grocery-shop-by-Web services are sprouting up. Wouldn't it be great if your refrigerator could stock itself? How about shoving a frozen turkey in a WebOven that dials up www.julia_child_cooks_4_u.com and takes it from there?

Admittedly, there are a few minor obstacles to overcome. Embedding the Web feature has got to cost tens, not hundreds (e.g., WebTV) or thousands (e.g., WebPC), of dollars. Also, the Web, which appears to be running on the ragged edge already, isn't likely to gracefully handle hundreds of millions of appliances coming online.

Oh, by the way, the refrigerator quote isn't mine. Lou Gerstner, head of IBM and hardly a wild and crazy guy, said it. Of course, I happen to agree that his vision will come true someday. The question is: Will it be sooner or later?

DAYS OF MODEMS PAST

I can kind of remember my first modem some twenty years back. At that time, they (like all computer stuff) were rather esoteric items. My budget already blown on 8-KB RAM boards, I tried to make do with a cheap [only a couple hundred bucks!] build-it-yourself modem kit.

It seemed to make the right sounds, and I did manage to get connected to the 360/91 across town, but it was just too flaky to be usable. Nothing against



For many solutions, just getting there is

what matters, not how fast. TDK's latest modem chips are how Tom proves this point. In fact, it appears that old modems—like the tortoise—sometimes win the race.

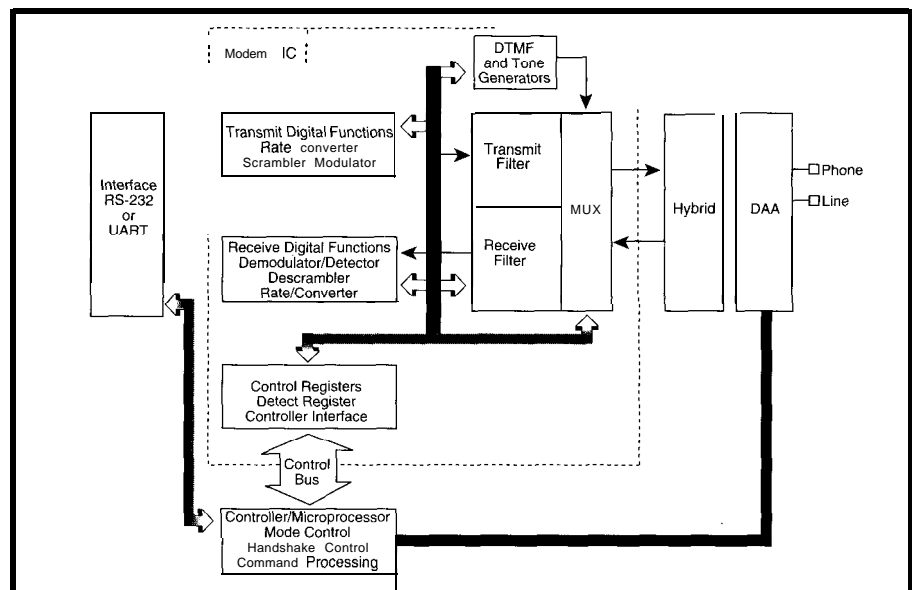


Figure 1—This generic block diagram of a modem shows the major building blocks—modulator and demodulator, filters, DTMF and tone generator, hybrid, and DAA.

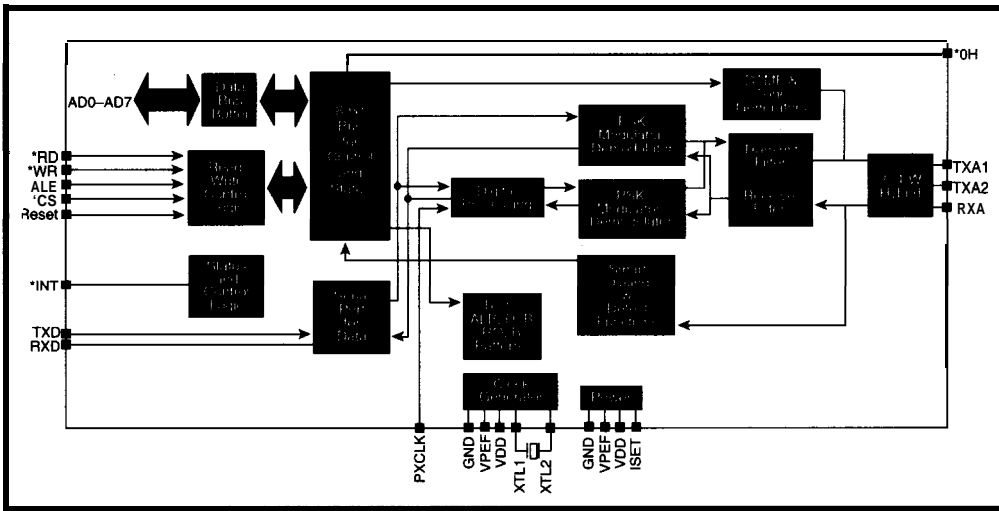


Figure 2—The 73K222BL integrates almost everything between a micro and the isolation barrier (i.e., DAA), up to and including the 2-wire-to-4-wire hybrid.

