EMBEDDED PC MONTHLY SECTION

CIRCUIT CELLAR INK®

THE COMPUTER APPLICATIONS JOURNAL

#84 JULY 1997

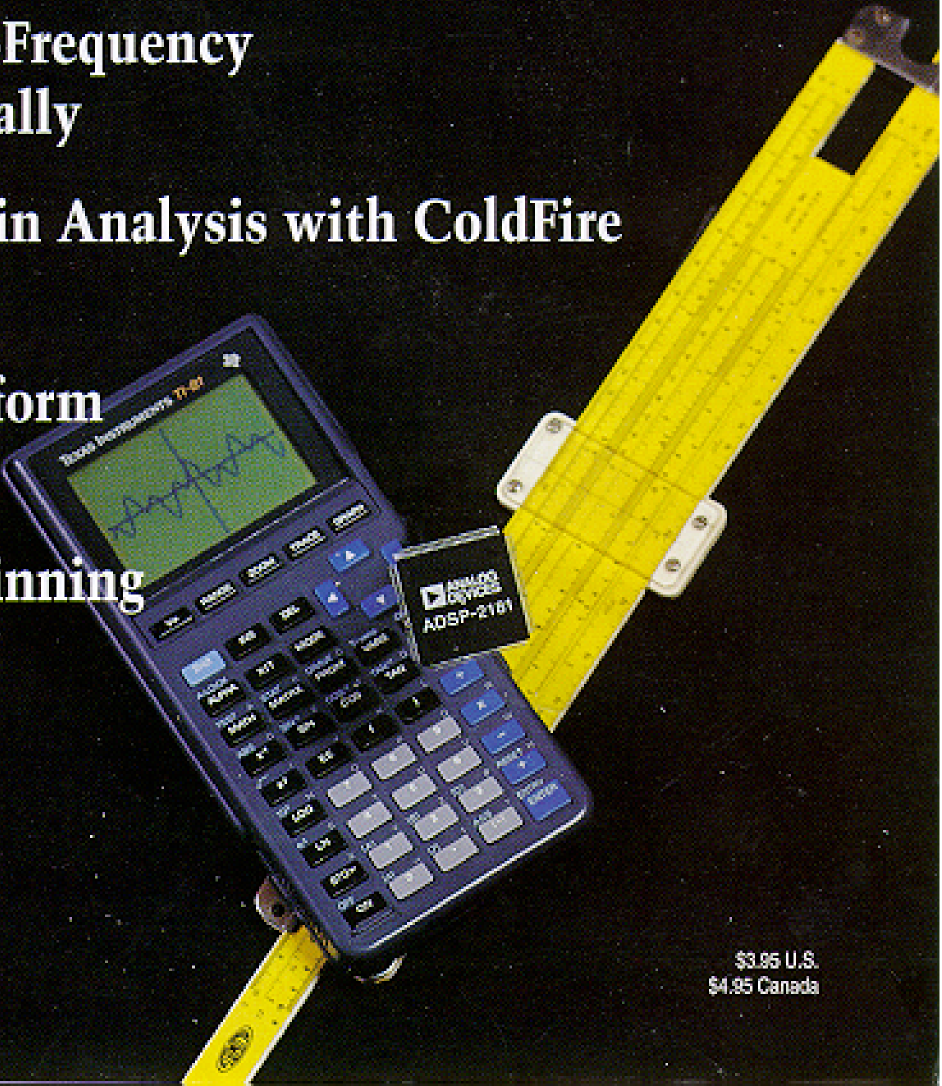
DIGITAL SIGNAL PROCESSING

Generating High-Frequency Waveforms Digitally

Frequency Domain Analysis with ColdFire

EPC: Embedding An Ethernet Platform

EPC Contest—A Formula for Winning



$3.95 U.S.
$4.95 Canada

TASK MANAGER

Unfulfilled Anticipation

O here you sit, hungry, frustrated, mouth watering. You've been thinking about it all day. The last time you were at the local mall's food court, you tasted a free sample of their special sandwich. Now you're back for a more substantive meal, and all you find is "Closed. Out of Business." Sure there are other vendors around, but you'd been really looking forward to this shop's treats.

What does this sad story have to do with Circuit Cellar *INK*? Well, I'm sorry to report that after last month's introduction to a MicroSeries on machine vision ("Machine Vision: Industrial Inspection," *INK* 84), we had to cancel the series. It came down after last month's issue had gone to press, and it was completely out of our hands and those of the author.

In its place, veteran Jan Axelson graciously agreed to fill in with a pair of articles on using serial EEPROMs. She starts this month with a description of the most common serial interfaces (SPI, Microwire, and I^{2}C), and concludes next month with a sample application.

I'm getting ahead of myself, though. Kicking off this month's features, we have an article from a favorite author, David Prutchi, on generating high-frequency waveforms using digital techniques. It's not just a simple matter of sticking in a small processor.

Next, William Hohl and Joe Circello light up ColdFire with some algorithms for doing orthogonal manipulations. Finally, David Tweed finishes up his Canadian timecode receiver project.

In our columns, I've already mentioned Jan Axelson's EEPROM lead-in article. Following Jan, Jeff teaches his robot to speak. And, Tom rifles around the parts drawer and comes out with a handful of new chips that do specific tasks very well.

EPC begins with Chip Freitag describing how to add an Ethernet interface to embedded controllers. With the Internet growing like wildfire, it's only a matter of time before you have an Ethernet backbone in your house with everything connected to it.

Next, David Feldman gives you some pointers for submitting a winning entry to *INK*'s Embedded PC Design Contest. In PC/104 Quarter, Mike Justice and Phil Marshall survey fieldbus interfaces for PC/104. Finally, Fred Eady continues the Internet-connected embedded controller concept by assembling the smallest Web server you've likely ever seen.

[illegible]

editor@circuitcellar.com

CIRCUIT CELLAR INK®

THE COMPUTER APPLICATIONS JOURNAL

EDITORIAL DIRECTOR/PUBLISHER Steve Ciarcia	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	

CIRCUITCELLAR INK®, THE COMPUTER APPLICATIONS JOURNAL (ISSN 0696-6965) is published monthly by Circuit Cellar Incorporated 4 Park Street, Suite 20, Vernon, CT 06066 (660) 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 checkdrawn 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 Dianne Barry
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	MIDWEST&SOUTHEAST	WEST COAST
Barbara (Best) Curley	Christa Collins	Barbara Jones & Shelley Rainey
(561) 694-2044	(954) 966-3939	(714) 540-3554
Fax: (561) 694-2051	Fax: (954) 965-6457	Fax: (714) 540-7103
B.Best-Hajar@worldnet.att.net	HajarChrista@worldnet.att.net	shelley.hajar@worldnet.att.net

Circuit Cellar BBS-24 Hrs., 2400/9600/14.4k bps, 6 bits, no parity 1 slop bit, (660) 871-1988. For information, mail to info@circuitcellar.com. World Wide Web: **www.circuitcellar.com**

INSIDE ISSUE 84

www.circuitcellar.com

READER I/O

CONTESTS AND INTELLECTUAL PROPERTY

As I looked at the rules for entering INK's Embedded PC Design Contest, I was concerned by this rule:

"All contest entry materials.. .become the property of *Circuit Cellar INK* and will not be returned under normal circumstances. All contestants entering projects in the Embedded PC Design Contest agree to assign *Circuit Cellar INK* exclusive first-publication.. .rights....

The winners will receive monetary prizes:

First Prize $5000
Second Prize $3000
Third Prize $2000
Three Honorable Mentions $250 (each)"

I fear that you and your sponsors are using $10,750 to "steal" other people's work and ideas.

TF
ficht@texas.net

First, let me explain that "entry materials" refers only to the paperwork sent in by entrants. You can imagine the logistical nightmare of tracking and returning every bit of paper to its rightful owner. We're simply making sure that we spend our energy making a good magazine, not just shuffling paper.

Secondly, as with all INK *design contests, the intellectual property of all submitted designs remains with the contestants. Rest assured, we won't forward your entry to the sponsoring companies. As well, the judges are bound by a nondisclosure agreement. They cannot use any submitted design to further their own business enterprises.*

This is our ninth design contest, and we have never—and will never-use any submission to manufacture a product. Remember, our interest is in bringing you a high-quality engineering journal. We won't jeopardize our relationship with you by "stealing" your designs.

Editor

RIGHT ON THE MONEY

Steve, thanks very much for your opinion ("When It Costs Nothing, What's It Worth?" INK *79)*! I especially enjoyed it when you asked, "What real value is there in rushing to upgrade to the latest.. .modem when your actual throughput is [less]?"

Bang on. Give the man a cigar!

William F. **Maton**
wmaton@enterprise.ic.gc.ca

ARM TIED BEHIND BACK

Randy Heisch's "A PowerPC 403GA-Based Embedded Controller Prototype" and Art Sobel's "Embedding the ARM7500" *(INK 82)* made for good reading. Art's statement that the ARM7500 is almost an entire PC in a chip may be true, but I think that Motorola has a better solution with their MPC821 CPU.

The MPC821 has a PowerPC as the core with a RISC CPU to handle all I/O. The I/O includes two high-speed serial ports-one set up as an Ethernet AUI port, and the other as an SDLC port. Alternative uses for these two high-speed serial ports would be ISDN channels.

An LCD controller is included along with an IR interface, dual-port PCMCIA controller, I^{2}C bus controller, single-wire serial bus, parallel port, and speaker port. The MPC860 drops the LCD controller in favor of two more normal-speed serial ports. DMA and a memory controller are also included with all MPC8*xx* devices. All this is contained in approximately a 1-in.^{2} ball-grid array package. All MPC8*xx* devices also support JTAG and basic debug ports, so an in-circuit emulator isn't required.

The ARM may have a future, but I think the PowerPC will capture a larger market share-especially in the VME market, where it will overtake the 68k. As for embedded applications, the MPC8*xx* family is an excellent choice.

Dan Farkas
dfarkas@uoft02.utoledo.edu

Contacting Circuit Cellar

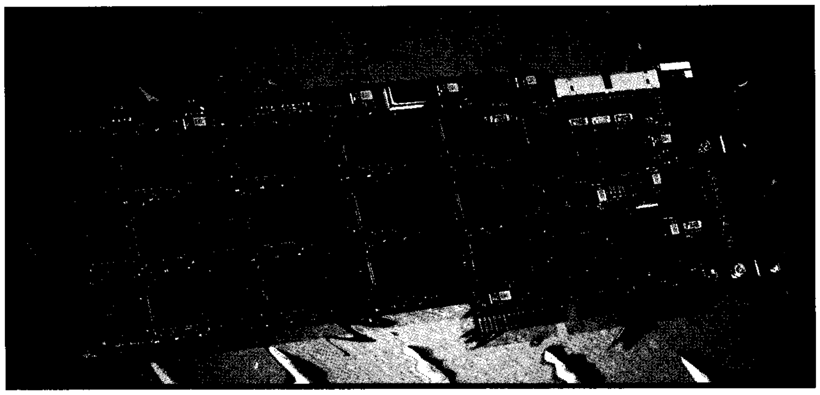
We at Circuit Cellar *INK* encourage communication between our readers and staff, so we have made every effort to make contacting us easy. We prefer electronic communications, but feel free to use any of the following:

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: Editors and regular authors are available to answer questions on the Circuit Cellar BBS. Call (860) 871-l 988 with your modem (300–14.4k bps, 8N1).
Internet: Letters to the editor may be sent to **editor@circuitcellar. corn**. Send new subscription orders, renewals, and address changes to **subscribe@circuitcellar.com**. Include your complete mailing and E-mail addresses 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@circuitcellar.com**.
WWW: Point your browser to **www.circuitcellar.com**.

NEW PRODUCT NEWS

Edited by Harv Weiner

MVIP DSP RESOURCE BOARD



DSP Research has announced the **VIPER-12, a** high-density MVIP (Multi-vendor Integration Protocol) DSP resource board for computer telephony and telecommunications infrastructure applications. The board is an ideal platform for wireless and cellular base stations, remote access servers, voice / modem / fax over ATM / frame relay, and satellite base stations.

Each VIPER- **12** services up to 24 IS- 136 digital cellular vocoders including line echo cancellation, keeping the per-channel hardware cost under $200. For voice-over-network applications, the board can translate, or transcode, between different voice-compression standards. It also services multiple channels of fax / modem connections-a useful feature in Internet remote access, fax-back, and pager servers. The VIPER-12 supports up to 12 simultaneous V.34, 24 fax, or 48 V.22bis connections per board. As an open DSP resource board, its MVIP bus interface gives access to 256 full-duplex 64-kbps channels.

The VIPER- 12 combines the MVIP bus with the power of 12 Texas Instruments TMS320C542 DSPs. At 40 MIPS of performance, the DSPs allow multiple channel or port assignments. With 12 DSPs per board, the VIPER-12 has an extremely high channel density.

The VIPER-12 is supplied with host APIs for the MVIP switch control and DSP-host communications, plus the QuickSTART DSP operating environment. The full complement of development tools includes the TI C compiler with assembler and linker, a DSP program loader, and GO DSP's CodeComposer Debugger.

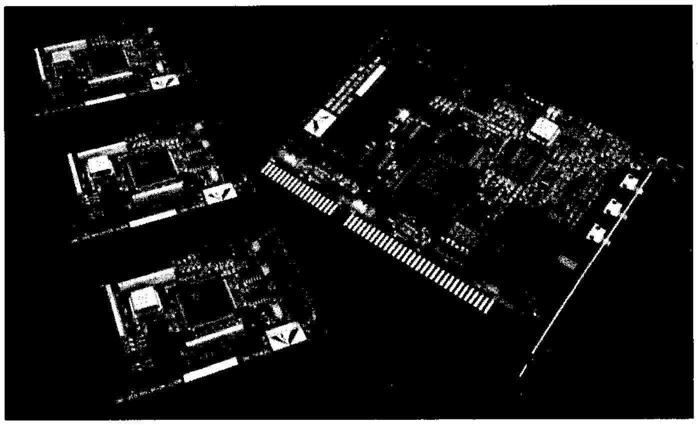
Single-board pricing for the VIPER-12 starts at $4995.

DSP Research
1095 E. Duane Ave., Ste. 203
Sunnyvale, CA 94086
(408) 773-l 042 • Fax: (408) 736-3451
www.dspr.com

#501

DSP UNIVERSAL EVALUATION MODULE

The **Mountain-Uevm** enables engineers to evaluate different fixed-point Texas Instruments DSP chips for various applications without the cost and time commitment of traditional prototyping. Interchangeable DSP modules (daughter cards) facilitate chip selection.



The Mountain-Uevm supports all of TI's popular fixed-point DSPs, including the TMS320C2*xx*, 'C5*x*, and 'C54*x* families, as well as the new 'C24*x* motor-control family. The half-size PC / AT plug-in card features an FCC-approved telephone Data Access Arrangement (DAA), 16-bit stereo audio interface codec, daughter-card site to accommodate different DSPs (**Mountain-Paks),** and debug using both on- and off-card emulation.

The Mountain-Uevm is priced at $995. Mountain-Pak modules for various fixed-point DSPs cost $495 each, including debugger software. Optional TI C / assembly source debugger and debugging environments from GO DSP are available.

White Mountain DSP
20 Cotton Rd.
Nashua, NH 03063
(603) 883-2430 • Fax: (603) 882-2655

#502

NEW PRODUCT NEWS

LOW-COST DSP CHIP

Analog Devices' **ADSP-21061** SHARC DSP features the same high-performance processor core as the current SHARC DSP family but costs under $100. It has l-Mb on-chip SRAM, six DMA channels, two serial ports with 240-MBps I/O (40 Mbps bidirectional), and the same pin-out as the ADSP-21060 and '62 DSPs. It runs the same source code, operates from a +5-V power supply, and is packaged in a 240-lead PQFP (plastic quad flatpack).

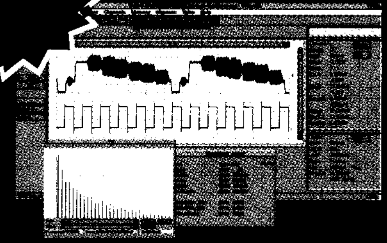
The 32-bit floating-point DSP core runs at 120 MFLOPS with a 25-ns instruction-execution time. Memory is organized in two banks for both dual operand fetches and independent core and DMA fetches. The dual-ported memory enables all I/O to occur in parallel with the core processing unit. The host/external port interfaces with up to 4 Gwords of off-chip memory, other peripherals, other SHARC processors in a cluster, and a host processor.

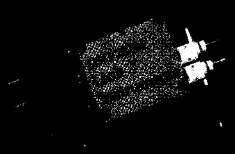
The **EZ-Kit Lite** development kit for SHARC DSPs ($179) offers a hardware platform and C compiler. The tool set includes a '6 I-based add-in board, optimizing ANSI C compiler, code compactor, assembler, linker, loader, instruction-level simulator, and run-time library. Numerical C extensions in the C compiler enable easy coding and fast execution of vector and matrix operations.

Analog Devices, Inc.
P.O. Box 9106 • Norwood, MA 02062-9106
(617) **329-4700 • Fax: (617) 329-1242**
www.analog.com **#503**

Digital Oscilloscopes

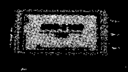
Logic Analyzers





- 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

$499



For $499 you get the model DSO-2102 Scope, Probes, Interface Cable, AC Adapter, and Windows and DOS Software.

- 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) $3799
All prices include Probes and Software

- 40 to 160 channels
- up to 500 MHz
- Variable Threshold
- 8 External Clocks
- 16 Level Triggering
- 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
LA45160-128K (500MHz, 160Ch)	$7000

All prices include Pods and Software
Call for information on our
100 MSa/s Pattern Generator option

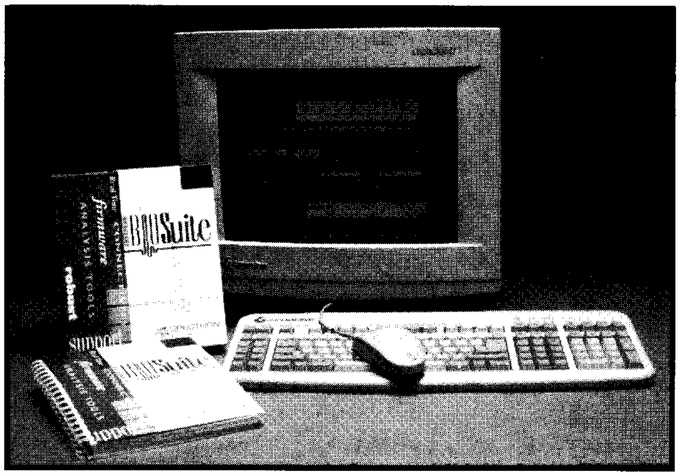
Link Instruments (201) 808-8990
369 Passaic Ave • Suite 100 • Fairfield, NJ 07004 • Fax (201) 808-8786
Web: http://www.LinkInstruments.com/cci7 · Email: Sales@LinkInstruments.com

#104

NEW PRODUCT NEWS

DSP/BIOS

The first commercial implementation of Texas Instruments' new standard DSP/BIOS API (Application Program Interface) has been announced by Spectron. **BIOStation** features real-time multitasking, capture, and statistics facilities to provide a standard foundation for interoperable DSP software-development tools.



The BIOStation Evaluation Kit consists of two components—a target-resident embedded kernel that implements the functionality specified in the DSP/BIOS API and a suite of visual host-based real-time analysis, monitoring, and capture tools that enhance BIOStation kernel services.

The BIOStation kernel provides a standard embedded platform with preemptive multitasking, I/O, capture, and statistics services. In addition to simplifying application development and supporting multivendor tools interoperability, it provides a means for automatically instrumenting applications and providing real-time information to host tools as they execute.

Host-based tools augment kernel services embedded in the target application to provide real-time analysis, performance monitoring, and event-/data-capture facilities. These facilities can be used by a broad range of DSP development tools, including debuggers, emulators, and manufacturing-test, timing-analysis, and diagnostic tools.

BIOStation will initially be available for the TI TMS320C54-based EVM board and DPS Research's C54-based Tiger54. The BIOStation targeted development kit costs **$995** and will be available from TI and its authorized distributors.

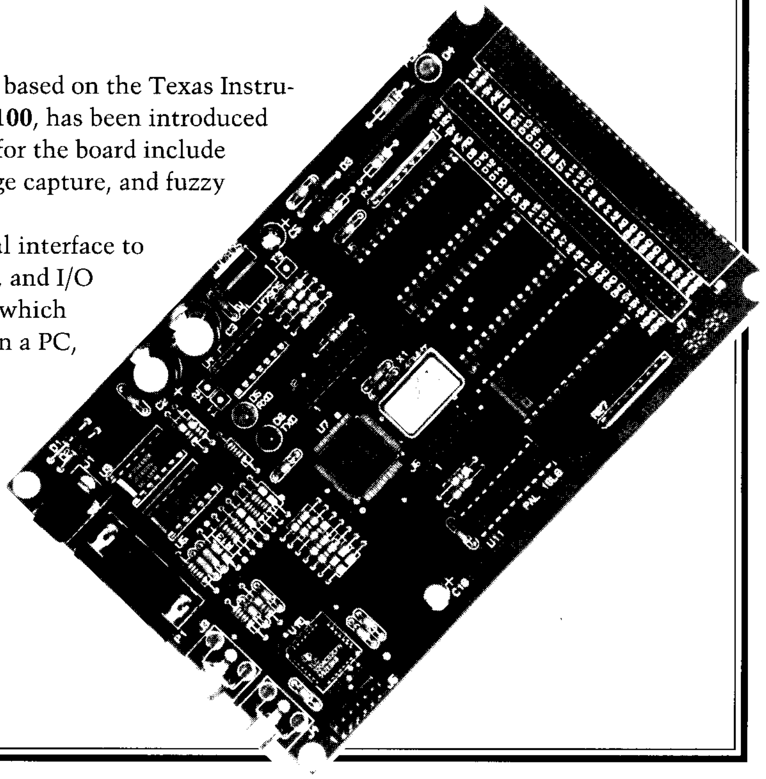
Spectron Microsystems, Inc.
315 Bollay Dr.
Santa Barbara, CA 93117
(805) 968-5100 • Fax: (805) 968-9770 • www.spectron.com

#504

LOW-COST DSP COMPUTER

A **digital signal processing single-board computer**, based on the Texas Instruments TMS320C5x DSP chip and selling for under **$100**, has been introduced by Advanced Microcomputer Systems. Applications for the board include high-speed communication, speech recognition, image capture, and fuzzy logic.

The board features a monitor EPROM, RAM, serial interface to PC, analog input and output, a built-in power supply, and I/O connector for expansion. EZ-DSP Manager software, which enables the user to edit, assemble, and debug codes on a PC, is also included.



Advanced Microcomputer Systems, Inc.
1460 SW 3rd St.
Pompano Beach, FL 33069
(954) 784-0900
Fax: (954) 784-0904
www.gate.net/~ams

#505

FEATURES

FEATURE ARTICLE

David Prutchi

Digital Generation of High-Frequency Waveforms

Cookbook waveforms from a function generator don't always provide the signals you need for testing. David shows how a numerical array and a DAC can yield nonstandard waveforms with arbitrary complexity.

In evaluating the behavior of signal-processing or control circuitry, it's common to use an analog function generator to produce the necessary test input signals.

Typical cookbook waveforms are used to investigate the circuit's behavior when stimulated by sine, square, and triangle waves of different amplitudes and frequencies.

In many applications, however, repetitive sine, square, and triangle waves seldom represent the signals the equipment under test can process.

For example, the heart's electrical signal is a waveform consisting of a complex mixture of these basic wave shapes intertwined with intermittent baseline segments.

Since a constant "live" feed of such signals may be impractical or dangerous for testing biomedical equipment, dedicated signal sources synthesize waveforms like those generated by their physiological counterparts.

Similar requirements are needed for generating test signals of video, radar, disk access, and other waveforms that can't be simulated by simple sines, ramps, or square waves.

Today, nonstandard real-world stimuli waveforms can be easily created as a numerical array and played back through a DAC to yield analog waveforms of arbitrary complexity. This is the operating principle of an Arbitrary Waveform Generator (arb).

Despite the concept's simplicity, a PC program that copies digital values stored in an array into a DAC severely limits the maximum frequency of spectral components for the arbitrary signal. Even an assembly program copying the contents of sequential RAM addresses to an I/O location results in DAC writing rates of a few megapoints per second at most.

Instead of having a DAC interfaced to memory through a processor, arbs have dedicated RAM interfaced directly to the DAC. So, update rates are limited only by the RAM's access time and the DAC's speed. As such, commercial arbs can be purchased with maximum writing rates around 1 Gpoint/s, yielding bandwidths of up to 500 MHz.

In this article, I discuss two simple but versatile waveform generators that can be programmed from a PC printer port. The first is a circuit that generates a sine wave by direct digital synthesis.

The second is an arb which, once loaded with a digital-data array, acts as a stand-alone instrument delivering two simultaneous analog signals. Each signal has an amplitude resolution of 12 bits and variable temporal resolution down to 50 ns (20 Mpoints/s).

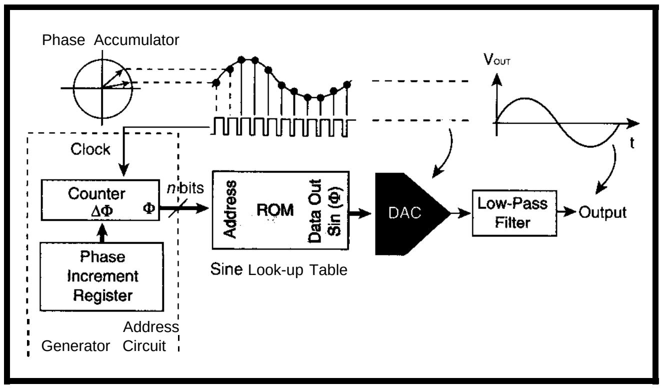


Figure 1 —*In a Direct Digital Synthesizer (DDS), an address generator circuit or phase accumulator controls how samples stored in a ROM lookup fable are delivered to the DAC's* **input. Control over** *the output frequency is achieved by selecting an appropriate phase-accumulator increment.*

DIRECT DIGITAL SYNTHESIS

At its core, a generator that can directly synthesize an analog signal from digital data has memory containing the full digital time domain of the desired waveform. To generate an analog signal, the discretized point-by-point version of the waveform is played sequentially through the generator DAC.

A simple form of this generator is a Direct Digital Synthesizer (DDS). As shown in Figure 1, an address-generator circuit controls how samples stored in ROM are delivered to the DAC's input. On each clock pulse delivered to the address generator, a new address is issued to the ROM so data for the next point in the sequence goes to the DAC.

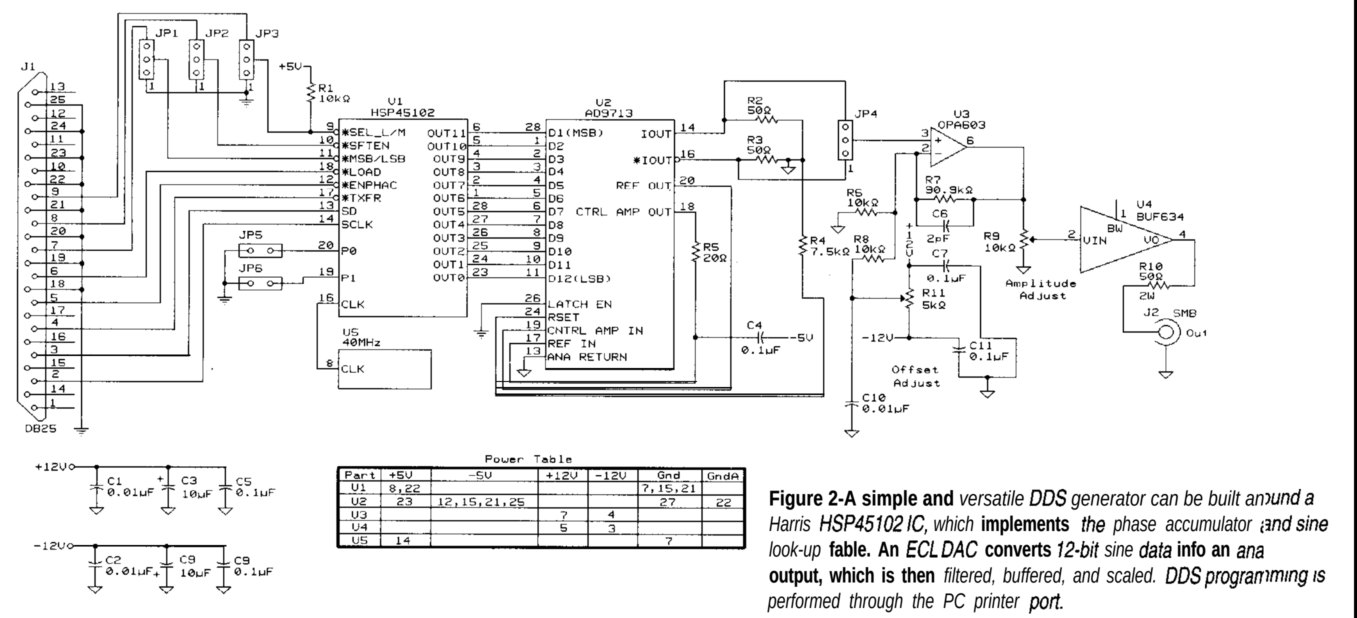
The ROM in a DDS generator usually contains data for a complete single cycle of a sinusoidal waveform. The address generator is a simple counter. Its addresses make up the phase angles ϕ of the sinewave sin(@) samples in ROM. A DAC translates into an analog sinewave the series of values of this ROM look-up table as a function of incrementing phase angles.

Obviously, if the clock presented to the phase-accumulator counter remains constant, then the phase-generation rate does too. The end result is a sinewave of a specific frequency.

However, DDS generators can vary the sine output frequency without altering clocking frequency by programming the phase increment value Δϕ. If the phase-accumulator output increments by Δϕ on each incoming clock pulse, then the output sinewave's frequency is given by:

$$f_{sine} = \frac{f_{clock}\,\Delta\phi}{360^\circ}$$

The frequency resolution f_r of a DDS generator is defined by the n bits of the phase-accumulator increment register and the clocking frequency:



Part	+5V	-5V	+12V	-12V	Gnd	GndA
U1	8,22				7,15,21	
U2	23	12,15,21,25			27	22
U3			7	4		
U4			5	3		
U5	14				7	

Figure 2-A simple and *versatile DDS generator can be built around a Harris HSP45102 IC, which* **implements** *the phase accumulator and sine look-up* **fable. An** *ECL DAC* **converts** *12-bit sine data* **info an** *ana* **output, which is then** *filtered, buffered, and scaled. DDS programming is performed through the PC printer port.*

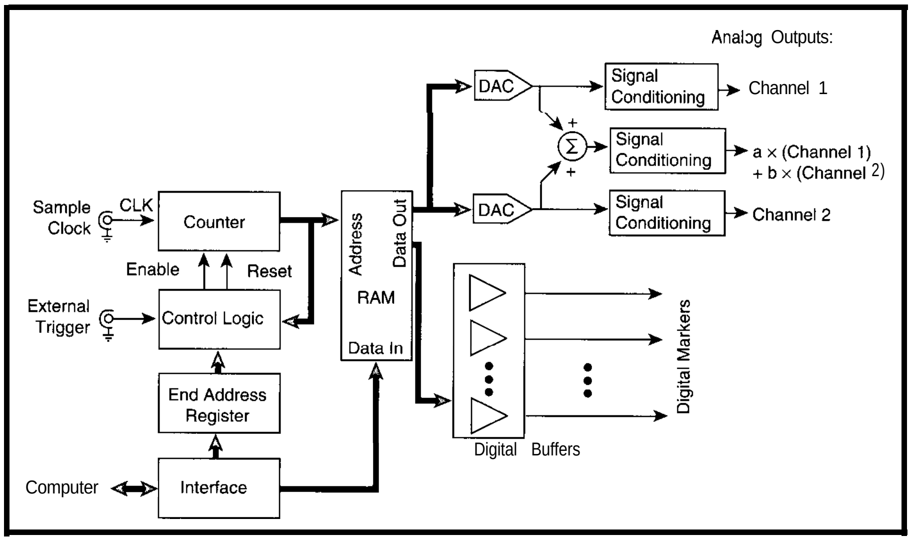


Figure 3-An *arbitrary waveform generator has at ifs core a RAM containing the full time-domain digital representa-fion of the desired waveform. To generate the analog signal, the discrefized point-by-point version of the waveform is played sequentially through the generator's DAC.*

$$f_I = \frac{f_{clock}}{2^n}$$

and the output frequency is directly set by the value *W* of the phase-accumulator increment register:

$$f_{sine} = \frac{W\, f_{clock}}{2^n}$$

Since wide registers, large counters, and ample ROMs are easily integrated, IC DDS generators can now generate sinewaves into the hundreds of megahertz with incredibly high resolution.

In Figure 2, for example, a Harris HSP45 102 IC implements the phase accumulator and sine look-up table. This 32-bit-wide phase-accumulator increment register accepts clock frequencies up to 40 MHz. So, the DDS IC can provide data to generate sinewaves from 0.009 Hz up to 20 MHz with a resolution of 0.009 Hz!

The sinusoidal signal at the DAC's output is not infinitely pure. The digital samples translated by the DAC are quantized in both time and amplitude, so some distortion is introduced.

Time quantization results from the fact that the signal can only change at specific time intervals dictated by the clock. Amplitude quantization results from the discrete nature of the digital system itself. Samples of the infinitely continuous series of a sine are stored in ROM with finite resolution.

Obviously, time-quantization errors are reduced by using as large a look-up table as possible. For the HSP45102, the look-up table is 8 192 samples wide.

Since the number of samples used to reconstruct the sinusoidal wave is the ratio of the clock frequency (40 MHz) and the selected output frequency, time-quantization errors worsen as the selected output frequency increases.

Voltage-quantization errors, on the other hand, are reduced by increasing the width of the data word presented to the DAC. Since price and complexity of a high-frequency DDS circuit increase with the DAC's resolution, a number of projects use only 8-bit video DACs to gain simplicity. But, that doesn't take full advantage of the HSP45 102's 12-bit amplitude resolution [1,2].

In the DDS circuit of Figure 2, a 12-bit TTL-input-compatible ECL DAC makes full use of Ul's data-word width. High-frequency harmonics generated by aliasing are low-passed by U3.

In more sophisticated systems, a steep digitally-tuneable low-pass filter passes the selected fundamental frequency and rejects the sampling aliases.

Using an appropriate low-pass filter (e.g., an elliptic filter) is critical to get clean output at high frequencies since steps become increasingly large and the DAC output resembles a sine-wave less and less. For example, while a 40-kHz output signal uses 1000 samples per cycle, a 13.33-MHz signal is generated using barely 3 samples per cycle!

The HSP45102 includes two 32-bit phase-accumulator increment registers. A digital input on pin 9 selects which register is used at any given time for sinewave generation, enabling direct frequency-shift keying (FSK) modulation of the output.

In addition, the DDS generator enables the phase to be changed on-the-fly by selecting the state of the PO and P1 lines (pins 19 and 20) as shown in Table 1. This enables direct quadrature phase shift keying modulation (QPSK).

These features open up tremendous possibilities for DDS generators in communications applications. A high-stability carrier can be generated via digital circuitry and a low-cost fixed-frequency digital crystal oscillator, and direct digital modulation is possible.

Program the HSP45 102 by loading 64 bits of data for the two phase-accumulator increment registers through the data input pin (SD) in serial format. While keeping the shift-enable (*SFTEN) pin low, each data bit is fed by a rising edge on Ul's clock input pin (SCLK).

Sinewave generation is turned on and off via the *ENPHAC pin. The *TXFER input line controls the transfer of the phase-accumulator increment register selected by the SEL_L/*M line (pin 9) to the phase accumulator's input register.

Here, I retained printer-port pin use compatibility with a DDS generator. (Control software is freely available [2].)

ARB BASICS

As you see in Figure 3, an arb shares the basic building blocks of a DDS generator. Instead of a ROM sine look-up table, however, a full time-domain digital representation of the arbitrary waveform is downloaded into RAM.

As well, the counter is not thought of as a phase accumulator. You can arbitrarily define the last data point of the waveform cycle (end address). Thus, the waveform can be replayed by looping from the last point to the address of the RAM location for the first point.