DAAT'S NOT ALL FOLKS

It gets worse. Though lightning may not strike twice, Murphy says it's sure to strike at least once just about the time you jack in.

With consumer safety in mind, UL zaps your box with simulated lightning strikes—

the kit designer or supplier. It could be that I just got one of the zillions of resistors and caps wrong. In any case, I had neither the tools nor the skill to debug it and ended up just spending more for a surplus commercial unit.

It's not surprising I had problems since modems are quite a black art. The concept starts out simple enough but runs into a witch's brew of analog foibles, regulatory restrictions, and profligate standards as soon as you plug into the phone jack.

Certainly at that time, the modulator and demodulator (from which the word "modem" is derived) were quite straightforward. Simply allocate one frequency for 0, another for 1, and have at it—an approach known as FSK (Frequency Shift Keying).

If you want simultaneous send and receive (i.e., full-duplex), allocate four frequencies—two for the originating modem and two for the answering one. Sample and update at 300 Hz, and

voilà, you've got the guts of that long-ago (Bell 103 standard) 300-bps modem.

Of course, there are a few gotchas, the most critical being that the signal can get ugly by the time it crosses hill and dale. Your design may work great with a clean 2-V signal in the lab, but when it gets 20 mV in the real world...?

Oh well, with all those op-amps you need for filters, a few more for adaptive gain won't hurt. While you're at it, be sure to provide equalization since the network is prone to attenuate and delay some frequencies more than others.

Going beyond this so-called data pump is where things get tricky. For full-duplex operation, the outgoing and incoming signals have to be joined. But, you can't just connect the wires since simultaneous transmit and receive will step on each other.

So, you've got to add a hybrid, duplexer, or 4-wire-to-2-wire converter to put your transmission on the line, but make sure you don't hear it.

the ultimate smoke test! It's OK if your gadget dies, but it must die quietly without fireworks. Protection is a two-way street, and the FCC (Part 68) steps in to ensure the phone network doesn't suffer from your design goofs.

The result: all modems (indeed anything connected to a phone jack) needs an approved Direct Access Arrangement (DAA) that bridges the isolation barrier between your electronics and the phone wire. Other functions that cross the bridge include the off-hook relay and ring detector.

You can use the relay for pulse dialing, but these days, it's best to throw in a DTMF (Dual Tone Multifrequency) touch-tone generator. Put it together with flashing LEDs and a speaker, and you've got all the pieces of the modem puzzle (see Figure 1).

In the old days, Ma Bell ruled the phone wires, and Bell 103 was all there was standards-wise, certainly in the U.S. and maybe even worldwide. Since