P1	P0	Phase Shift
0	0	0°
0	1	90°
1	0	270°
1	1	180°

Table l--The *HSP45102's input lines PO and P1 (pins 19 and 20) control the introduction of a phase offset to the phase accumulator's output*

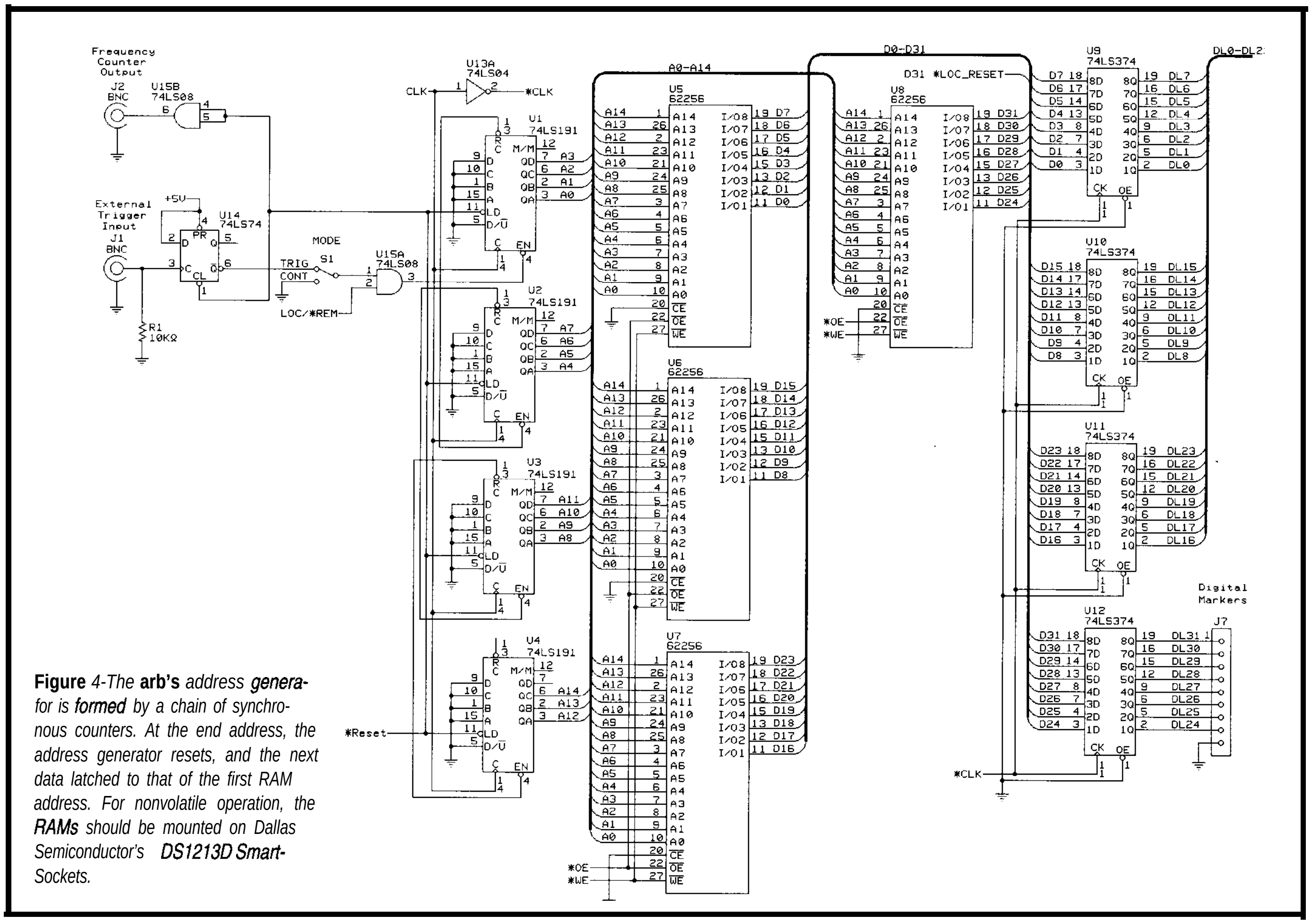


Figure 4-The arb's address generator is formed by a chain of synchronous counters. At the end address, the address generator resets, and the next data latched to that of the first RAM address. For nonvolatile operation, the RAMs should be mounted on Dallas Semiconductor's DS1213D Smart-Sockets.

For some applications, the waveform may be issued only once after a trigger event. Additional circuitry in the address generator receives a trigger signal that allows addresses to be cycled once between the beginning and end of a waveform sequence for every triggering event. A typical application is the testing of ultrasonic echo systems, where the arb-generated echo must be synchronized to the excitation of the transmitting transducer.

As well, instead of maintaining the clock-rate constant and jumping over sample points to change the period of a cycle, an arb's clock frequency is programmable. Thus, the waveform can be compressed or expanded through time, resulting in a controlled shift in frequency of all spectral components of the waveform.

Of course, reproducing a signal requires the stored waveform to be sampled at a rate of at least twice its highest frequency component. In turn, the complexity and time duration of the reproduced waveform are limited by the arb's memory size (depth).

The output waveform's time duration is:

$$T_{waveform} = \frac{\#\text{ of waveform pts.}}{f_{clock}} = \frac{\text{end addr} - \text{start}}{f_{clock}}$$

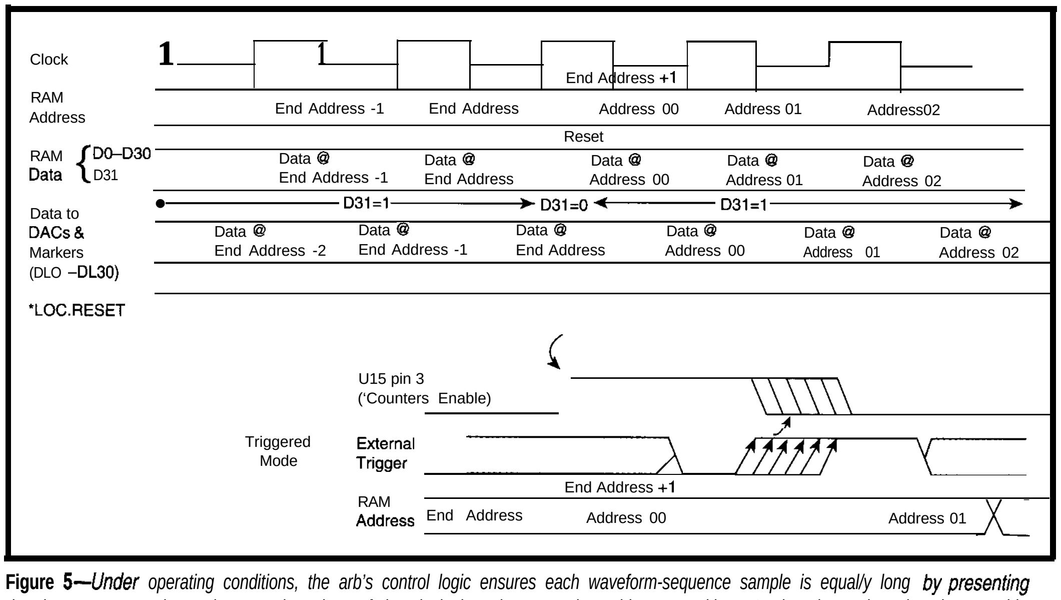


Figure 5—Under operating conditions, the arb's control logic ensures each waveform-sequence sample is equally long by presenting the data contents only on the opposite edges of the clock than those causing address transitions. In the triggered mode, trigger ambiguity is less than one clock cycle.

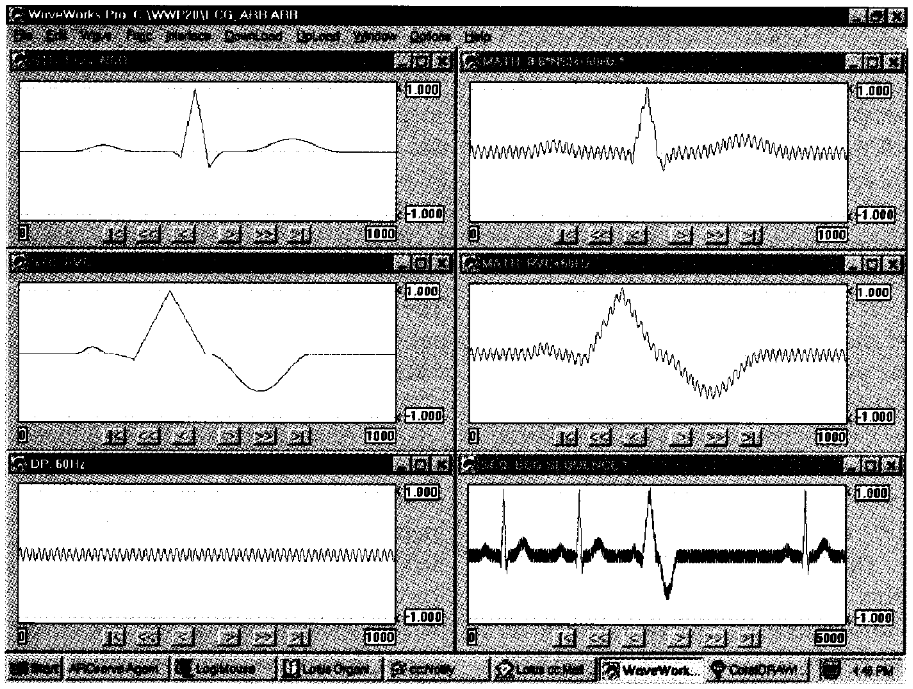

WaveWorks Pro C:\WWP20\ECG_ARB.ARB
File Edit Wave Page Interface Download Upload Window Options Help
1.000
-1.000
0
1000
5000
DP. 60Hz
Start

Photo 1—*It's* **easy to create complex** *waveform with Pragmatic Instruments WaveWorks Pro. Here's a signal for testing electrocardiography equipment by defining the waveform's basic components from predefined templates.*

Speech, for example, requires a sampling speed of -8 kS/s (where S stands for samples). An arb with a depth of 32 kS suffices for only 4.096 s of recording.

With fixed memory size, longer waveform durations are only achieved by limiting the bandwidth to allow lower sampling rates. Obviously, limiting the bandwidth reduces the number of spectral components available to describe the waveform's details.

If an arb has more than sufficient memory to generate a waveform, additional memory can be used for a second waveform channel. Since both channels are generated using a single clock, the two output waveforms are precisely synchronized. This capability is essential for testing instruments that derive their measurements from the phase relationship between two signals.

Also, purely digital lines (i.e., marker channels) can synchronize and position markers coincident with specified points of the arb waveform. They can trigger external instruments (e.g., oscilloscopes) at specific times within the arb waveform cycle.

However, an additional channel's greatest advantage is the possibility of summing both channels. Two synchronized arbitrary components of a single waveform can be independently controlled, making it possible to test the effect of each system component.

To study a circuit's immunity to an unwanted phenomenon, channel **1** can be loaded with the waveform normally seen by the system under test. Channel 2 can be loaded with the anomaly at the desired time within the normal waveform. By varying the gain of channel 2, you can adjust the anomaly's amplitude without changing the amplitude of the normal signal.

Summing arb channels extends the dynamic range of the combined signal beyond the maximum dynamic of each independent channel. Altering the gain of the summed channels makes it possible to generate large signals with very small features on them.

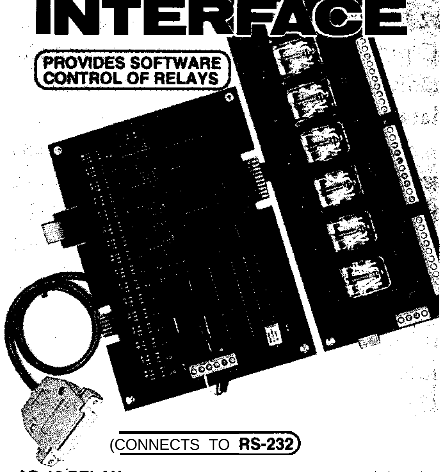
Here, macroscopic changes occupy the full dynamic range of one channel. The smaller waveform details occupy the other channel's full dynamic range. By correctly ratioing the gains between the channels, the summed signal can have a theoretical maximum resolution equal to the sum of the independent channels' resolutions.

PC-PROGRAMMABLE ARB

A simple arb can be built with standard SRAMs, a few counter ICs, some glue logic, and DAC ICs. In this project, three 32 K x 8-bit RAMs store two

RELAY INTERFACE

PROVIDES SOFTWARE CONTROL OF RELAYS

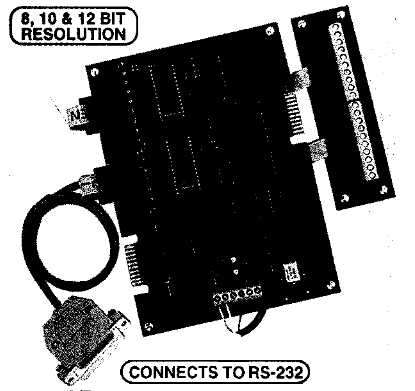


(CONNECTS TO **RS-232**)

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 (6 relays, 10 **VA)......$ 49.95**
RH-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)..$** 99.98
ADC-8G AID CONVERTER* (6 channel/IO **bit).$124.90**
Input voltage, amperage, pressure, **energy usage,** light, joysticks and a wide variety of other types of analog **signals.** RS-422/RS-485 available (lengths to **4,000').**
Call for info on other A/D configurations and 12 bit **converters** (terminal block and cable sold **separately).**
Includes Data Acquisition software for Windows 95 or 3.1
ADC-8E TEMPERATURE INTERFACE' (6 **ch)..$ 139.95**
Includes term. block & 8 temp. sensors (-40' to 146° F).
STA-6 DIGITAL INTERFACE' (6 **channel).........$ 99.95**
Input on/off status of relays, switches, HVAC equipment, security devices, keypads, and other devices.
PS-4 PORT SELECTOR (4 channels **RS-422)....$ 79.95**
Converts an RS-232 port into 4 selectable RS-422 **ports.**
CO-422 (AS-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** anal0g inputs or up to 128 temperature inputs using the PS-4. EX-16, 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:** (614) **464-9656**
Use for information. **technical** support **& orders.**

ELECTRONIC ENERGY CONTROL, **INC.**
380 South **Fifth** Street, **Suite 604**
Columbus, **Ohio** 43215-5491,

#107

12-bit waveforms. An additional RAM IC provides 7 marker channels, and the additional bit encodes the last valid data sample of a waveform sequence.

As Figure 4 shows, the 15-bit address generator of the arb is formed by a chain of 74LS191 synchronous counters (U1–U4). The counter chain's output is sent to 50-ns access-time SRAMs (U5–U8).

From the timing diagram in Figure 5, as long as U1 is enabled, each clock pulse supplied in parallel to all counter ICs advances the address. This process continues until the address points to a data element (D31) on U8, in which bit 7 is low, causing the asynchronous reset of the counter chain.

The arb's circuitry ensures that each sample of the waveform sequence has equal length. Data contents presented on the RAM data bus (DO-D30) are latched on edges of the clock opposite to those causing address transitions.

Since the data at the output of latches U9–U12 lags the data of their inputs by half a clock cycle, the reset signal issued when the counters reach END_ADDRESS+1 (where bit 7 of U8 is low) resets address to zero without upsetting the data related to END-ADDRESS.

The clock line's next falling edge causes the data contents of the first RAM address to be sent to the latches' output. While the time that the first address is available is shorter than for any other address, the corresponding data is available at the output for the same amount of time as other addresses.

When triggered, rather than continuously cycling through the waveform, flip-flop U14 controls Ul's enable line via switch S 1. In the triggered mode, the flip-flop's ● Q output goes low when enabled by the rising-edge of a trigger pulse at its clock input line.

This state is maintained until reset at the end of the waveform cycle by the same reset pulse that zeroes the counter chain. In this mode, trigger ambiguity is less than one clock cycle.

A simple, software-implemented serial protocol downloads and uploads RAM waveform data from and to the PC through the printer port. On the arb, the chain of 74LS323 ICs (U16–U19) of Figure 6 forms a 32-bit serial-to-parallel and parallel-to-serial converter.

When the remote mode is selected by the computer (digital low on bit 1 of the printer port's output port), LOC/*REM goes low, causing U20 to transfer control of the clock (CLK), address generator reset (*RESET), RAM output-enable (● OE), and RAM write (● WR) to the lines of the printer port.

The 74LS323s' mode-control lines (pins 1 and 19) select between hold, shift left or right, and parallel load of the bits of the chain's 32-bit register. Data is clocked serially into U16 and shifted down the chain towards U19 by each rising edge of the serial clock line (SCLK).

Once a complete 32-bit word is in the chain's register, U16–U19 drive the RAM data bus and a write strobe stores the register's contents in the current address. The address generator advances, and the cycle repeats to store successive waveform data points. Data can be read from RAM into the computer by reversing this process.

Once an address is selected, data loads from the RAM data bus into the register formed by the chain U16–U19. The register's contents are then shifted out of U16 into one of the printer port's status input lines (pin 10 of J4).

Two different DACs work with the arb. An Analog Devices AD9713 high-speed ECL DAC capable of updating its output at up to 100 MS/s restores high-frequency signals with high resolution [see Figure 7). Alternatively, the lower cost AD667 offers more limited performance for applications with DAC writes of no more than 300 kS/s.

Unfortunately, it's difficult to take full advantage of the AD9713s. With high-speed DACs, the arb's speed is limited by the RAM's access time. Under this arb's direct addressing architecture, RAMs with 50-ns access time allow a maximum writing speed of:

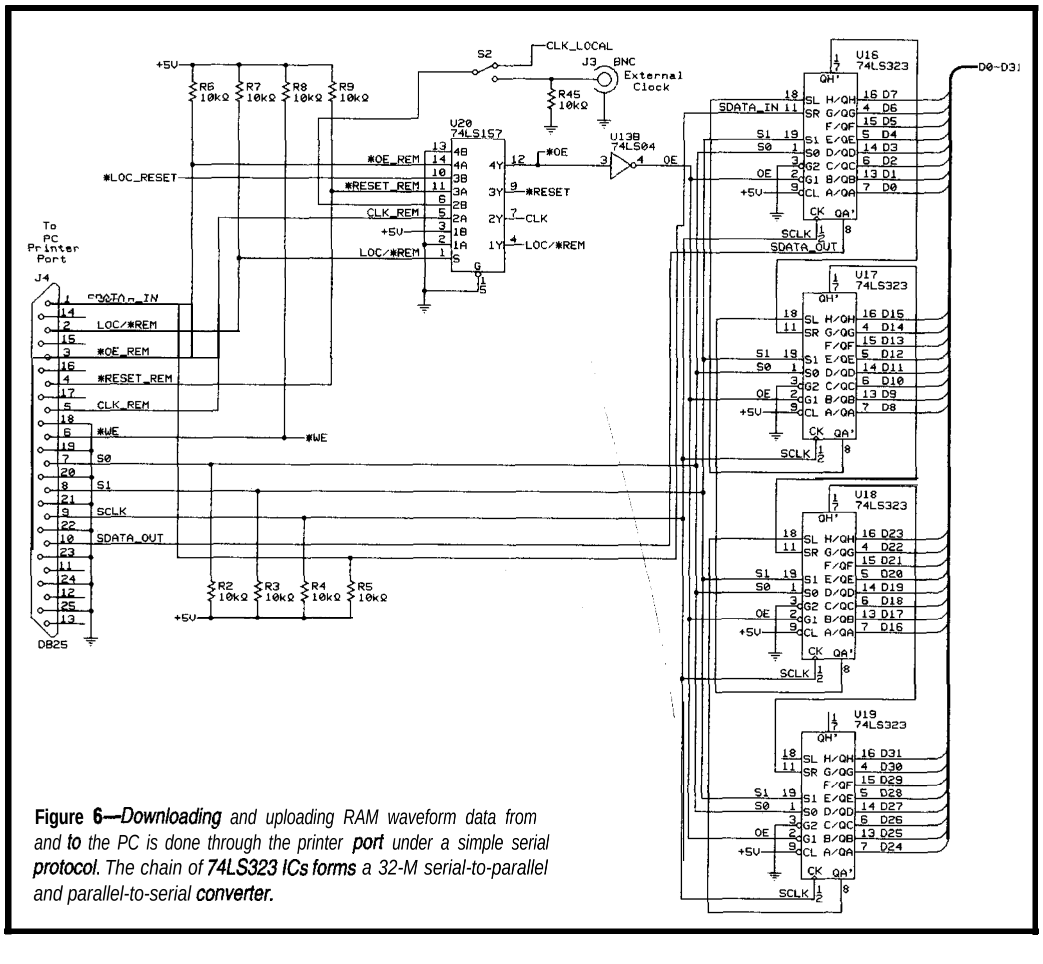


Figure 6—Downloading and uploading RAM waveform data from and to the PC is done through the printer port under a simple serial protocol. The chain of 74LS323 ICs forms a 32-M serial-to-parallel and parallel-to-serial converter.

$$\frac{1}{50 \times 10^{-9}\text{s}} = 20 \text{ MS/s}$$

Achieving 100 MS/s writing speeds requires 10-ns RAMs. Although they're available [e.g., cache RAM], they are very costly, limited in size, and generally power hungry. Rather than using a direct addressing scheme, very high-speed arbs overcome the RAM's access-time shortcomings by operating several RAM banks in parallel.

In this multiplexed address scheme, one or more RAM banks are accessed and allowed to settle while current data is taken from a different RAM. As the address updates, data is taken from a RAM that already has valid data available.

A 4:1 multiplexed memory arb uses four low-cost 50-ns RAMs to achieve 80 MS/s. I decided against the more complex multiplexed approach since 20 MS/s provides sufficient flexibility in generating relatively low-frequency signals to test biomedical instruments.

Once analog signals are at the DAC outputs, the circuit offsets and scales them prior to buffering them for output. A summing channel is also provided to expand the arb's versatility.

The local sampling clock is generated by U33, Maxim's MAX038 high-frequency waveform generator IC. Although this IC typically acts as a function generator, in Figure 8, it's an oscillator whose frequency can be controlled from 20 Hz to 20 MHz. Alternatively, the sampling clock may be supplied by an external TTL-level clock through connector J3 and switch S2.

The arb's circuitry requires +5 V for the logic circuitry, -5 V for the ECL logic of the high-speed DACs, and ±12 V for the analog circuitry. The power supply in Figure 9 generates these voltages from a 12-VAC input.

The arb loses waveform data as soon as power is removed. For nonvolatile operation, the RAMs may be mounted on Dallas Semiconductor's Smart-Socket DS1213D intelligent sockets.

Remember, these sockets are. designed to be compatible with RAMs of up to 128 K x 8. So, when using the DS1213Ds, you need four more PCB pads than those required for each RAM

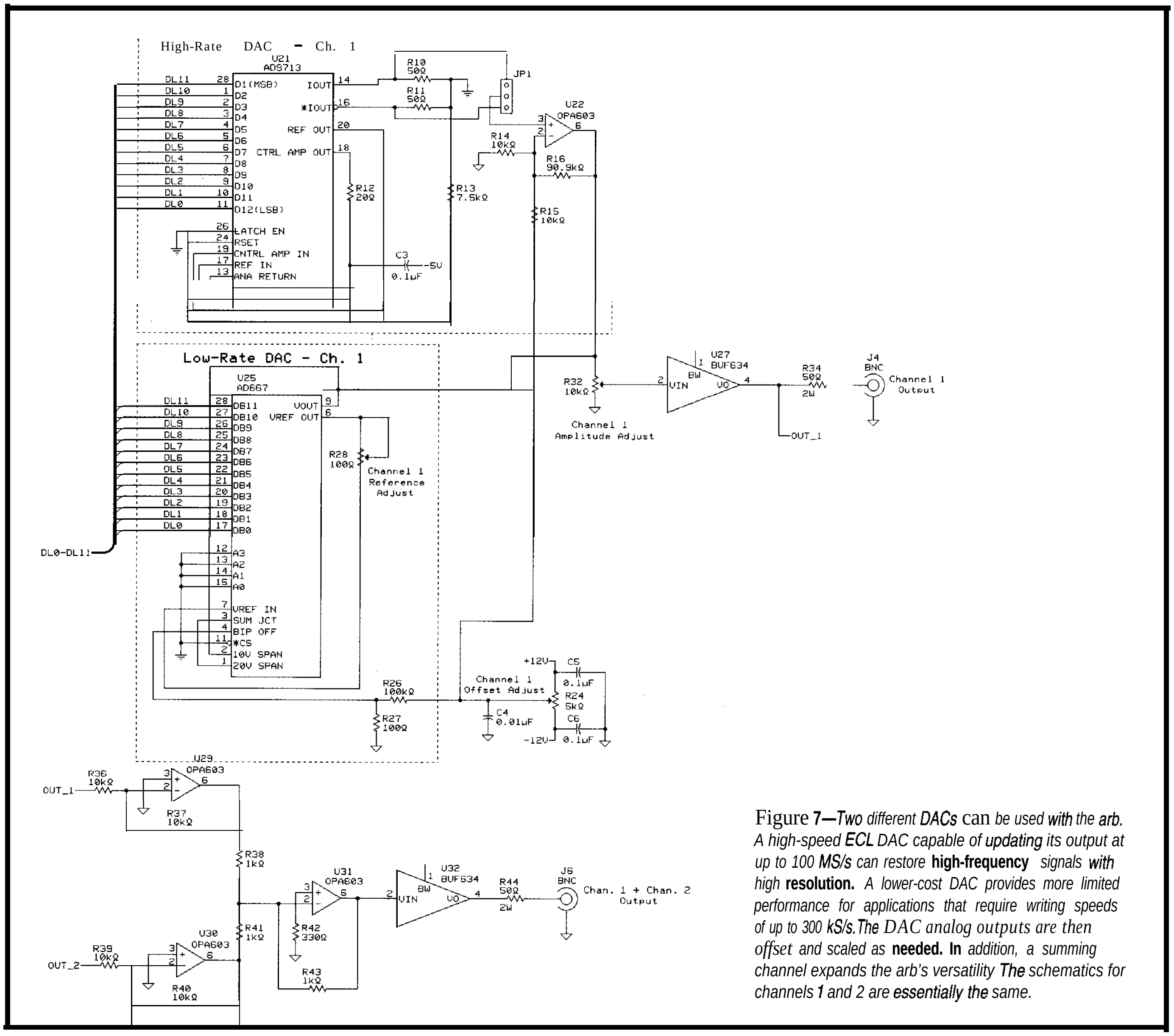


Figure 7—*Two different DACs can be used with the arb. A high-speed ECL DAC capable of updating its output at up to 100 MS/s can restore* **high-frequency** *signals with high* **resolution.** *A lower-cost DAC provides more limited performance for applications that require writing speeds of up to 300 kS/s. The DAC analog outputs are then offset and scaled as* **needed. In** *addition, a summing channel expands the arb's versatility The schematics for channels 1 and 2 are essentially the same.*

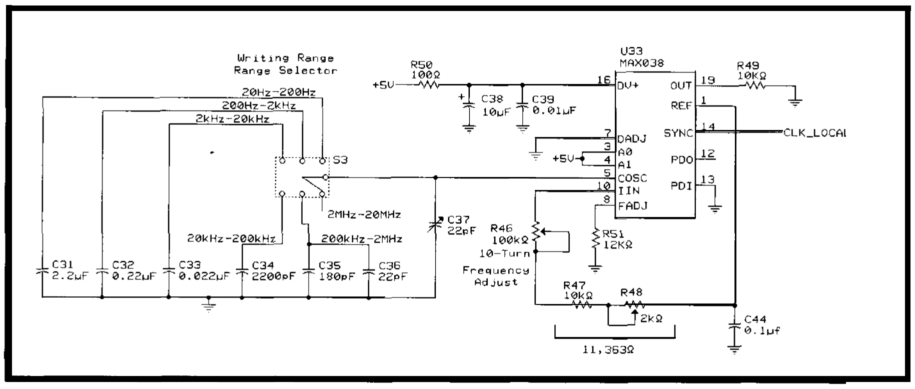


Figure 8—*The* **local** *sampling* **clock** *is generafed by* **U33,** *Maxim's MAX038 high-frequency waveform generator IC. Although this IC is typically used as a function generator, within fhe arb, if functions as a clock oscillator whose frequency can be controlled over the range* **of20** *Hz to 20 MHz.*

IC. The uPD43256B RAMs are then mounted on pins 3-30 of the Smart-Sockets.

You could also use Dallas Semiconductor DS1210 ICs to handle RAM power backup from a small battery [see Jeff Bachiochi's "Creating a Nonvolatile RAM Module," *INK* 16).

Last, a word of caution. High-frequency clocks and signals demand proper layout techniques (see "Designing Printed Circuits for High-Speed Logic," *INK 42).*

Preferably, use a multilayer PCB. Separate the analog ground from the digital ground, and join them at a single point at the power source.

Be sure to keep interconnection over the buses short and equal. In addition, use good-quality high-frequency capacitors to decouple the power rails close to each IC's power input pins.

ARBITRARY WAVEFORMS

Signal creation for reproduction by an arb is usually done by capturing an analog signal using a digital storage oscilloscope (DSO) or creating the waveform on a PC via a numerical representation of the waveform's mathematical formulation.

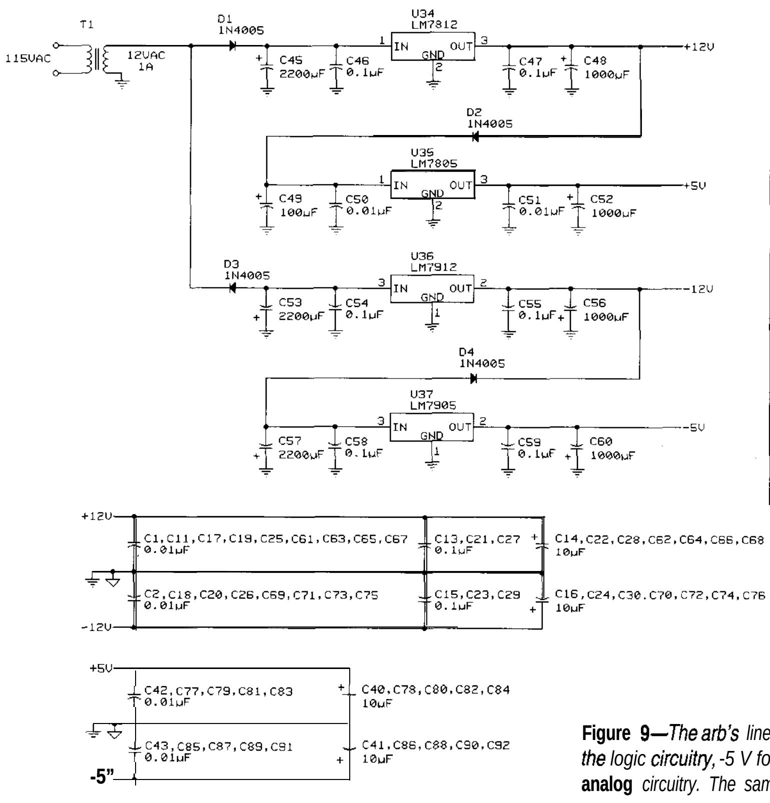
In the latter case, although short BASIC programs or numerical processing packages (e.g., Matlab) can generate waveform data, truly flexible waveform creation is possible only through dedicated software.

One of my favorite packages-pragmatic Instruments' WaveWorks Pro—offers an intuitive environment for waveform creation from a comprehensive menu of standard templates, math operations, and transfer functions.

Waveforms can also be imported from other programs or directly uploaded through GPIB or RS-232 from popular DSOs. Waveform synthesis and analysis can be performed either in the time or frequency domains.

Photo 1 shows how easy it is to create waveforms with a package such as WaveWorks Pro. The software has 30 standard waveshapes with programmable parameters.

They provide immediate solutions for generating test waveforms for general-purpose applications (e.g., sinusoidal, square, triangular waves, etc.), communications testing (e.g., AM, FM,



Power Table

Part	+5V	-5V	+12V	-12V	Gnd
U1	16				8
U2	16				8
U3	16				8
U4	16				8
U5	28				14
U6	28				14
U7	28				14
U8	28				14
U9	20				10
U10	20				10
U11	20				10
U12	20				10
U13	14				7
U14	14				7
U15	14				7
U16	20				10
U17	20				10
U18	20				10
U19	20				10
U20	16				8
U21	23	12,15,21,23,25			22,27
U22			7	4	
U23	23	12,15,21,23,25			22,27
U24			7	4	
U25			8	10	5,16
U26			8	10	5,16
U27			5	3	
U28			5	3	
U29			7	4	
U30			7	4	
U31			7	4	
U32			5	3	
U33	17				2,6,9,11,15,18

Figure 9—*The arb's linear power supply requires a* **12-VAC input to** *produce +5 V for the logic circuitry, -5 V for* **the ECL logic of** *the high-speed DACs, and ±12 V for the* **analog** *circuitry. The same circuit can power* **the DDS** *generator.*

BFSK, QPSK, NTSC waveforms, etc.), as well as other signals for advanced signal processing and control (e.g., sin(x)/x, ECG waveform, digital and analog noise, etc.).

After a waveform is defined, it can be modified using the 20 predefined transfer functions or I3 mathematical operators. Once the desired waveform is created, an FFT-based spectral estimator offers frequency analysis with the possibility of spectral editing and IFFT-based transformation back into time-domain.

A long, complex waveform can be created by looping and seamlessly linking previously created waveforms.

MORE FOR YOUR MONEY

As faster high-resolution DACs, wider RAMs, and higher performance processors enter the market, digital waveform generators are rapidly replacing analog sources.

High-performance integrated DDS generators have taken over the spread-spectrum communications field. They enable low-cost cable modems bringing you super-high-speed access to the 'Net from home.

Arbs are also becoming popular with design and test engineers, and they're more versatile sources than their analog counterparts. In fact, even with standard waveforms, arbs can compete with analog generators.

Of course, the neat control and waveform-design screens of commercial arbs, their powerful DSPs, and exotic high-frequency mixed-mode circuitry make them costly pieces of equipment. Most range from $3000 to $7000, whereas an analog signal generator with similar bandwidth costs just a few hundred dollars.

So, don't feel your reliable analog waveform generator doesn't deserve space on the workbench. Just keep the arb in mind when you demand ultimate flexibility and lots of performance. ❑

David Prutchi has a Ph.D. in Biomedical Engineering from Tel-Aviv University. He is an engineering specialist at Intermedics, and his main R&D interest is biomedical signal processing in implantable devices. You may reach him at davidp@mails.imed.com.

SOFTWARE

Software compatible with the DDS generator in this article is available at the ARRL ftp site at <oak.oakland.edu> in `DIGIVFO.ZIP`. It's also available on the Circuit Cellar Web site.

REFERENCES

[1] R. J. Portugal, "Programmable Sinewave Generator," *Electronics Now,* January 1995, 43-66.

[2] J. Craswell, "Weekend DigiVFO," *QST,* May 1995, 30–32.

SOURCES

HSP45102
Harris Semiconductor
1301 Woody Burke Rd.
Melbourne, FL 32902
(407) 724-3000
Fax: (407) 724-3937

AD9713
Analog Devices, Inc.
One Technology Way
Norwood, MA 02062
(617) 329-4700
Fax: (617) 326-8703

SmartSocket DS1213D, DS1210
Dallas Semiconductor Corp.
4401 Beltwood Pkwy. S
Dallas, TX 75244-3292
(214) 450-0448
Fax: (214) 450-0470

MAX038
Maxim Integrated Products, Inc.
120 San Gabriel Dr.
Sunnyvale, CA 94086
(408) 737-7600
Fax: (408) 737-7194

WaveWorks Pro
Pragmatic Instruments, Inc.
73 13 Carroll Rd.
San Diego, CA 92 121
(619) 271-6770
Fax: (619) 271-9567

I R S

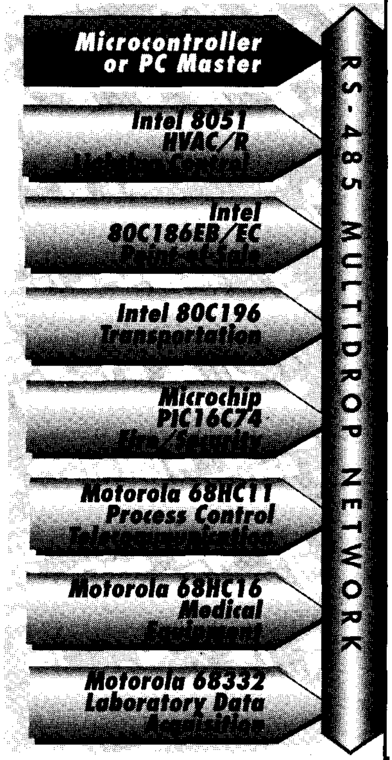
401 Very Useful
402 Moderately Useful
403 Not Useful

Finally, Standard RS-485 Network Software

With Cimetrics' 9-Bit µLAN 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 9-Bit µLAN 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 µLAN 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
Motorola 68332 Laboratory Data
RS-485 MULTIDROP NETWORK

Cimetrics TECHNOLOGY

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

#109

FEATURE ARTICLE

William Hohl & Joe Circello

Frequency Domain Analysis with ColdFire

Want a DSP and a microcontroller in the same chip? Listen in to hear how Cold-Fire's processor core, its multiply-accumulate engine, and signal transforms work together to provide cost-effective DSP functionality.

Way back during the Carter administration, a modem looked like a big, clunky shoebox with two large cups for holding your telephone handset. It could have been a prop in a Lost *in Space* episode. Of course, that was back when punch cards were the cutting edge in storage media.

Today, a modem is almost nothing more than a piece of software running on a dedicated DSP or the newest processor with multimedia extensions.

In embedded applications, DSPs are quickly replacing their analog counterparts as consumer electronics integrate voice/data and graphics capabilities in everything from telephones to automotive displays.

Now that technology is able to actually implement some of those gory algorithms you ignored in college, more applications are starting to use them—orthogonal transforms (remember Fourier?) and IIR filters, for example.

In embedded environments, however, there's a tradeoff between the amount of functionality you can assign the controller and the amount of board space you have for dedicated processors. Multiple-chip solutions are expensive.

A processor providing both the control functions and the necessary signal processing would be a great benefit to such designs.

Enter Motorola's ColdFire. Its architectural design specifically targets the emerging applications in advanced consumer electronics.

Its core is small enough to easily add on-chip memory, peripherals, and other system modules while remaining cost effective [1]. As you know, in cost-driven embedded systems, memory can sometimes end up costing more than the processor.

Since the ColdFire ISA is based on the 68k, it retains a high-density, variable-length instruction set that maximizes code density and keeps memory requirements down. Its architecture and implementation philosophy are flexible enough that different configurations within the core are also possible.

As for signal processing, the addition of a new multiply-accumulate (MAC) engine within the core supports a limited set of DSP operations that creep up in today's embedded applications. It also supports the existing multiply instructions already in the architecture-just more quickly.

In this article, we examine ColdFire's processor core, the MAC unit, and transforms used in signal processing, and we show how it all fits together.

PROCESSOR CORE

Let's start with a look at the V.2 processor core. It features two independent, decoupled pipeline structures that maximize performance while minimizing core size [see Figure 1).

The Instruction Fetch Pipeline (IFP) is a two-stage pipeline for prefetching instructions. The instruction stream is then gated into the two-stage Operand Execution Pipeline (OEP). This decodes the instruction, fetches the required operands, and executes the function.

The OEP includes the two standard execution units-a barrel shifter and the main ALU. The new MAC unit resides in the OEP and resembles another execution unit to the core. Each unit is a three-ported device that takes two operands as input and generates a result.