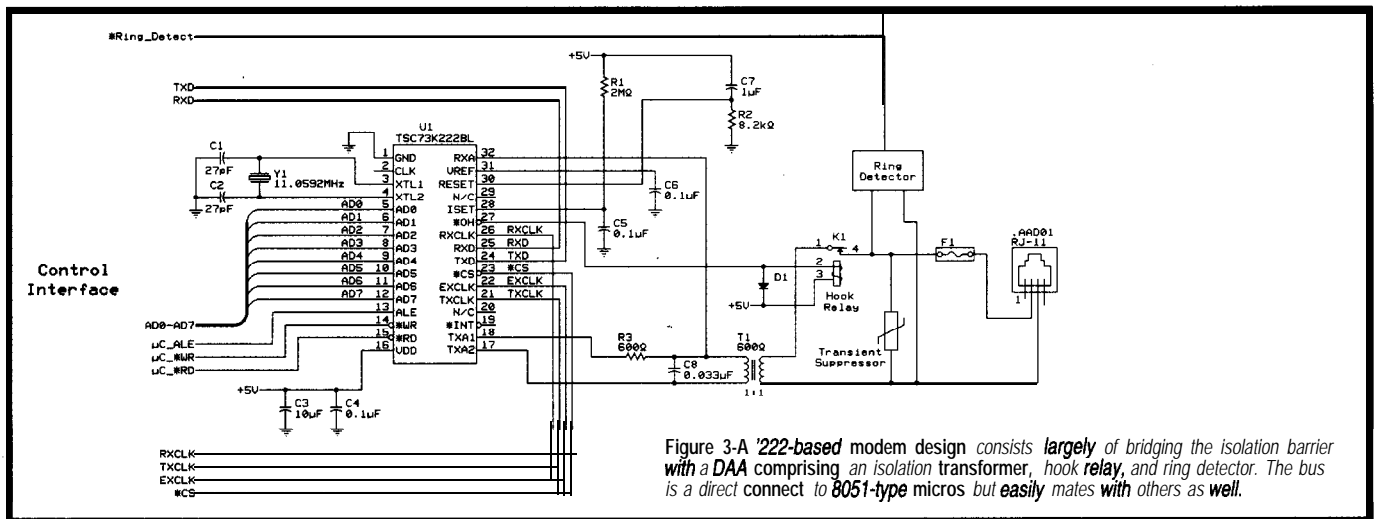


Figure 3—A '222-based modem design consists largely of bridging the isolation barrier with a DAA comprising an isolation transformer, hook relay, and ring detector. The bus is a direct connect to 8051-type micros but easily mates with others as well.

then, however, the high-tech equivalent of the U.N. (CCITT) has blessed a bewildering variety standards—all those funny V.xx numbers littering modem ads.

Modulation has gotten more clever, encoding more bits into each sample by fiddling with phase (e.g., DPSK or Dibit Phase Shift Key) and amplitude (e.g., QAM or Quad Amplitude Modulation) with the latest schemes cramming up to 12 bits per sample. This, combined with speedier sampling (from 300 to 2400 Hz) is how we got today's fast modems (e.g., 2400 x 12 = 28.8 kbps).

Just to spice things up, remember all these international standards mean every country's equivalent of UL and FCC gets to bless your design, too. The bottom line is modem design wasn't easy then, and it's even harder now. Fortunately, IC wizards ease the pain.

DIP WHISTLE

If you're willing to sacrifice speed, modem chips like the K-series from TDK Semiconductor deserve a closer

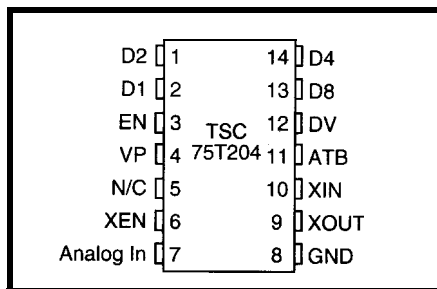


Figure 4—Connected to an inexpensive TV color-burst (3.58 MHz) crystal (XIN, XOUT), the '204 DTMF receiver listens to the line (Analog In) and signals (DTMF Valid) the occurrence of DTMF tones and outputs them (D1, 2, 4, 8) when enabled (EN).

look. While modem design is still tough, the latest chips make it as easy as it can, and pretty much ever will be.

The K-series comprises a family of semiplug-compatible chips that cover the popular (Bell 103/212A, V.21, V.22, V.22bis) 300-, 1200-, and 2400-bps standards. Essentially, the family line-up is a matrix permuting modulation (FSK, FSK+DPSK, FSK+DPSK+QAM), host CPU interface (serial+parallel or serial-only) and package (PLCC and DIP). TDK also offers a couple of models with built-in PC-compatible UART.

Let's take a look at a recent introduction, the TSC 73K222BL, which handles FSK and DPSK coding for 300- and 1200-bps operation.

Figure 2 shows how the latest chips make modem design as easy as can be. The '222 incorporates practically all the equipment-side electronics—modulator, demodulator with all the filters, call progress (e.g., dial tone, busy, etc.) detection, DTMF generator, 40-mA off-hook relay driver, and now even the 2-wire-to-4-wire hybrid.

Notice that the chip goes as far as it can toward the phone network. That is, everything else (the DAA composed of coupling transformer, off-hook relay, and ring detector) has to cross the isolation barrier as shown in Figure 3.

Thus, short of adding the processor, it's hard to imagine a more integrated device on a single piece of silicon. You still have to provide the DAA, by making, testing, and getting your own design approved, or buying a preapproved unit from a specialist like Cermetek.

The '222 includes serial and parallel interfaces. The serial interface (TXD

EPROM EMULATORS

NEW 3V emulators from \$229.00



E1 up to 128Kx8 E4 up to 512Kx8

- 85ns and 35ns standard access times
- 3V LV models operate at 3V and 5V
- High-speed downloading (LPT1 -3) with error checking and correction
- Loads binary, Mot-S, Intel
- Power-up emulation
- *Compact size, with protective case
- Low power design, 5mA max.
- Software configurable
- *Discounts on 2+ units

Prices

E1-85	\$199
E4-85	\$249
E1-35	\$229
E4-35	\$299
A-PLCC	\$65
E1LV-90	\$229
E4LV-90	\$299

SDI SCANLON DESIGN INC. 800 352 9770
(902)425-3938 Int'l
(902)425-4098 FAX

internet: 71303.1435@compuserve.com
5224 Blowers St. Halifax, N S, Canada B3J1J7

#130

Does your Big Company marketing department
conceive with more ideas than the engineering
department can cope with? Are you a small
company that can't afford a full-time engineering
staff for once-in-a-while designs?