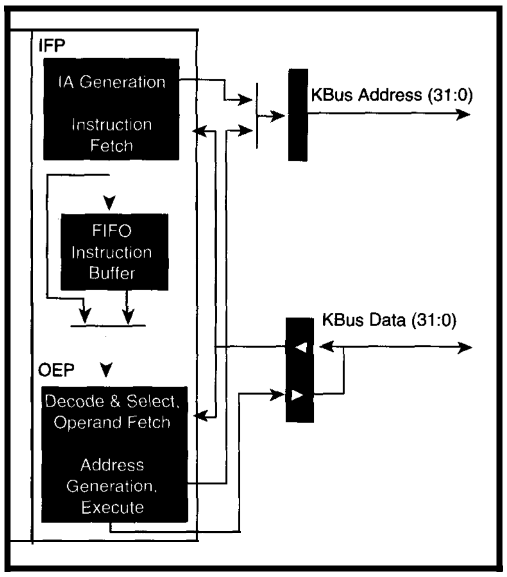


Figure 1—*The ColdFire* V.2 core consists of two *independent and decoupled pipeline stages. Once in the Execute stage, operands are sourced to one of three execute engines—the main ALU, a barrel shifter, or the MAC unit.*

THE MAC ENGINE

Before designing a new execution block, Motorola decided that redesigning the wheel wasn't a hot idea. The goal wasn't to create another DSP from the 68k architecture. However, they did want engineers to be able to implement a variety of DSP routines for practical applications.

To strike a middle ground between speed, size, and functionality, the new MAC unit implements a three-stage arithmetic pipeline containing a multiplier array followed by adder logic.

Since multiplier arrays can chew up silicon in a hurry, the MAC unit is optimized for 16 x 16 multiplies with a possible accumulation cycle to follow. At the expense of a little extra control logic, 32-bit operations are still supported.

The new MAC instructions provide for the multiplication of two numbers, followed by the addition or subtraction of this product to or from the accumulator's value (see Table 1).

Some of the truly useful additions to the ColdFire architecture come from new instructions that enable an operand fetch in parallel with a MAC operation. This results in an overall performance increase for operations like convolution and filtering.

The product can also be shifted 1 bit to the left or right before addition or subtraction takes place. For situations where you might use saturation arithmetic (e.g., in a filter with input values that may not be within the range you expected), a bit in the MAC unit's status register enables or disables saturation on an overflow.

The MAC engine is pipelined, so you can issue MAC instructions once every clock for word-length operations and once every three clocks for long operations. Since only the MAC unit sees the value in the accumulator, an additional *move* instruction is necessary to transfer data to and from a general-purpose register.

You can also choose which word you want in a long word during calculations. This feature is extremely useful for DSP operations, since you can load two 16-bit coefficients into one register and two 16-bit data samples into another.

Alternating the word choice means you can perform two 16-bit MAC operations without fetching additional operands between instructions.

TRANSFORMS

Embedded processors are getting smaller, faster, and smarter. You can now implement a number of computationally intensive algorithms that were relatively uncommon in embedded code.

While we're not talking about routines that make your toaster talk, there are some fairly common algorithms (e.g., orthogonal transforms) that convert time-domain signals into the frequency domain.

In this article, we discuss two transform implementations-one for the Discrete Fourier Transform (DFT) and one for the Discrete Cosine Transform (DCT).

DISCRETE FOURIER TRANSFORM

Let's look first at the DFT. Think back to that partial differential equations course you took in college. If you recall, for a continuous-time signal x(t), the Fourier transform is defined as:

$$X(f) = \int_{-\infty}^{\infty} x(t)\, e^{-j2\pi ft}\, dt$$

and in general, x(t) is a complex function. A discrete-time signal x(n) is created by sampling the continuous waveform x(t).

If we restrict the length of the sequence *x(n)* to n samples and we assume that the signal is periodic outside that range, then the Fourier transform becomes discrete with the distance between samples being $(2\pi/N)$ in normalized frequency units.

So for a discrete sequence *x(n),* the forward transform winds up as:

$$\mathbf{X}(k) = \sum_{n=0}^{N-1} x(n)\, e^{-i\frac{2\pi kn}{N}}$$

For years, a number of new and interesting approaches have been taken

Operation	Mnemonic	Description
Mult Signed	MULS <ea>y,Dx	Multiplies two signed operands, signed result
Mult Unsigned	MULU <ea>y,Dx	Multiplies two unsigned operands, unsigned result
Mult Acc	MAC Ry,RxSF MSAC Ry,RxSF	Multiplies two unsigned/signed operands, then adds/ subtracts product to/from acc
Multiply Acc with Load	MAC Ry,RxSF,<ea>,Rw MSAC Ry,RxSF,<ea>,Rw	Multiplies two unsigned/signed operands, then adds/ subtracts product to/from acc while loading a reg with memory operand
Load Acc	MOV.L {Ry,#imm},Racc	Loads acc with 32-bit operand
Store Acc	MOV.L Racc,Rx	Writes contents of acc to a reg
Load MAC Status Reg	MOV.L {Ry,#imm},MACSR	Writes a value to MAC status reg
Store MAC Status Reg	MOV.L MACSR,Rx	Writes contents of MAC status reg to a reg
Move MACSR to CCR	MOV.L MACSR,CCR	Writes contents of MAC status reg to processor's CCR
Load Mask Reg	MOV.L {Ry,#imm},Rmask	Loads mask reg with lower 16-bits of operand
Store Mask Reg	MOV.L Rmask,Rx	Writes mask reg to a reg

Table I--The *MAC unit supports the existing ColdFire multiply instructions and provides the two new* MAC *commands. A new feature in the architecture can* load data **in parallel** *with the MAC instruction.*

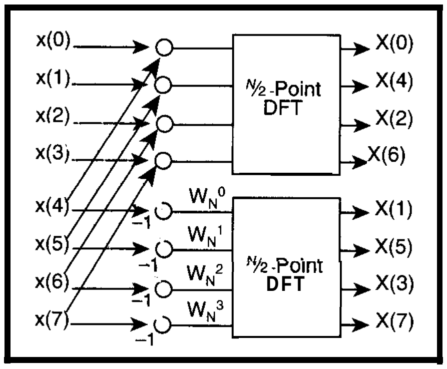


Figure 2—*This flow graph for an eight-point decimation-in-frequency FFT shows the algorithm's iterative nature. Notice that the input data is sequential, whereas the output is in bit-reversed order.*

to reduce the computational requirements to implement this equation. Most try to exploit the properties of the phase (or twiddle) factors:

$$e^{-j\frac{2\pi kn}{N}}$$

Such methods include the Goertzel algorithm [2], the chirp z-transform method, and other flavors involving convolution.

The more popular Fast Fourier Transform-the Cooley-Tukey algorithm-has a fairly lengthy history, so we'll only examine it from the point of the signal-flow graph.

In Figure 2, a decimation-in-frequency algorithm is shown for an eight-point transform. You can see that the output X(k) is successively divided into smaller and smaller subsequences. For this case, there are three stages of calculation, and each stage computes four two-point DFTs or butterflies (see Figure 3).

How do you implement this? First, consider the amount of memory you have, the processor speed, and the application's limitations. Obviously, if you have a time-critical application, you may have to use extra memory and straight-line code the algorithm.

The source code presented here for the FFT routine implements the famous triple-nested DO loop for a complex 128-point FFT [3]. It has a small code size but not the fastest execution time.

Second, you have to decide on a data format. Floating-point formats are messy, and having your binary point wander all over the place makes the bookkeeping tedious. Plus, you have to worry about overflow conditions.

So for this example, the data is assumed to be in a fixed-point fractional notation, where the most significant bit represents the sign of the number and the remaining bits represent digits behind a binary point.

In other words, you find the decimal value by treating the 16-bit field as a 2's complement number and dividing it by 2^{30}. This implies, however, that the number 1 .O can't be represented by this notation.

But, such is life. We can work around this.

When working with fixed-point numbers, overflow is a definite possibility. That is, you cannot represent the product of your two numbers using only 32 bits. To deal with these cases, use interstage scaling so that at each stage of the FFT calculation, the output is divided in half.

Specifically, for *M* stages, you end up with the final results X(k) scaled down by 2^M. Why is this necessary?

If you examine the results at each stage of the calculation, the newest values of *A* and *B* are found by:

$$\begin{aligned} A(n+1) &= (AR + jAI) + (BR + jBI) \\ &= (AR + BR) + j(AI + BI) \\ B(n+1) &= [(AR + jAI) - (BR + jBI)] \\ &\quad [\cos\theta - j\sin\theta] \\ &= [\cos\theta(AR - BR) + \sin\theta(AI - BI)] \\ &\quad + j[\cos\theta(AI - BI) - \\ &\quad \sin\theta(AR - BR)] \end{aligned}$$

For the value of $A(n+1)$, the sum of the imaginary or real parts can produce a value greater than 1. So, the values *AR, BR, AZ,* and *BZ* are all scaled down by a factor of two before they get used. This way, the sums are guaranteed to be realizable in this fractional notation.

For the output $B(n+1)$, the largest value that either the real or the imaginary values could have is:

$$\cos(45°)(1) + \sin(45°)(1) < 2$$

So, if the difference terms, such as *(AR – BR),* are already scaled down by a factor of two, the largest value'for the sum is guaranteed to be less than 1.

Listing 1 offers the assembly code calculating the real and imaginary values for $B(n+1)$. In this example, the 16-bit values are stored sequentially.

For example, the complex value *AR + jAI* is contained in one long-word operand. Each coefficient (twiddle factor) is also stored in memory with real and imaginary halves.

The difference *(AR – BR)* and the sum *(AR + BR)* have already been stored in registers d5 and d6, respectively, and scaled down by a factor of two.

For the real portion of $B(n+1)$, the cosine part of the twiddle factor is chosen with the upper/lower word select, then multiplied by the difference and shifted one bit to the left with one MAC instruction.

Listing 1—*This section of the FFT routine computes the* *new value of B(n+1). The real and imaginary portions of each operand are stored sequentially and the operands are considered fractional values.*

```
# clear MAC's   accumulator
sub.l     %a6,%a6
mov.l     %a6,%acc
#start bottom of butterfly
#cos+jsinx gets loaded first
mov.l     (0,%a1,%d2.l*4),%d6

#AR and BR have already been prescaled
#calculate  sinx(AI - BI)/2 + cosx(AR - BR)/2
mac.w     %d6:u,%d5:u<<1
mac.w     %d6:l,%d3:u<<1
mov.l     %acc,%d4
swap      %d4

#store BR to memory
lea       (0,%a0,%d1.l*4),%a6
mov.w     %d4,(%a6)

#calculate  cosx(AI - BI)/2 - sinx(AR BR)/2
mov.l     %a4,%acc
mac.w     %d6:u,%d3:u<<1
msac.w    %d6:l,%d5:u<<1
```

That left shift realigns the binary point and removes the extra sign bit. You get the sine portion in a similar way. This value is then added to the value already in the accumulator.

A final store of the accumulator to a general-purpose register lets us move the results out to main memory. The imaginary part of $B(n+1)$ is found in much the same way as the real part.

As we mentioned, the code for the entire FFT routine is written as a set of nested loops. The outer loop controls which stage of the FFT you're in.

A second, inner loop determines the step size needed for pointing at the right set of values to use.

The innermost loop does most of the work. Since the two-point butterfly sits inside this loop, you obviously want to minimize the number of cycles spent calculating it.

You can use a few tricks to reduce both the cycle time and the instruction count inside of tight loops.

For example, normally you might be tempted to just move zero into an address register to clear it. However, in the ColdFire ISA, this instruction would occupy two words and could force two instruction fetches if it sits on a funny boundary.

A better approach is to subtract the register from itself. It's a one-word opcode, and it does the same thing in one cycle.

Another operation to watch for is a multiplication involving two 16-bit signed numbers, such as the ones we're using. When you multiply two 16-bit numbers to produce a 32-bit result, you end up with an extra sign bit. A left shift is needed to keep the binary point in the correct spot.

Here's where the optional shift on a MAC instruction is useful. You can realign the data before adding it to the accumulator, saving another cycle.

DISCRETE COSINE TRANSFORM

In the next transform example, we look at a two-dimensional Discrete Cosine Transform (DCT).

This transform has been used for data compression in a number of different standards, including CCITT Rec. H.261, the JPEG standard for still images, and the MPEG standards for video.

There are several efficient ways to implement this algorithm. Over the years, a number of fairly clever routines have emerged, such as those by Hou [4] and Lee [5].

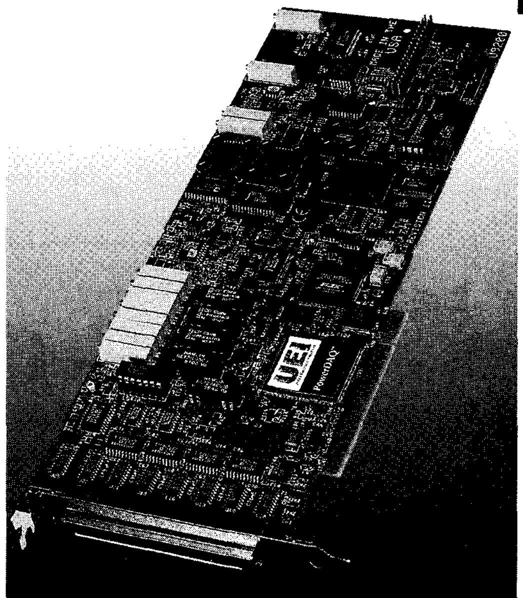
In fact, you can show that the algorithm for computing the DCT looks like the one for computing the FFT. However, for the purposes of illustration, we use a more direct method—the matrix formulation.

A two-dimensional DCT is given by:

$$\theta(k,l) = \frac{2}{N}\alpha(k)\alpha(l)\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} x(m,n) \times \cos\left(\frac{\pi k(2m+1)}{2N}\right)\cos\left(\frac{\pi l(2n+1)}{2N}\right)$$

.isting 2—*For the DCT, one vector of data is calculated by successive MAC operations. By using movm, data s quickly loaded info four general-purpose registers first, avoiding any later operand fetches that may add cycles.*

```
  movm.l   (%a0),&0x0078    # load d3-d6 with source data
loop_one:
  movm.l   (%a1),&0x7800    # load a3-a6 with coeff data
  mov.l    &0x8000,%acc     # rounding value
  mac.w    %d3:u,%a3:u
  mac.w    %d3:l,%a3:l
  mac.w    %d4:u,%a4:u
  mac.w    %d4:l,%a4:l
  mac.w    %d5:u,%a5:u
  mac.w    %d5:l,%a5:l
  mac.w    %d6:u,%a6:u
  mac.w    %d6:l,%a6:l
  mov.l    %acc,%d7
  swap     %d7
  mov.w    %d7,(0,%a2,%d0) # output rearranged for next pass
  add.l    coffset,%d0      # on first pass, add 16, else add 0
  lea      (16,%a1),%a1     # always add 16
  subq.l   &1,%d1
  bne.b    loop-one
```



[illegible] plete implementation of the PCI specification, including such features as:

- 1.25 MHz A/D continuous to system memory
- DSP-enhanced interface with simultaneous data I/O
- Bus-mastering with scatter/gather DMA
- 32-bit drivers for Windows NT and Windows 95.

{Second in a continuing series. iook for Reason #3: "16-channel on-board simultaneous sampling.')

Call today for your free catalog:

(800) 829-4632

UEI

United Electronic Industries

10 Dexter Ave, Watertown, MA 02172

Tel: (617) 924-1155
Fax: (617) 924-1441

internet: www.ueidaq.com
E-mail: sales@ueidaq.com

#111

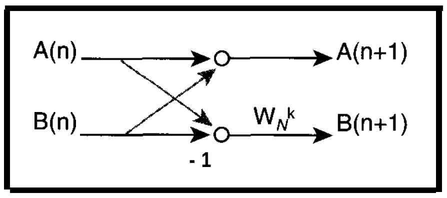


Figure 3—*The* **two-point** *butterfly* **is** *the* **simplest** *FFT and the basic building block of larger transforms.*

where $x(m,n)$ is an N x N field and

$$\alpha(k) = \frac{1}{\sqrt{2}}$$

fork = 0 , unity otherwise.

The input in this case might be something like an 8 x 8 block of pixel data from an image. If the DCT is carried out with a matrix formulation, then the routine is based on:

$$\begin{bmatrix}\theta_0\\ \theta_1\\ \theta_2\\ \theta_3\\ \theta_4\\ \theta_5\\ \theta_6\\ \theta_7\end{bmatrix} = \left|\begin{array}{cccccccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\\ \lambda & \gamma & \mu & v & -v & -p & -y & -h\\ \beta & \delta & -s & -p & -p & -s & \delta & \beta\\ y & -v & -h & -p & \mu & \lambda & v & -\gamma\\ \alpha & -\alpha & -\alpha & \alpha & a & -a & -a & a\\ \mu & -\lambda & \nu & y & -y & -v & \lambda & -\mu\\ s & -p & \beta & -\delta & -\delta & \beta & -\beta & \delta\\ \nu & -\mu & \gamma & -\lambda & \lambda & -\gamma & \mu & -\nu \end{array}\right| \begin{bmatrix}x_0\\ x_1\\ x_2\\ x_3\\ x_4\\ x_5\\ x_6\\ x_7\end{bmatrix}$$

where: $\beta = \sin\left(\frac{\pi}{8}\right)$, $\delta = \cos\left(\frac{\pi}{8}\right)$,

$\lambda = \cos\left(\frac{\pi}{16}\right)$, $\gamma = \cos\left(\frac{3\pi}{16}\right)$,

$\mu = \sin\left(\frac{3\pi}{16}\right)$, and $\nu = \sin\left(\frac{\pi}{16}\right)$.

To calculate the entire transform, you have to perform the above operation 64 times. Since the DCT kernel is separable, the two-dimensional transform can be done in two passes-first along the rows of the input matrix, then along the columns.

In this implementation, a separate matrix to hold the transposed data isn't needed before the second pass. The operands are stored in memory in their transposed positions during the first pass.

Eight small loops comprise the bulk of the DCT routine (one is shown in Listing 2). Each loop calculates one vector of output data. In this implementation, all the coefficients and data samples are word-length operands.

Instead of fetching two operands to multiply and accumulating the result, a movm instruction loads four general-purpose registers with all eight data samples. Another movm loads four more registers with all eight coefficients.

The MAC instructions are done in series, effectively one per clock. Using the upper/lower word select bit, you can perform two MAC operations with the same registers. The accumulator is then transferred to a general-purpose register and moved out to memory.

The format of the coefficients is similar to that of the FFT routine. In other words, the binary point is assumed to be after the first bit and the remaining bits are the fractional value.

The input data is an integer value and, depending on the image, can be between 0 and 255. During the intermediate multiplications, only the integer portion of the results are kept [the upper 16 bits).

Using a technique similar to the one by Srinivasan et al [6], a rounding value is added at the beginning of the routine to account for round-off errors and truncation.

FINDING YOUR SOLUTION

These algorithms are intended as illustrations, not actual concrete solutions. While they provide a working model, certain enhancements speed up execution times (e.g., using caches or storing some data blocks in on-chip RAM instead of external memory).

Depending on the DSP problems you want to solve, you might find yourself looking for a fast predictive filter, another type of transform, or just a run-of-the-mill FIR filter. We hope these examples give you some ideas for starting the code.

For situations where you have some kind of DSP functionality built into your design but don't want to spend the money on a full-blown signal processor, the MAC unit on the ColdFire processors is a nice alternative.

The optimized performance coupled with an approximate gate count of 8500 makes the module a cost-effective solution for embedded applications that require fast signal processing. ❑

William Hohl is a systems architect with Motorola's Imaging and Storage Division, He designed the debug unit for the ColdFire product family and, most recently, developed the MAC architecture. You may reach William at (512) 314-1054.

Joe Circello is a microprocessor architect for Motorola's Imaging and Storage Division. He specializes in pipeline organization and performance analysis. He was also the pipeline architect for the 68060 and developed the ColdFire architecture. You may reach Joe at circello@sedona.sps.mot.com.

SOFTWARE

The complete source code listings for the DFT and DCT are on Motorola's Web site at <www.mot.com/coldfire>.

REFERENCES

[1] J. Circello, "ColdFire: A Hot Processor Architecture," ***BYTE, 20,*** 173–174, 1995.

[2] ***G.*** Goertzel, "An Algorithm for the Evaluation of Finite Trigonometric Series," ***Amer. Math. Monthly, 65,*** 34–35, ***1958.***

[3] A.W. Oppenheim & R.W. Schafer, ***Digital Signal Processing,*** Prentice-Hall, Englewood Cliffs, NJ, 1975.

[4] H.S. Hou, "A Fast Recursive Algorithm for Computing the Discrete Cosine Transform," ***IEEE Transactions on ASSP, ASSP-35, 1455-1461, 1987.***

[5] B.G. Lee, "FCT-Fast Cosine Transform," ***Proceedings of 1984*** Conference on ASSP, 28.A.3.1–28.A.3.4, 1984.

[6] S. Srinivasan et al, "Cosine Transform Block Codec for Images Using the TMS32010," ***Proceedings of IEEE ISCAS, 299302, 1986.***

SOURCES

ColdFire
Motorola
Imaging and Storage Div.
6501 William Cannon Dr. W
Austin, TX 78735-8598
(512) 891-2000
Fax: (512) 891-8315
www.mot.com/coldfire

I R S

404 Very Useful
405 Moderately Useful
406 Not Useful

DSP-Based Canadian Timecode Receiver

FEATURE ARTICLE

David Tweed

Part 2: Application Considerations

Having conquered the theory of DSP transforms last month, David moves on to a practical application—getting his local timebase to receive and adjust its clock to match that of CHU, a Canadian radio station.

In Part 1, I looked at various ways to detect audio tones and their relevance to building a software decoder for the signal from radio station CHU in Ottawa, Canada. I also covered cross-correlation and FIR filtering.

This month, I get back to the Fourier Transform and examine its use in real-time applications. I conclude by looking at how to build a local copy of the UTC timebase, plus the details of decoding the modem signal.

FOURIER TRANSFORM

In cross-correlation and FIR filtering, you basically multiply time-delayed copies of the input signal with a series of numbers representing a template function and/or FIR coefficients. You then sum the results to get a single number per trial.

In particular, the cross-correlation example "slides" a sine-wave template along the input signal to find the best match.

But if instead of sliding the template along, you simply multiply point-by-point the input signal with a sinewave on a continuous basis, you get an interesting result. Assuming the input signal is also a sinewave, the resulting function contains two new signals representing the sum and difference frequencies and not the original signals (see Figure 1).

In electronics, a circuit performing this function is called a "balanced mixer" or "product detector." Now, you know why the word "product" is used.

If the input and template frequencies are equal, the difference frequency is, of course, 0 (DC). The specific DC level depends on the phase relationship between the two signals.

Low-pass filtering the result of the multiplication effectively eliminates the sum frequency component, leaving just the difference component. If you make this filter with a very low cut-off frequency, the (nearly) DC output indicates whether the template frequency exists in the input signal.

This concept can be easily generalized. Suppose you want to see what frequency components exist in an arbitrary input signal. You can do the same analysis for many different frequencies.

To keep things tractable, I use sinewave template frequencies that are integer multiples of the lowest frequency fitting in the sample window. The layers of Figure 2 illustrate this process.

The input signal in the top section is the same for each frequency. The center section shows the template

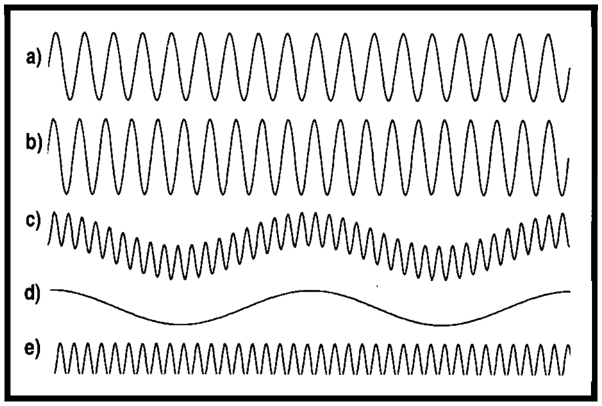


Figure Ia & b—Multiplying together the 180- and 200-Hz waveforms gives the waveform in trace (c). d & e—You get this same waveform by adding together the 20- and 380-Hz waveforms.

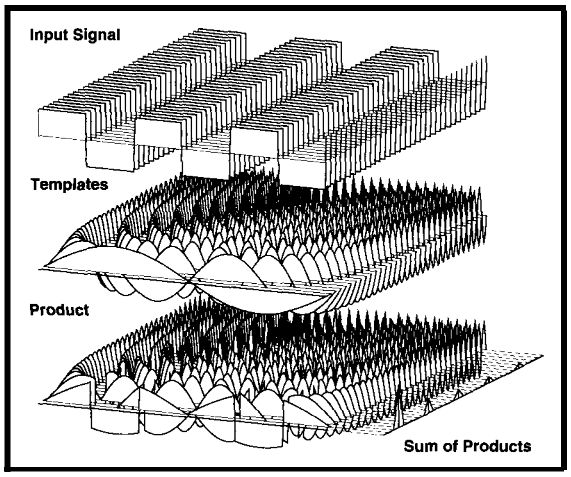


Figure 2—*Analyzing the input signal* for various frequencies clearly shows the characteristic harmonic *structure of a* square *wave.*

frequencies, with the lowest one in front. The bottom section is the result of point-by-point multiplication of these two sections.

Integrating a continuous signal, or adding up the samples of discrete signal samples, is one form of low-pass filter-although not a real good one for this application. So, the graph to the right of the bottom section in Figure 2 shows the result of summing the results from each trial.

The input signal is a square wave. The graph clearly shows the decreasing series of odd harmonics expected in the spectrum of a square wave.

Figure 2's input signal has frequency components exactly matching the frequencies used for analysis. However, consider analyzing a single sinewave with a frequency that doesn't match any template exactly.

Although there were plenty of nonzero results after the multiplication stage, most of them canceled out in the integration stage.

In Figure 3, none of the trials have results that cancel exactly. So, all the frequency bins show nonzero values after integration.

Note that the difference frequency present in each trial is not exactly DC in any trial. Using straight integration as a low-pass filter enables these difference frequencies to show up in each output bin because its frequency response drops off relatively slowly.

The solution: design a better low-pass filter for after the multiplication stage, using the FIR technique I mentioned before.

The bottom sections of Figure 3 show the results. The fourth section shows the FIR coefficients for a relatively steep low-pass filter. The last one shows the result of multiplying this point-by-point with the results in the third section and the curve generated by summing them.

Now, let's optimize it. If you look at Figure 3 and consider a single point in all five layers, you see I'm doing two multiplications in the sequence A x B x C. A is the input signal, B is the frequency analysis template, and C is the FIR coefficient.

If you consider all the points in a layer running front to back in the diagram, you can see that while B is different for each layer, A and C are the same.

Why not multiply A x C once, and multiply that result by the different B values? As Figure 4 shows, the result is the same.

We don't have a Fourier Transform quite yet. Remember the sine/cosine analysis with the cross-correlation? And, recall that the DC value depends on the phase relationship?

I deal with these issues by running a second analysis using cosine waves of the various analysis frequencies. Then, I combine the two sets of results into an overall magnitude value by taking the square root of the sum of the squares of the individual results.

I can also get a phase angle by taking the arctangent of the ratio of the two results frequency by frequency. Now, that's the complete Discrete Fourier Transform (DFT).

To summarize, the DFT analyzes a signal represented by n points in time. The answer is in the form of $n/2$ numbers representing discrete frequency bins.

This result occurs because we can't use an analysis sinewave (or cosine wave) with a frequency greater than half the sample rate. Again, Nyquist rules.

Also, if the signal being analyzed has frequency components that don't exactly match the bin frequencies, the answer is spread across several bins. You can minimize this with proper filtering.

While the DFT shows whether energy is present at a particular frequency during a sample window, there's no indication of when the energy started and/or stopped. I need small sample windows to get better time precision.

However, small windows, with a small number of samples per window, give a smaller number of frequency bins. They also give poorer resolution in the frequency domain and more opportunity for noise to get in and obscure the result.

If you're just looking for energy at a particular frequency (bin), why run all the calculations for all bins? Why not just calculate for the bin you're interested in, and use the other CPU time to try the calculation with different sample windows shifted in time. This idea brings us back to the cross-correlation we started with.

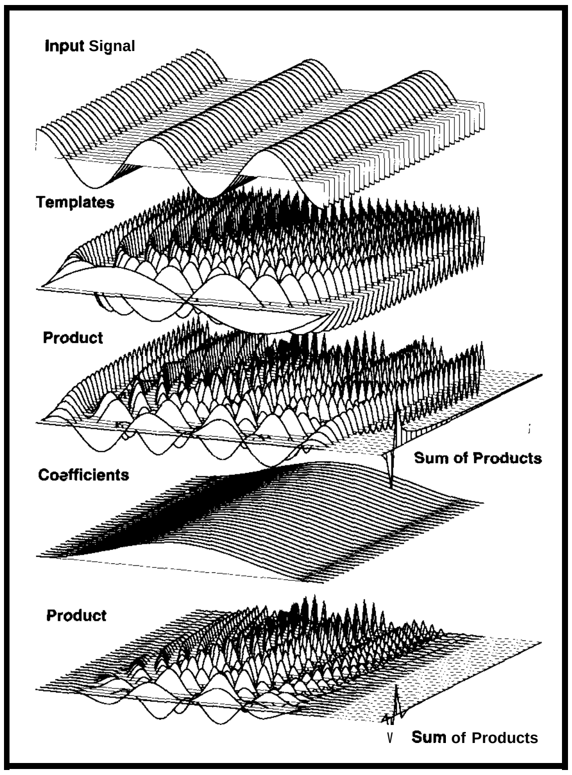


Figure 3-When fhe *input* signal doesn't *fall neatly* info one of *the* frequency bins, there *is leakage info all of the other bins. This example shows a worst-case* condition. Adding a better *low-pass* filter *improves* the *situation* tremendously.

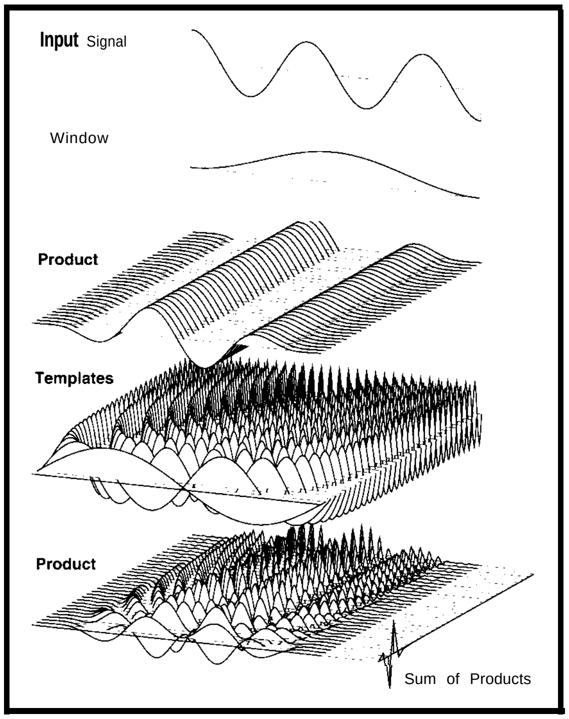


Figure 4—*Rearranging the order of operations eliminates much of the redundancy. Now, I can call the low-pass filter a "window function."*

By the way, if you want to detect energy at a particular frequency and don't care precisely when the energy starts and stops, there's an efficient form of the one-bin DFT known as Goertzel's algorithm. Analog Devices' ADSP-2101 app notes use this algorithm to build a DTMF decoder for telephone applications.

The standard one-bin DFT calculation requires you to store the n samples, multiply them by the values of the template sine and cosine waves, and sum the results. This so-called nonrecursive implementation is analogous to a finite-impulse-response (FIR) filter.

Goertzel's algorithm transforms this into the equivalent recursive form, analogous to an infinite-impulse-response (IIR) filter. The calculation then proceeds step-by-step as samples come in, without storing them individually. This change significantly decreases the amount of RAM required for each time the algorithm is used.

DIGITAL SYNTHESIS

Now that we've nailed down some techniques for tone detection, I need to deal with the fact that there are two timebases. Figure 5 shows the entire system from master oscillator to receiver output.

One timebase is driven by the master oscillator at the transmitter. It controls the starting, stopping, and frequency of the audio tones transmitted by the station.

It also provides the data for the modem signal that gives the coarser measurements of time (minutes, hours, and date). I want my receiver to become a replica of this master timebase.

However, the receiver already has a second timebase-the DSP's crystal-controlled clock. It is divided down to drive various aspects of the DSP's operation, including the sample clock of the ADC that reads the radio receiver's audio output.

How can I replicate the master timebase using the local crystal-controlled timebase?

Let's assume the crystal is moderately accurate, within 100 ppm (0.01%) of the frequency marked on its case. Let's also assume its exact frequency doesn't change significantly over time regardless of what it is.

A powerful and flexible approach of synthesizing any frequency from an arbitrary existing clock is called Direct Digital Synthesis (DDS).

DDS is nothing more than a binary register of some number of bits that gets a number added to it at a steady, repeated rate (see Figure 6a). After a few clocks, the register may overflow, but you just keep going.

Figure 6b shows the results for a simple 4-bit DDS circuit that's clocked at 16 Hz. The register numbers wrap around at a rate proportional to the number being added.

In fact, they give discrete samples of a sawtooth wave whose frequency in Hertz is the same as the number applied to the adder.

Watch what happens as I add more bits to the register, as in Figure 6c. With a 6-bit register, I can now specify S-bit numbers representing O-8 Hz as before. But, can I generate some fractional frequencies between the ones I could do with the 4-bit register?

This concept can be extended. If I take my 8000-Hz sample clock and use a 32-bit DDS, I get a frequency resolution of 1.86 µHz.

To give you a feel for what this kind of accuracy implies, I can set one such DDS to the value 536870,912, causing it to generate a waveform at exactly 1000.00000000 Hz. I set another to 536,870,913, making it generate 1000.00000186 Hz.

After one day, the two generators differ by a phase angle of just 58". It takes nearly a week before they're off by a full cycle, providing the kind of precision that enables me to account for the difference between the master and local crystal-controlled timebases.

If this seems a little confusing, take a step back and consider a different point of view.

As far as the DSP is concerned, it "thinks" that its local timebase is perfectly accurate and that I'm using the DDS to track a distant timebase at a "wrong" frequency.

That's fine for it, but you and I know that the remote timebase is perfectly accurate. The errors are in the local

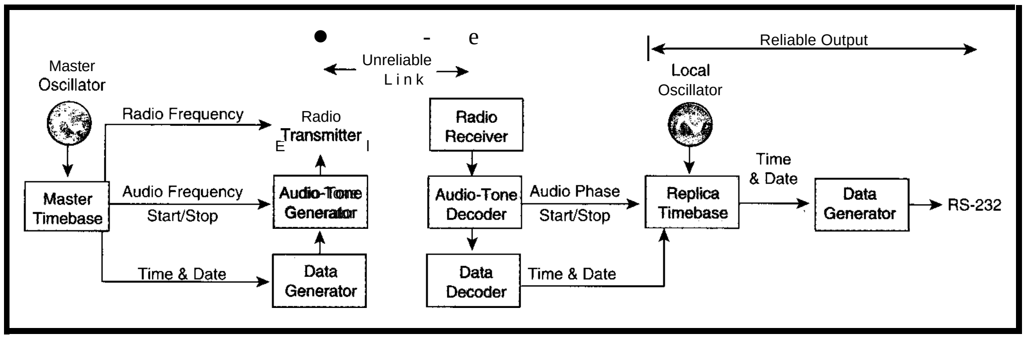


Figure 5—*This project should provide a continuous time and date signal even though the radio link from the master clock is frequently interrupted.*

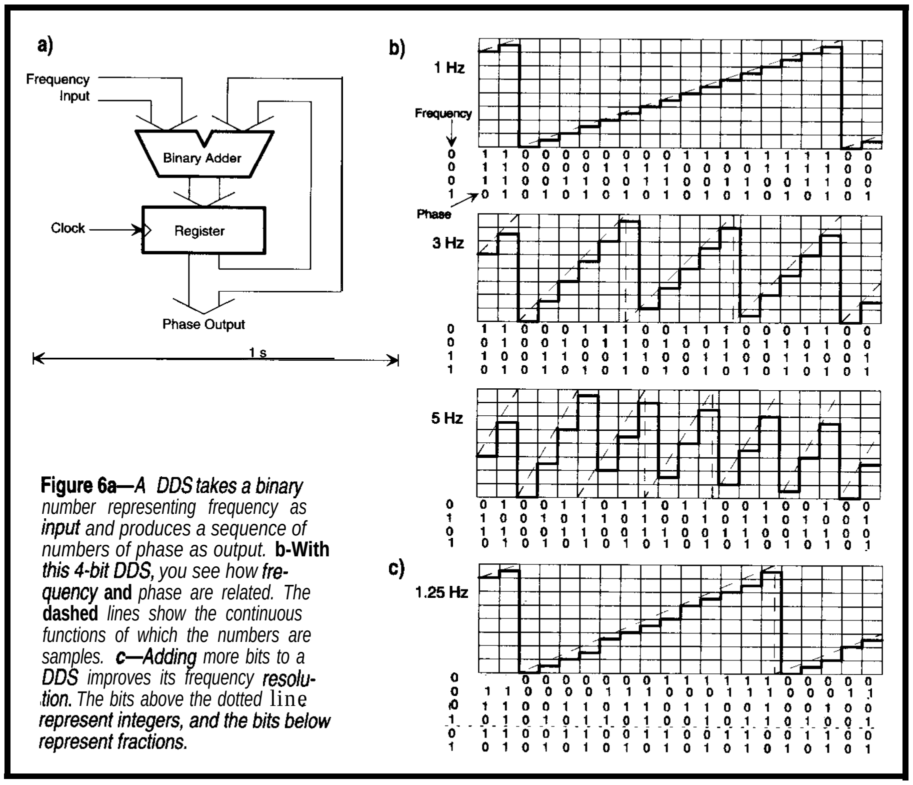


Figure 6a—*A DDS takes a binary number representing frequency as input and produces a sequence of numbers of phase as output.* **b-With** *this 4-bit DDS, you see how frequency and phase are related. The dashed lines show the continuous functions of which the numbers are samples.* **c—Adding** *more bits to a DDS improves its frequency resolution. The bits above the dotted line represent integers, and the bits below represent fractions.*

for each of those two frequencies, using DDS to generate the local template waveforms.