Steve Ciarcia and the Ciarcia Design Works
staff may have the solution. We have a team of
accomplished programmers and engineers
ready to design products or solve tricky
engineering problems.

Whether you need an on-line solution for
a unique problem, a product for a start-up
venture, or just experienced consulting,
Ciarcia Design Works is ready to work with you.

FAX: (860) 871-8986

steve.ciarcia@circellar.com • www.circellar.com

#134

and RXD) transfers data across the phone line, while the parallel interface provides access to on-chip status and control registers and is compatible with 8051 and other multiplexed (address/data) micros.

Some models route both data and control/status through the serial bus to shrink into a 22-pin DIP. Even those with the parallel bus offer a 7-wire serial interface scheme using ADO-AD2 as AO-A2 (to address the status and control registers), AD7 as the data bit, ● RD and *WR as direction, and EXCLK as the clock. An 'INT output requests service on detecting dial tone, carrier, and other line-status indicators.

The clock generator starts with an 11.0592MHz crystal which must be accurate ($\pm 0.01\%$) and stable over temperature and time. For convenience, the CLK output pin can be programmed to pass through 11.0592 MHz or a clock that's 16 times the data rate, which is handy for connecting to a UART.

While we're talking about clocks, let's clear up some confusion about asynchronous and synchronous since the terms have different meanings in computer- and modem-speak.

Computer types recognize async as the well-known UART format in which start and stop bits, not a clock, frame data transfers. Sync refers to data transfers slaved to a clock, either a separate pin or buried in the data.

From the modem's point of view, only FSK (i.e., 300 bps) is asynchronous in the sense that you can send 1s and 0s anytime you want and they appear at the other end with similar timing. All decoding issues are left for the devices at each end to deal with.

By contrast, beyond-FSK schemes (e.g., DPSK and QAM) are synchronous. They need the data transferred at precise intervals for the more sophisticated modulation/demodulation schemes. Thus, the '222 provides TXCLK and RXCLK outputs. If you run FSK (300 bps), you can ignore them. If you run DPSK (1200 bps), you must use them.

However, there's nothing wrong (except a little confusion) with funneling async (computer-speak) data across a sync (modem-speak) connection. For instance, you can use a UART and DPSK at the same time.

To do so, however, the somewhat sloppy timing (a couple of percentages) of the UART has to be cleaned up to meet the much stricter ($\pm 0.01\%$) synchronous timing spec. Fortunately, the '222 includes an ASYNC/SYNC.

Alternatively, you can switch to a synchronous (computer-speak) data format such as HDLC.

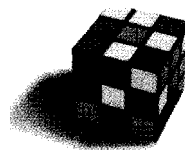
A number of suppliers (e.g., Zilog and Motorola) offer CPUs with integrated synchronous communications. In fact, TSC offers a version of the 8032 with HDLC support called the 73M2910A.

While you're shopping, keep an eye out for a DTMF receiver if you need tonal touching from time to time. TSC sells those too, including the 75T204 (\$7 for 10k pieces) shown in Figure 4.

LITTLE CPU AND BIG FIRMWARE

Thanks to the relatively limited speed and protocol complexity, an 8-bit CPU is up to the task of driving the '222. According to TDK app notes, if you can handle interrupts at 600 Hz, you've just about got it licked.

The micro directs traffic in the modem via five registers shown in Figure



Par • a • digm: your source for the most high-powered, comprehensive set of time-saving software and hardware development tools for embedded application development.

1: Paradigm LOCATE the most popular tool for creating embedded C/C++ applications with Borland and Microsoft compilers; **2: Paradigm DEBUG** the only x86 debugger with C++, RTOS, scripting language, and full in-circuit emulator support; **3: Paradigm SUPPORT** the best technical support in the industry supplied to our customers for free.

Developing real-time embedded applications doesn't have to be time consuming or difficult—you just need to have the right tools. Paradigm alone has the high performance development tools you need to streamline the embedded system software development process so your Intel and AMD x86 applications are ready in record time. Paradigm's complete suite of tools work with industry standard C/C++ compilers from Borland and Microsoft, as well as hardware development tools from Applied Microsystems, Beacon Development Tools and other popular in-circuit emulator vendors.

Call us at 800-537-5043 today and let us take care of all your development tool needs, so you can keep your focus where you need it—on your application.