I use a sliding window for the DFTs. Since I'm doing only single bins, the load on the CPU is no greater than an FIR filter of similar size.

It also has the advantage of giving a result for every input sample, providing the needed resolution in the time domain.

I then compare the magnitudes of the outputs of the detectors, using their relative levels to pick out the bit edges and decide whether the individual bits are 1s or 0s.

Each tone might exist for only one bit time (at 300 bps at a time). To get the maximum possible output from the tone detectors, I use a window size that's as close as possible to the bit size (8000/300 = 26 samples).

When this window, which is used for both tone detectors, is lined up on a single bit worth of tone, one detector output is at a maximum while the other is at a minimum. Figure 7 shows how this looks in real time.

I want to reduce these detector outputs to two logical concepts. The first is carrier detect, or whether either of these tones is present at all, and the second is the actual data, or which of the two tones is present. If I was just building a stand-alone modem, these would become the CD (carrier detect) and RXD (receive data) signals in the RS-232 connector.

timebase. Still, it doesn't make any difference to the algorithm.

In any case, I use the replica timebase as the basis for the ASCII output of the decoder. Plus, I may later decide I want other kinds of outputs that can be derived from it.

FEEDBACK CLOSES THE LOOP

So, now I have a timebase I can adjust relative to the local crystal. But, how do I adjust the timebase so that it's a replica of the master timebase, especially given that the link from there to here is rather unreliable?

The answer: feedback. After getting the timebase going, I compare its output (or portions of its output) to the signal from the radio station. The results tell which way and how much to adjust the replica timebase so that it is synchronized.

In addition, I'll assess the quality of the radio signal so I know how much to trust the results of the comparison. If the signal is no good, I'll ignore the results and let the replica timebase free-run for a while.

It doesn't matter what the local crystal's accuracy is. Only its stability over time-or lack thereof-affects the free-running accuracy of the replica timebase.

There are two parts of the radio signal in particular that 1'11 use in the comparison between the master and replica timebases. First, I need to decode the 300-bps data to get coarse information like date, hour, minute and second.

Then, I'll look at the 1000-Hz tone bursts to line things up to the submillisecond level. This is where the tone-detection algorithms from Part 1 come into play.

SOFTWARE MODEM

The main task associated with decoding the modem signal is recognizing the 2025 and 2225Hz tones. My approach involves setting up single-bin DFT-style tone detectors

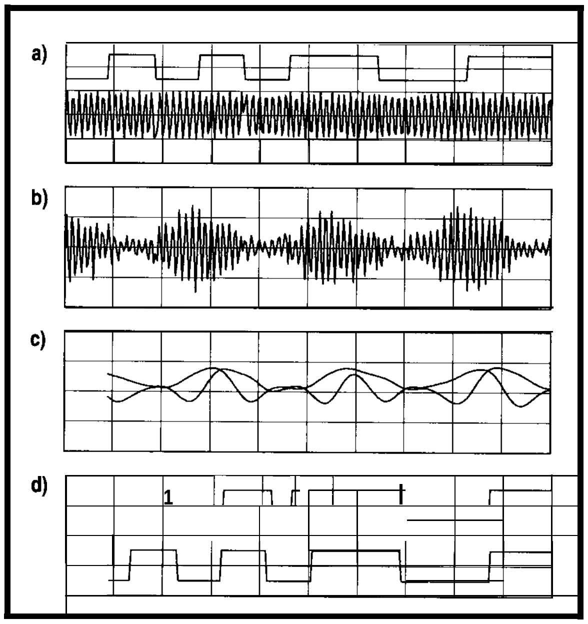


Figure 7a—*This originaldata and modem signal is created at the radio transmitter. b--Notice the change after noise and fading are added. c-These outputs are from the 2025- and 2225-Hz tone decoders. d-The upper signal is the recovered data produced by comparing the two signals in (c). The lower signal is the original data shifted by ha/f the sample window size.*

Once these two logical signals are established, I still have to duplicate the functionality of a UART's receive portion to identify the start, stop, and data bits. A hardware UART normally works by oversampling the data using a free-running clock that operates at 16 times the expected data rate.

Once the logic detects the 1-to-0 transition at the beginning of the start bit, it uses the clock to predict where the bits' centers are going to be and checks the value of the data line at those instants in time. A shift register captures the data bits and presents them to the CPU in parallel form.

I mentioned I'd use DDS to generate the sine and cosine waves at 2025 and 2225 Hz mainly because it's an easy way to get these frequencies. It doesn't really matter whether these DDS generators are frequency-locked to the master timebase. The difference of a few 10s of ppm doesn't make any real difference to the operation of these tone decoders.

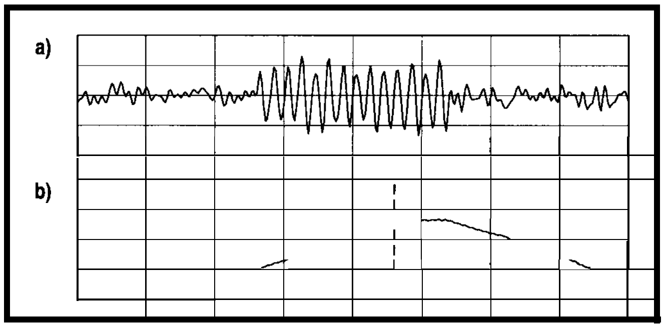


Figure 8a—This noisy, band-limitedtone burst is obtained from the radio receiver. b—This is the output of the 1000-Hz tone decoder, using an 80-sample window size and using straight integration as the low-pass filter. The vertical markers show an interval of interest.

ULTIMATE ACCURACY

Identifying the bit edges and decoding the data enables me to set the replica timebase to within -10 ms of the master timebase.

I should be able to get the accuracy to another order of magnitude (1 ms) by carefully searching for the leading and trailing edges of the 1000-Hz tone bursts. Since these bursts exist for a minimum of 10 ms, I can use a longer tone detector window for better noise rejection.

Figure 8 shows the results of such a tone decoder working on the noisy, band-limited signal from the radio receiver. I take the output signal from this decoder and try to identify two moments, separated by the window size, where the difference is greatest.

I adjust the positioning of these moments by altering the frequency of the DDS that's producing the replica 1000-Hz signal. Once that's done, the rest of the replica timebase can be derived from this DDS.

Once the timebase is set to within one cycle at 1000 Hz (1 ms), the final refinement would be to use the phase-angle measurements that we get from the tone decoder.

This change could theoretically get the accuracy down to a fraction of our sample period (< 100 us). However, the short-term radio path length variations may make this a moot point.

Good, Fast, Low Cost:

PICK 2

With PromICE, You CAN Have It All!

High performance memory emulation and debugging:
- Stable and reliable on today's embedded systems.
- New faster access speeds now standard.
- The best connection solutions for **TSOP, PSOP** and PLCC chips.
- Expanded Virtual **UART** support for industry-standard debuggers.

Ultra-Fast code downloads reduce development time:
- New high-speed download support for Windows NT
- 90 **KBytes/Second** over a PC parallel port.
- low-cost Ethernet support for UNIX systems.

New lower Prices for 1997:
- 128 **KByte PromICE** now just $495.
- Source-level debugging systems at a fraction of an ICE's cost.

Today's PromICE really is "better, faster, less expensive". Call 1.800.PROMICE (776-6423) for more information, or visit us at www.gei.com.

Voice: (614) 899-7878 • Fax: (614) 899-7888 • email: info@gei.com
921 Eastwind Drive • Suite 122 • Westerville, Ohio 43081 • USA
US and Canadian Sales: 1-800-PROMICE (776-6423)

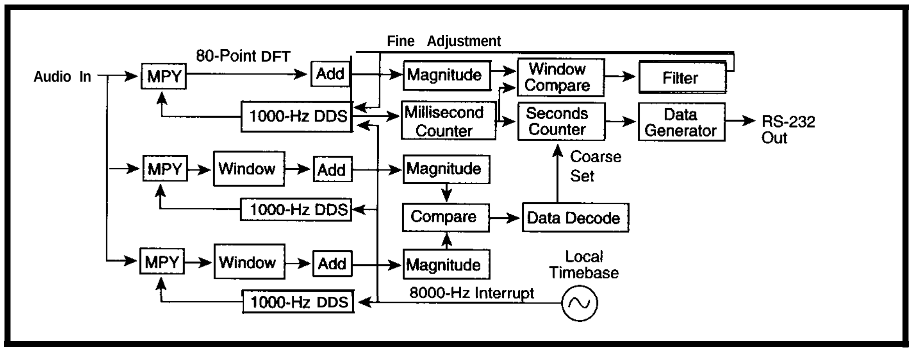


Figure 9-The final sofhvare *dataflow diagram shows the details of selected algorithms.*

GENERATING THE OUTPUT

Remember, the receiver's output signal was specified as RS-232C, using ASCII characters in an 8N1 configuration. The data rate was left unspecified, except to say that it will be between 300 and 9600 bps.

The output string is in the form (using C `printf()` notation):

```
\r%4d %3d %2d:%02d:%02d
```

where the individual fields are year, day of year, hours, minutes, and seconds UTC, using 24-hour notation.

I also stated that the string would be transmitted so that the last character ended just as the named second began. Using a bit-rate generator tied directly to the replica time (i.e., another DDS) would be an easy way to accomplish this. It would be just another small task running at the 8000-Hz sample rate of the rest of the system.

However, with this approach, the bit edges in the output signal can only occur exactly on the sample clocks—every 125 µs or multiples thereof. If I want to use standard computer data rates like 9600 bps, timing jitter at the bit edges distorts the bit widths.

Lower data rates minimize this distortion in terms of a percentage of the bit width. At 300 bps, the distortion is at most 3.75%, which should be acceptable to any computer SIO port.

PROJECT ENHANCEMENTS

Figure 9 shows the detailed software dataflow diagram with the tone detection and timebase filled in. You can see that the initial architecture wasn't far off. Now, it's specific enough for me to start thinking about implementation details.

The CHU receiver isn't quite complete. In this series, I wanted to make clear through the extensive use of graphics some of the theory and implementation issues behind some common DSP techniques.

This project is a basic CHU receiver that emits a simple time and date signal that's locked to the master timebase at the transmitter. It could be enhanced by:

- adding an automatic gain control to help correct for the effects of fading. It could be keyed on the amplitude of the 1000-Hz tone bursts in much the same way that a TV receiver keys on the sync tips.
- using a remotely-controllable radio receiver and scanning all three radio frequencies to select the best signal
- making the RS-232 output format configurable for different applications. This would include making more of the data available in the output (e.g., UT1 for astronomers).
- generating one or more continuous audio frequency reference outputs from the DDS via the DAC

And, if you build your own radio receiver, you can use RF carrier frequency and phase measurements.

I'm interested in any enhancements you might think of, so feel free to contact me. ❑

Dave Tweed has been developing real-time software for microprocessors for more than 18 years, starting with the 8008 in 1976. He currently designs equipment for carrying high-quality audio and wide-bandwidth data over digital telephone services such as T1 and ISDN. You may reach him at dave. tweed@circuitcellar.com.

SOFTWARE

The MathCad documents I used to generate graphics for this article are on the Circuit Cellar Web site.

REFERENCES

Radio station CHU, www.nrc.ca/inms/whatime.html.

SOURCES

MathCAD
MathSoft, Inc.
101 Main St.
Cambridge, MA 02142-1521
(617) 577-1017
Fax: (617) 577-8829
www.mathsoft.com

TRS-80 Model 100
Andy Diller's Web 100 Main Page
www.clark.net/pub/m100/pages/comp/m100/m100Main.html

ADSP-2101, ADSP-2181, EZ-Lab, EZ-Lab Lite
Analog Devices
One Technology Way
Norwood, MA 02062-9 106
(617) 329-4700
Fax: (617) 329-1241
www.analog.com

DSP56000
Motorola
6501 William Cannon Dr. W
MS 0E314
Austin, TX 78735-8598
(512) 891-2030
Fax: (512) 891-3877
www.mot.com/SPS/DSP/products/TMS320

TMS320 **series**
Texas Instruments, Inc.
34 Forest St.
MS 14-01
Attleboro, MA 02703
(508) 699-5269
Fax: (508) 699-5200
www.ti.com

I R S

407 Very Useful
408 Moderately Useful
409 Not Useful

NPC

DSP COPROCESSOR BOARD

The SPIRIT-32 is a low-cost, high-performance, PC/104-form-factor DSP module based on the Texas Instruments TMS320C32 DSP. With up to four channels of A/D and D/A, it's ideal for use with PC-compatible SBCs.

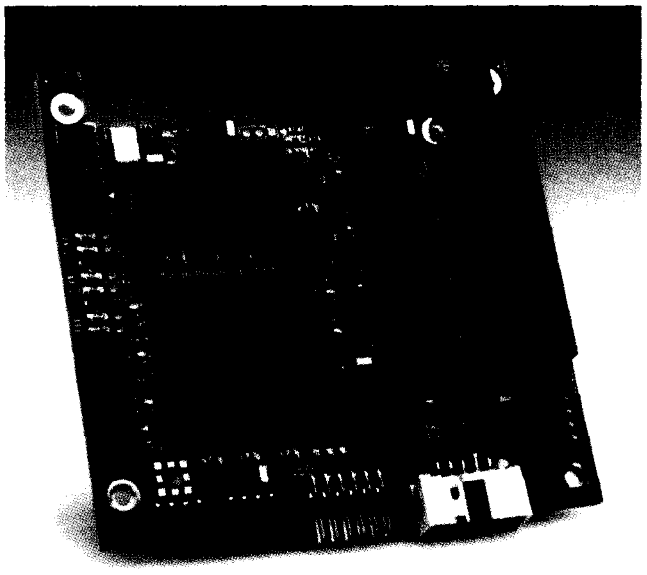
The SPIRIT-32 features a 32-bit floating-point processor with a 40-, 50-, or 60-MFLOPS performance rating, two banks of internal 256 x 32 zero-wait-state SRAM, 64 x 32 cache, flash memory, and an RS-232 interface off the main memory bus. It offers all the functionality of the TI DSP devices (e.g., a program debugging interface via the MPSD emulator port). As well, the chip's timers, interrupts, and software-controllable I/O flag lines are brought out to a processor expansion connector (PEC) on the module.

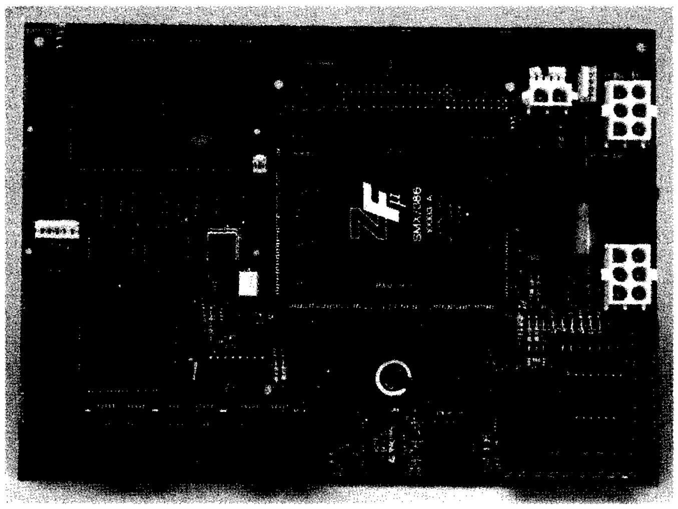
The suite of development tools for the SPIRIT-32 includes RadiSys's Brahma MPSD emulator/debugger, an RS-232 library for application development, and RadiSys's PC-based Run Time Library for DOS, Windows 95, or Windows NT, as well as TI's optimizing C compiler/assembler/linker for the C32. A DSP function library for the C32 DSP is also provided.

The 60-MHz SPIRIT-32 with standard 32 K x 32 SRAM sells for $795 in 100-piece quantities.

RadiSys Corp.
5445 NE Dawson Creek Dr.
Hillsboro, OR 97124
(503) 615-1100
Fax: (503) 615-1150
www.radisys.com

#510





SINGLE-BOARD COMPUTER

The **AKT/386** is a single board computer designed for rugged, mobile, embedded transportation applications. The board is builtaround theSMX/386 OEModule—a single-device PC (SDPC) that includes a 33-MHz 80386SX CPU, core logic, a DRAM controller, and 8- or 16-bit ISA busing. It also has serial and parallel I/O ports, floppy and IDE disk controllers, an AT-compatible BIOS, an embedded version of DOS, and 256 KB of flash memory.

The AKT/386 integrates the SMX/386 OEModule, seven serial ports, two parallel ports, 2.2 MB of flash memory configured as a solid-state disk, and support for up to 15 MB of additional external removable disk storage. The board is designed to cope with vibration, electrical noise, and power fluctuations.

The AKT/386 includes multiple industry-standard interfaces to maximize design flexibility. The full 16-bit ISA bus (accessible through a PC/104 connector) enables easy expansion with a wide choice of modules. An SAE J 1708 serial interface for communications between microcomputer systems in heavy-duty vehicle applications is included, as is a port for connection of a GPS.

The AKT/386 sells for $731 in quantity.

ZF **MicroSystems,** Inc.
1052 **Elwell** Ct.
Palo Alto, CA 94303
(415) 965-3800
Fax: (415) 965-4050
www.zfmicro.com

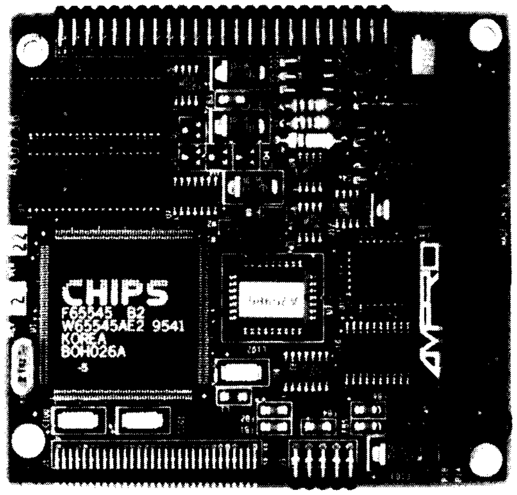
#511

NouveauPC

edited by Harv Weiner

GRAPHICS CONTROLLER

The **MiniModule/VFP-II** expansion module is a highly versatile flat-panel and CRT display controller that can interface PC/104-expandable embedded systems to color and monochrome LCD panels, color and monochrome electroluminescent (EL) displays, and analog CRTs. Simultaneous display output on both a CRT and a flat panel is also supported. Full software compatibility with five popular video standards—SuperVGA, VGA, EGA, CGA, and MGA—ensures easy system development and support in a broad range of embedded applications.



The module supports resolutions of up to 1280 x 1024 in 16 colors, as well as 24-bit true color in 640 x 480 resolution. In addition, the display controller includes a GUI accelerator that can significantly increase the performance of Windows and many other graphic-intensive applications.