Paradigm

Paradigm Systems
3301 Country Club Road
Suite 2214
Endwell, NY 13760

1-800-537-5043

Phone 607-748-5966
Fax 607-748-5968
info@devtools.com
<http://www.devtools.com>



5. Control register 0 (CRO) defines the basic mode of operation (i.e., modulation standard, answer or originate, and data format), while Control Register 1 (CR1) configures the CLK and *INT pins and loopback modes for testing.

CR1 can also force certain output patterns (i.e., mark, space, alternate) to test (also used when modems of different types "negotiate") and bypass the scrambler, which stuffs the bits necessary to guarantee enough signal activity for clock recovery at the other end.

CR1 also contains the all-important RESET bit. TDK points out you can (and probably should) use it instead of providing (or relying) on an external power-up reset circuit.

The Detect Register (DR) gives you some clue of what's happening on the line, including a copy of the RXD bit so the CPU can monitor incoming data through the parallel interface. The other bits detect specific line status (e.g., unscrambled marks, carrier, answer tone, dial tone, etc.).

The long-loop bit is an indication of a weak connection likely to experience errors, so maybe you should hang up and try again. As mentioned, many of

these lines can assert the ● INT output if so enabled by bit 5 of CR1.

The Tone Register (TR) does what it says. It outputs relevant tones, including guard (used in Europe to prevent bleeding into verboten frequencies), answer, and DTMF for touch-tone dialing.

The ID register contains a code so your firmware can figure out what chip it's talking to. Since many K-series parts share basic features (i.e., register formats), it's possible (but not trivial) to come up with firmware that works with many chips and standards. The ID register is also where you twiddle the *OH (Off-hook) output that connects to the external relay.

It may not take many MIPS to get the job done, but don't underestimate the complexity of dealing with protocol. Consider especially the negotiation phase that takes place when two unlike modems try to communicate.

For instance, Figure 6 shows the V.22 connect sequence, and there's a different figure like it for each standard. And, I haven't even started talking about bells and whistles like the ubiquitous AT command set.

Fortunately, TSC offers sample firmware (though only for '5 1 family chips) that can get you off to a running start. It even includes MNP 4 and 5, which are popular enhancements for error correction and data compression.

It all sounds grand, but just keep some "make versus buy" realities in mind. If you only need a few-or even a few dozen-modems, simply head over to the local PC shack.

If you're buying a few hundred or thousand, check out some of the modem modules on the market. Cermetek, the source of the mentioned turn-key DAAs, offers complete modem modules as well.

However, if you're looking at 10k+ units, know what you are doing, and aren't intimidated by the regulatory inquisition, TSC may be the way to go.

GO SLOW OR GO BROKE?

Even if you don't buy the WebFridge concept, there's no shortage of applications where price is more important than speed. Like your shopping list, many day-to-day transactions like credit-card, automated-teller, and vending machines; security and inventory systems; and data loggers just don't call for much data.

Surprisingly, many of these applications not only don't need a faster modem, but would be poorly served by one. For instance, as long as the telephone company bills by the minute, the toll is the same (if your message is shorter than a few thousand bytes) no matter which modem you use.

Sometimes, it takes longer to set up the connection than transfer the data. Ironically, due to the simpler negotiation phase for older standards, there's some amount of data (perhaps at 100 bytes or so) below which a 1200-bps transaction completes while the higher speed modems are still trying to figure out who they're talking to.

The slower modem is also more robust in the face of

Register	Address	D 7	D6	D5	Data Bit Number			D1	D0
Register	AD2-AD0	D 7	D6	D5	D4	D3	D2	D1	D0
Control Register CR0	000	Modulation Option	0	Transmit Mode 3	Transmit Mode 2	Transmit Mode 1	Transmit Mode 0	Transmit Enable	Originate/ Answer
0 = 1200-bps DPSK 0 = 2400-bps DPSK 1 = V.21 FSK 0000 = PWR Down 0001 = INT Sync 0010 = TX Sync 0011 = Slave Sync 0100 = Async 8 bits/char 0101 = Async 9 bits/char 0110 = Async 10 bits/char 0111 = Async 11 bits/char 1100 = PSK 0 = Disable TXA Output 1 = Enable TXA Output 0 = Answer 1 = Originate									
Control Register CR1	001	Transmit Pattern 1	Transmit Pattern 0	Enable Detect Interrupt	Bypass Scrambler	CLK Control	Reset	Test Mode 1	Test Mode 0
00 = TX Data 01 = TX Alternate 10 = TX Mark 11 = TX Space 0 = Disable 1 = Enable 0 = Normal 1 = Bypass Scrambler 0 = XTAL 1 = 16 x Data Rate Output at CLK pin in DPSK Mode Only 0 = Disable 1 = Enable 00 = Normal 01 = Analog Loopback 10 = Remote Digital Loopback 11 = Local Digital Loopback									
Detect Register DR	010	x	x	Receive Data	Unscr Marks	Carrier Detect	Answer Tone	Call Progress	Long Loop
Outputs Received Datastream 0 = Condition not Detected 1 = Condition Detected									
TOW Control Register TR	011	RXD output Control	Transmit Guard/ Tone	Transmit Answer Tone	Transmit DTMF Tone	DTMF3	DTMF1/ DTMF2	DTMF0/ Overspeed	DTMF0/ Guard/ Answer/ Tone
RXD Pin 0 = Off 0 = Normal 1 = Weak Pull-Up 0 = Off 1 = On 0 = Off 1 = On 0 = Data 1 = TX DTMF 4-bit Code for 1 of 16 Dual-tone Combinations 0 = 2225 Hz A.T. 1800 Hz G.T. 1 = 2100 Hz A.T. 500 Hz G.T.									
ID Register 10	110	1	0	x	OH	x	x	x	x
00xx = 73K212AL, 322L, 321L 01xx = 73K221AL, 302L 10xx = 73K222AL, 222BL 1100 = 73K224L 11 11 = 73K224BL 11 01 = 73K324R X = Undefined, mask in software 0 = OH Relay Driver Open 1 = OH Open Drain Driver Pulling Low									

Figure 5—The programmer interface to the '222 consists of five registers. Most of the bits are read/write except those in the Detect Register (DR), which are read-only.

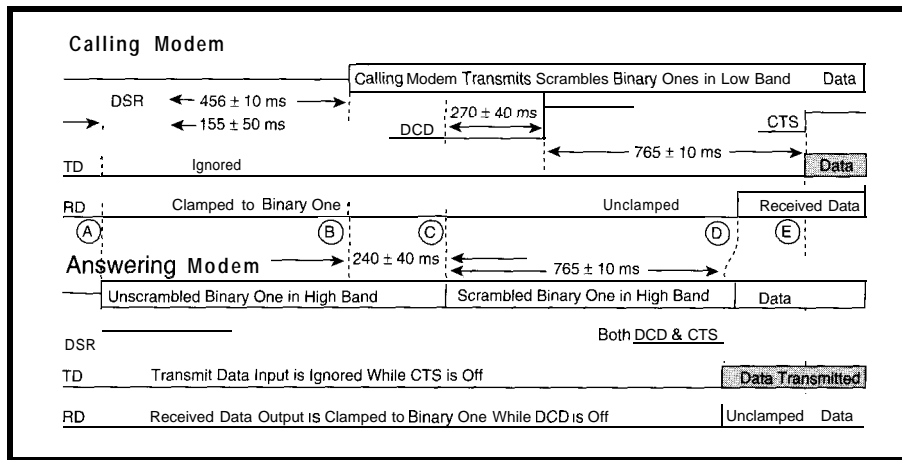


Figure 6—The connection handshake for a V.22 (1200 bps) modem takes a little more than a second. Modems that support many standards have more complex and time-consuming negotiation phases.

poor line quality, meaning fewer errors and retries. Remember the tortoise and the hare?

Shortly, we'll see a rash of small and inexpensive E-mail gizmos in the guise of phones, organizers, or pagers. Whatever costumes they wear, it's for sure they can't afford a Pentium, 28.8 modem, and TCP/IP stack.

In fact, 1200 bps is just about right for something with a 4 x 16 display

and squint-and-peck keyboard. Anything faster is overkill, kind of like filling your fish tank with a firehose.

Priced at only \$7.50 in 10k quantities, a chip like the '222 may be just what it takes. Why pay for more speed than you need? □

Tom Cantrell has been working on chip, board, and systems design and marketing in Silicon Valley for more

than ten years. He may be reached by E-mail at tom.cantrell@circellar.com, by telephone at (510) 657-0264, or by fax at (510) 657-5441.

SOURCES

K-series modem chips
 TDK Semiconductor Corp.
 1435 1 Myford Rd.
 Tustin, CA 92780-7068
 (714) 508-8800
 Fax: (714) 508-8877
www.tsc.tdk.com

DAAs, modem modules
 Cermetek Microelectronics
 406 Tasman Dr.
 Sunnyvale, CA 94089
 (408) 752-5000
 Fax: (408) 752-5004
www.cermetek.com

IRS

425 Very Useful
 426 Moderately Useful
 427 Not Useful

PC COMPATIBLE!

SINGLE BOARD COMPUTERS

Just connect a keyboard, monitor/LCD, a disk drive and your ready to run. Or forget the drive and boot directly from a Flash disk. Add PC/104 Modules for Fax/Modem, SCSI, Ethernet, Digital/Analog I/O, and PCMCIA. Great for Point Of Sale and Web Browsers/Servers. Prices start at \$200.00 Qty. 1.

- *Wide CPU Selection: 386SX, 486DX, DX2, DX4, 586, Pentium.
- *All SBCs have Real Time Clock, Serial, Parallel, IDE, and Floppy.
- *On Board Watchdog Timer.
- *BIOS with Power Saving Green Mode.
- *Wide Bus Selection: PC/104, ISA, PCI.
- *10.4" TFT super bright LCD Panel Kits.
- *Hardware and Cable kits included for most boards.

1985-1997
 OVER
 12
 YEARS OF
 SINGLE BOARD
 SOLUTIONS

EMAC inc.

618-529-4525 Fax 457-0110 BBS 529-5708
 11 EMAC WAY, CARBONDALE, IL 62901
 WORLD WIDE WEB: <http://www.emacinc.com>

REMOTE POWER CARD!

3 VERSIONS:

RESET

WATCHDOG RESETS PC IF IT HANGS FOR HARDWARE OR SOFTWARE REASONS

PHONE

TURN ON PC WITH PHONE, SHARE VOICE / MODEM LINE, CONTROL AC APPLIANCES

TIMER

WAKEUP OR SHUTDOWN PC, LATE NITE BACKUP / MODEM, CONTROL AC APPLIANCES

95\$ 278 OEM



ALL MVS CARDS INCLUDE ASM, BASIC, AND C SOURCE FOR PC OR SBC

a CHAN ADC

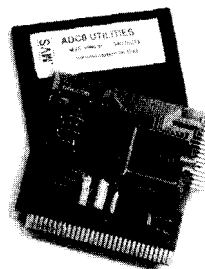
DATA ACQUISITION, SERVO CTL, AUDIO
 8-BIT RESOLUTION 22KHZ SAMPLE RATE
 SHARP CUTOFF ANTI-ALIAS FILTER
 CREATE STEREO BLASTER (.VOC) FILES

95\$

2 CHAN DAC

VOICE MAIL, MUSIC, ALARMS, CTL VOLT
 8-BIT RESOLUTION 44KHZ SAMPLE RATE
 PLAYS MONO / STEREO BLASTER FILES
 FUNCTIONS AS DIGITAL ATTENUATOR TOO

75\$



MVS

MVS BOX 850
 MERRIMACK, NH
 (508) 742-9507

5 YEAR LIMITED WARRANTY
 P I N G I N U S A

CONNECT TIME

conducted by Ken Davidson

The Circuit Cellar BBS
300/1200/2400/9600/14.4k bps
24 hours/7 days a week
(860) 871-1 988-Four incoming lines
Internet: www.circellar.com

The Circuit Cellar BBS Joins the Big League

What do you suppose is one of the most common questions I've been asked via E-mail over the last year or two? "Can I access the BBS using the Internet?" The answer has always been no-until now.

After months of research, experimenting, and procuring, I've finally brought the Circuit Cellar BBS to the Internet. Let me list my original goals:

- Multiple simultaneous users
- Support for a dial-up interface so people can continue to call the modem lines directly
- Allow any desktop platform and operating system to dial in to the system
- Internet E-mail for in-house staff and users
- Access to messages and files from the Internet using either telnet, ftp, or the Web
- Ability to host our Web sites
- LAN connections for in-house staff

The ultimate solution that let me achieve all my goals turned out to be the Wildcat! Interactive Network Server (WINS) from Mustang Software. Now that the new system is on-line, I'll give you a brief overview of what it offers and how to access it.

Features

The Circuit Cellar BBS continues to offer the same basic features that we've supported for the past 10 years-direct access to Circuit Cellar staff and other callers using public and private messages, and access to files related both to past Circuit Cellar articles and to computers and electronics in general. We also offer information about *Circuit Cellar INK* such as how to subscribe, get back issues, or write articles; upcoming editorial events and features; and other Circuit Cellar products.

What's new is how the information can be accessed. As I've mentioned, your options are dial-up, telnet, ftp, and Web.

Dial-up

Calling (860) 871-1988 connects you to a modem that leads into the text-based version of the BBS. ANSI menus continue to be supported, as does flat ASCII for those of you with dumb terminals. I've tried to keep the menu, message area, and file area structures as close as possible to the old system, so veteran callers shouldn't have much trouble navigating.

Telnet

Those of you whose only exposure to the Internet is Web browsing may not be familiar with telnet, a text-based transport protocol that remotely accesses hosts across the Internet. When you run a telnet client on your local system and give it the Circuit Cellar BBS net address, you immediately connect to the text-based version of the system exactly as if you'd dialed into one of the modems. The biggest difference is the lack of long-distance phone charges. Internet users in Australia can use the BBS as easily and inexpensively as those across the street.

You need an Internet connection and a telnet client. Windows 95 and NT have simple ones, and they are readily available for other platforms. Many newer comm programs also come with telnet support (e.g., Qmodem Pro from Mustang and Procomm Plus from Quarterdeck). While I try to get the name server straightened out at our ISP, you may telnet directly to our IP address (206.119.19.172). Eventually, you'll access bbs.circellar.com to get in.

ftp

Most Internet users have used ftp (file transfer protocol) before. Obviously, it transfers files from machine to machine across the Internet. While we've had a minimal ftp site for some time now, it's only had recent article-related files available on it.

We now have all the files from the BBS available via ftp. Point your ftp client or Web browser at <ftp://ftp.circellar.com/> to see what we have to offer.

Web

Perhaps the most exciting new access method is your trusty Web browser. Access all the public messages, your own E-mail, and all the file areas without leaving the comfort of your favorite browser. We do require your browser to support frames (Microsoft Internet Explorer or Netscape Navigator), but that covers most of the world.

When you first connect to our Web address at <http://www.circellar.com/>, you're asked whether you want to access one of our Web sites or the BBS. When you select the BBS, you'll be asked for a user name and password. Enter the information you use to log onto the BBS. New users can also sign up for access.

Once in, check messages, see who else is on, modify user settings, answer questionnaires, and download files.

Join Us

I've been extolling the virtues of the Circuit Cellar BBS for years here in these pages. If the cost of the phone call has kept you from joining us, I encourage you to jump right in. Chances are, you'll either learn something new, teach someone else something new, or both.

PRIORITY INTERRUPT

Processing-A Matter of Definition



As you might have guessed, the real Circuit Cellar is underground. Well, not really-it's the bottom floor of the house, and like a raised ranch, only one end of the house is at ground level. In my case, having ground-level access is moot. I still can't see what's going on outside unless I look at a closed-circuit TV monitor. It's an insulated reality.

Last night, I sat down at my computer and proceeded to judiciously pen an eloquent dissertation on the relative comparisons of various embedded-processor architectures. One eye was on the word-processor screen, the other on the howling April 1st blizzard with 50-mph winds outside. The businessman/publisher side of me considered, "Let's get this over with fast, write the editorial, and get the magazine to press." On the other hand, the untamed/experimenter side of me mused, "Oh boy, disaster time again. Let's see if all these back-up systems really work."

Let's face it. Connecticut isn't exactly wilderness. It's just a matter of what you consider normal. When Atlanta gets 2' of snow, the whole city shuts down. People down Maine (as they call it) think 2' is nothing.

Unfortunately, the definition of "normal" in Connecticut is a moving target. You might as well use a dartboard to predict the weather around here. Oklahoma may be tornado alley, but if memory serves me correctly, two of the three worst tornadoes in U.S. history were in Connecticut.

The good news is that we're a rich state and close to everything. The bad news is that we're a rich state and close to everything. When that one tornado touches down every couple of years, it can't help but land on something expensive. When that thunderstorm rips through the state on a hot August afternoon, every lightning bolt strikes a real target.

After about two hours of retyping the same sentence innumerable ways, attention to the blizzard won out. If it weren't for the generator and E-mail, I could have hoped that being snowbound and without power might afford yet another excuse for a late editorial.

The morning proved to be everything that the night predicted. Up to 2' of heavy snow certainly was not something I considered normal. But, it was normal to consider that it would inevitably happen. This stuff was so heavy that opening the front door was impossible. I opened the garage door and my Westie, **Katy**, went bounding out into the snow. About 5' out, she suddenly realized dogs with 6" of ground clearance don't do well in deep snow. I trudged out and rescued her.

That should have been a message. You know-something like "where dogs fail to tread." While I pride myself on being ready for practically anything, the reality of having to use it all suggests that I might have a little too much wilderness in my urban lifestyle.

The solution was anything but easy. It was one of those situations where you needed everything just to use one thing. Yeah, I know it sounds stupid, but you had to be there.

Our driveway is about 600' long. Usually, I plow it with a light pickup truck. I opened the door in front of the truck, jammed it into 4WD low, dropped the plow, and got about 6'! Undaunted, I went to the next garage door, fired up the 10-hp snow blower and proceeded to clear a path for the plow. That went about 20' before the heavy wet snow made using it an equal aggravation. The third garage door offered the final confidence that machine could ultimately prevail. I climbed up on the tractor and fired up the diesel engine. Here I was, using a tractor to dig out the snow blower to dig out the truck to dig out.... You get the picture.

Well, I guess I won't be discussing bits as they apply to embedded processors this time. Believe it or not, it's possible to get a 2-ton tractor stuck in 2' of snow. Now I'm down to using a shovel to dig out the tractor to dig out, etc., etc. I didn't lie when I told Janice that my editorial was related to the magazine focus. Using a shovel is after all, processing snow-a bit at a time.

steve.ciarci@circellar.com