A number of features to facilitate the incorporation of flat-panel displays into embedded applications are included. A programmable VGA BIOS in onboard flash memory supports the diverse signal timing and interface requirements of different flat-panel technologies and manufacturers. A variety of programmable display centering and stretching functions allows the use of displays whose resolutions exceed that of the software in use. Power management and sequencing functions are included to control the use of system power by the flat panel and to prevent damage to LCD panels during system power-up and powerdown. Software-programmable grayscaling, frame-rate control, and dithering functions provide highly effective color simulation on monochrome displays.

The MiniModule/VFP-II sells for $299 in quantity. A development kit for first-time purchasers, which comes with a comprehensive technical manual and software utilities, is available for $416.

Ampro Computers
990 **Almanor** Ave.
Sunnyvale, CA 94086
(408) 522-2 100
Fax: (408) **720-** 1305

#512

PC/I 04 RESOURCE GUIDE

The PC/I 04 Consortium announces the tenth edition of its PC/104 Resource Guide. The free 225-page book is also available on CD-ROM. In addition to overviewing the PC/I 04 standard, this year's edition details PC/I 04's PCI equivalent, PC/I 04-*Plus*.

To order, call or fax the PC/I 04 Consortium at:

(415) 903-8304
Fax: (415) 967-0995

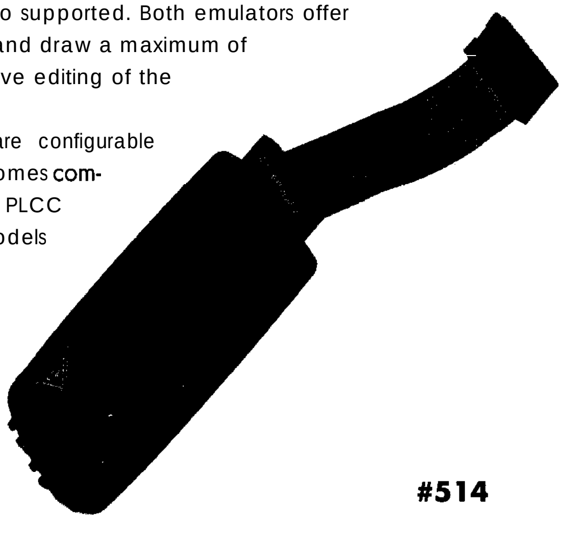
#513

LOW-VOLTAGE EPROM EMULATOR

Scanlon Design has introduced two low-voltage EPROM emulators. The El LV-90 and **E4LV-90** enable operation at 3.3 and 5 V, respectively, and emulate EPROMs up to 1 (El) and 4 Mb (E4). RAM and flash memory are also supported. Both emulators offer error checking and correction while downloading and draw a maximum of 5 mA. The emulators include software that permits live editing of the emulation memory.

These compact emulators are completely software configurable with an integrated memory back-up circuit. Each comes complete with 28- and 32-pin DIP adapters, and 32-pin PLCC adapters are available. Fast access time (30 ns) models are also available.

The emulators retail from $229.



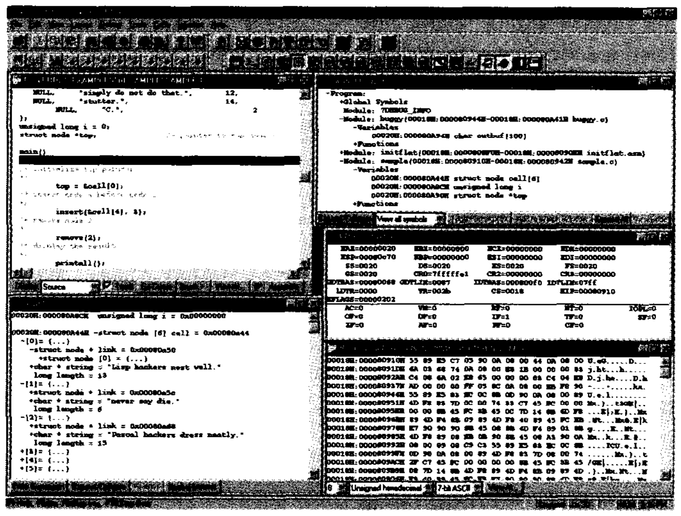
Scanlon Design, Inc.
5224 Blowers St.
Halifax, NS • Canada **B3J1J7**
(902) 425-3938
Fax: (902) 425-4098

#514

Nouveau-PC

REMOTE SOFTWARE DEBUGGER

VisualProbe Remote Debugger is a Windows-hosted GUI source-level debugger for embedded-system software development. A Remote Debugger enables an embedded application to be downloaded from the host PC and debugged on the target system. Program execution on the target system is controlled by the Target Monitor, which communicates with the host via an RS-232



```
    NULL,     "simply do not do that.",       12,
    NULL,     "stutter.",                     14,
        NULL,     "C.",                             2
};
unsigned long i = 0;
struct node *top;

main()

        top = &cell[0];

        insert(&cell[4], 3);

        remove(2);

        printall();
```

```
00020H:000080A8CH  unsigned long i = 0x00000000

00020H:000080A44H -struct node [6] cell = 0x00080a44
 -[0]= {...}
   -struct node * link = 0x00080a50
     +struct node [0] = {...}
   +char * string = "Lisp hackers nest well."
    long length = 13
 -[1]= {...}
   +struct node * link = 0x00080a5c
   +char * string = "never say die."
    long length = 6
 -[2]= {...}
   +struct node * link = 0x00080a68
   +char * string = "Pascal hackers dress neatly."
    long length = 15
 +[3]= {...}
 +[4]= {...}
 +[5]= {...}
```

```
-Program:
   +Global Symbols
    Module: ?DEBUG_INFO
   -Module: buggy(00018H:00008094 4H-00018H:000080A41H buggy.c)
      -Variables
          00020H:000080A94H char outbuf[100]
      +Functions
   +Module: initflat(00018H:000080 0FOH-00018H:0000809 0EH initflat.asm)
   -Module: sample(00018H:00008091 0H-00018H:00008094 2H sample.c)
      -Variables
          00020H:000080A44H struct node cell[6]
          00020H:000080A8CH unsigned long i
          00020H:000080A90H struct node *top
      +Functions
```

```
   EAX=00000020      EBX=00000000      ECX=00000000      EDX=00000000
   ESP=00080c70      EBP=00000000      ESI=00000000      EDI=00000000
    SS=0020           DS=0020           ES=0020           FS=0020
    GS=0020          CR0=7ffffffe1     CR2=00000000      CR3=00000000
GDTBAS=00000060 GDTLIM=0087       IDTBAS=000800f0 IDTLIM=07ff
  LDTR=0000           TR=002b           CS=0018          EIP=00080910
EFLAGS=00000202
    AC=0            VM=0              RF=0              NT=0           IOPL=0
    OF=0            DF=0              IF=1              TF=0             SF=0
    ZF=0            AF=0              PF=0              CF=0
```

```
00018H:00008091 0H 55 89 E5 C7 05 90 0A 08 00 44 0A 08 00 U.e.....D...
00018H:0000809 1DH 6A 01 68 74 0A 08 00 E8 1B 00 00 00 83 j.ht...h.....
00018H:0000809 2AH C4 08 6A 02 E8 65 00 00 00 83 C4 04 E8 D.j.he....D.h
00018H:0000809 37H AD 00 00 00 FF 05 8C 0A 08 00 EB F8 90 -.......ka.
00018H:0000809 44H 55 89 E5 83 EC 0C 8B 0D 90 0A 08 00 89 U.e.l.......
00018H:0000809 51H 4D F8 8B 7D 0C 00 74 83 C7 45 FC 00 00 Mx.)..tAGE|..
00018H:0000809 5EH 00 00 8B 45 FC 8B 45 0C 7D 14 8B 4D F8 ...E|;E.)..Mx
00018H:0000809 6BH 89 4D F4 8B 07 89 4D F8 40 89 45 FC EB .Mt...Mu@.E|k
00018H:0000809 78H E7 90 90 90 8B 45 08 8B 4D F4 89 01 8B g....E..Mt...
00018H:0000809 85H 4D F8 89 08 EB 0A 90 8B 45 08 A3 90 0A Mx..k...E.#..
00018H:0000809 92H 08 00 89 08 C9 C3 55 89 E5 83 EC 0C 8B ....ICU.e.l..
00018H:0000809 9FH 0D 90 0A 08 00 89 4D F8 83 7D 08 00 74 ......Mx.)..t
00018H:000080 9ACH 2F C7 45 FC 00 00 00 00 8B 45 FC 3B 45 /GE|.....E|;E
00018H:0000809 B9H 08 7D 14 8B 4D F8 89 4D F4 8B 09 89 4D .)..Mx.Mt...M
```

serial port. VisualProbe has a number of preconfigured target monitors, and it can enable the user to build a custom target monitor.

VisualProbe provides source-level and symbolic debugging for C and mangled C++. It also provides various windows to show code in source, disassembler, or mixed views, as well as define breakpoints based on symbolic address, expressions, and debug register expressions. These windows view memory in various formats, evaluate and view C expressions, and show active chain of function calls and active local variables. They also let the user read and write any I/O port, as well as view and edit symbols.

VisualProbe may be hosted on systems running Windows 95, NT, or 3.x. The software is supported by SSI's absolute linker/loader and Link&Locate '386. It accepts files in the Intel OMF86/286/386 bootloadable formats along with the binary image formats.

VisualProbe Remote Debugger sells for $1795.

Systems & Software, Inc.
18012 Cowan, Ste. 100
Irvine, CA 926 14
(714) 833-1700 • Fax: (714) 833-1900
www.ssi.com

#515

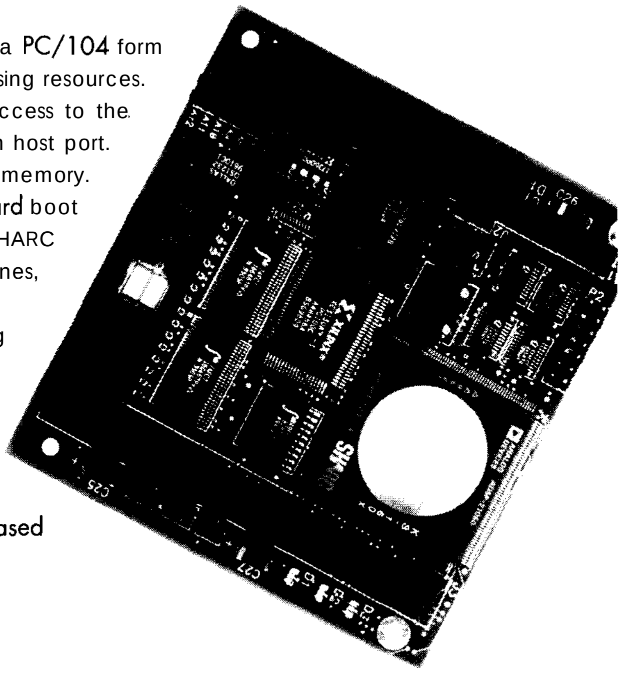
SHARC DSP MODULE

BittWare Research's **Silvertip** is a powerful floating-point DSP module in a PC/104 form factor. Based on the Analog Devices SHARC DSP, it offers 120-MFLOPS processing resources.

Silvertip's PC/104 bus interface gives host computers direct 16-bit access to the onboard 40-MHz SHARC processor's 32-bit-wide IOP registers and DMA-driven host port. The host can reset and boot the SHARC, load program images, and examine memory. Silvertip can operate with a PC/104 host or in stand-alone mode using its onboard boot flash. The board's minibus expansion connector extends a portion of the SHARC processor bus to additional I/O or memory devices via 22 address and 32 data lines, plus two interrupt inputs, two serial ports, and a SHARC link port.

Silvertip is supported by sourcecode development tools including Analog Devices' SHARC ANSI-compliant C compiler assembler, linker, simulator, and source-code debugger. True real-time in-circuit emulation is available with the optional EZ-ICE emulator. BittWare's DSP21k Toolkit provides C-callable host I/O and DSP functions, sample code, and diagnostic utilities under DOS and Windows 3.1, 95, and NT.

List price for an ADSP-21062-based Silvertip is $1695. An ADSP-21060-based version is available for $2495.



BittWare Research Systems
33 N. Main St.
Concord, NH 03301
(603) 226-0404
Fax: (603) 226-6667
www.bittware.com

#516

Chip Freitag

A Stand-Alone Embedded Ethernet Platform

Adding Ethernet to an embedded design gives you *control* via he Internet. Chip brings together an embedded Ethernet platform and an Am 7 86. He gives guidelines for layout, chip clocking, *data* storage, as we*ll* us *software* concerns.

With the explosion of the Internet, the need for embedded Ethernet connectivity is becoming more common.

Ethernet networking provides a convenient, standardized means of connecting diverse systems-from software development tools, to point-of-sale systems, to the much anticipated "smart house." **It's also** widely used to interconnectsubsystem components in larger designs.

In this article, I explain how to design a simple, low-cost, standalone Ethernet platform using the Am1 86 family of micro controllers with the Am79C940 Media Access Controller for Ethernet (MACE).

Using the microcontroller's features, this platform can be the brains of a variety of embedded devices. Ifyourembedded application already uses a '186, you'll see how Ethernet can easily be **added** to your design.

Of course, the hardware is just the start of the solution. The popularity of Ethernet as a system-connectivity solution lies in the universal acceptance of the Internet Protocol (IP) standard.

Compliant Ethernet devices rely on the services of IP, TCP, UDP, and other well-understood and readily available protocols. So, I also discuss software issues such as device drivers and protocol stacks.

HARDWARE CONNECTION

Figure 1 diagrams a complete Ethernet solution for an embedded microcontroller application as well as the interface be tween the Am1 86ES and the MACE.

The Am 186ES is a high-performance 16-bit 'x86 microcontroller available in 20–40-MHz speed grades. **The integrated peripherals and glueless interface to memory make it an ideal solution for many embedded devices.**

The MACE is a highly integrated slave-type Ethernet controller incorporating the logical MAC and PHY layer (Manchester encoder/decoder and 1 OBaseT transceiver). The 16-bit interface makes connection to a 'x86-style Local bus straightforward.

The microcontroller section consists of the '186ES, flash, and SRAM. This configuration includes 256-KB flash memory and 256-KB SRAM, but **of** course, exact memory size varies by application.

Some designs eliminate the flash and download the microcontroller'scodedirectly into SRAM. This task is accomplished by asserting HOLD or RESET to the processor to gain control of the processor's memory bus.

A typical general-purpose TCP/IP stack requires about 48 KB of code and 48 KB of data memory. So, most **embedded** Ethernet applications can fit in **128 KB of total memory space.**

The Ethernet section consists of a PAL for glue logic and the Am79C940 MACE. The design relies on the Am 186ES's integrated DMA channels for high-performance data movement between the MACE and the microcontroller's memory.

Support circuitry not shown (e.g., the RS-232 interface, Ethernet isolation transformers, and RJ45 connector for 1 OBaseT) is covered in app notes available from AMD. Also not shown is the rest of the application-the embedded target design itself.

Just keep in mind that the microcontroller performs other duties as well. Their interaction with the Ethernet and TCP/IP tasks must be taken into account.

My design uses the Am1 86ES with SRAM, which is a good combination when RAM requirements are small. But, if your RAM requirements exceed 128 KB, the Am1 86ED plus DRAM is a more cost-effective solution.

DESIGN OBSERVATIONS

Al of the microcontroller bus connects to A0 on the MACE processor interface. Thus, all accesses to the MACE's internal 8-bit registers are on even addresses. From the microcontroller's perspective, they can be 8- or 16-bit accesses.

The least significant byte of such accesses contains the valid 8-bit data. The most significant byte can be ignored. The MACE register chip select (*CS) connects to *PCS3 on the microcontroller.

Accesses to the MACE's FIFOs are accomplished via the '186ES DMA channels. One channel is responsible for transmitting data, and the other for receiving data.

Both DMA channels should address PCS2. For receive operation, DMA channel 0 should have its source address set to PCS2 and its destination address set to the software-supplied buffer memory address. Fortransmit, DMA channel 1 should have its source address set to the supplied buffer memory address and its destination set to PCS2.

*PCS2 connects to the MACE's FIFO Data Strobe (*FDS). Thus, both DMA channels address the FIFOs on the MACE. As you see in Listing 1, a term in the PAL equations ensures that R/*W on the MACE is driven correctly during these accesses.

With a 40-MHz '186ES, two wait states must be inserted for PCS2 and PCS3 cycles. *TC is pulled down on the MACE, resulting in three-cycle MACE timing

*EOF is supplied, to the MACE during writes, according to the setting of PIO25. For transmit operations, the MACE device driver should set the transfer counter for the write DMA channel to 1 less than the numberofwords to write. PIO25 is set to 0, meaning *EOF is inactive.

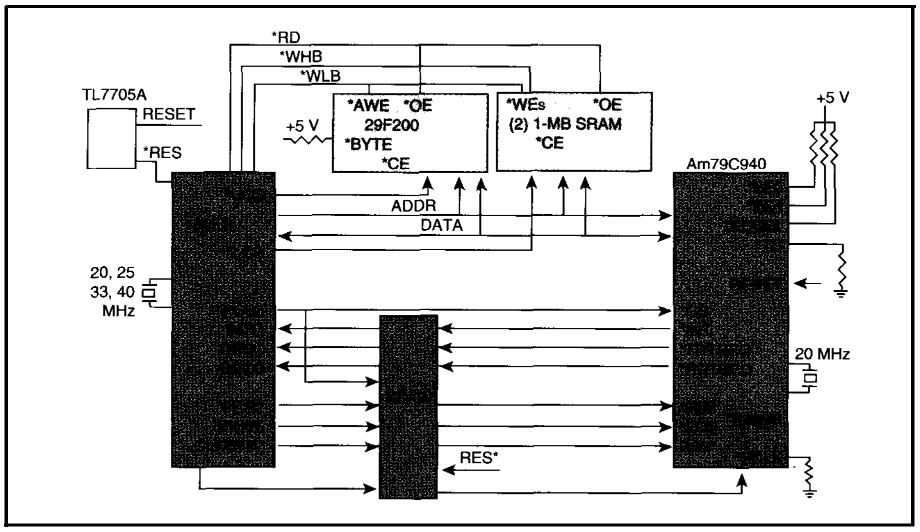


Figure I-This simplified diagram ***shows how to connect a Am 186ES to the Am79C940 MACE.***

After the DMA channel starts and the MACE begins transmitting, a DMA terminal-count interrupt occurs. In this event's ISR, PI025 is set to 1 and the DMA channel is set to transfer one more word. *EOF is active during this transfer, and the MACE recognizes the end of the transmit packet.

This design uses two separate clock sources-one for the microcontroller and the 20-MHz source for the MACE XTAL1 and XTAL2 inputs.

If your design deviates from this arrangement, ensure that you still meet the stringent requirements for an external clock source for the MACE. Improper Ethernet chip clocking can be a major source of equipment incompatibility and is often difficult to track down as the cause of equip ment malfunction.

The *EAD/R pin is tied low as per its description in the MACE manual. This assumes that the external address matching feature is not used.

Associated register bits must be correctly set by software. This includes the Match/ Reject bit in the Receive Control Register, which should be left at its default. This configuration allows internal address matching (or promiscuous mode) to override the external address detection.

The PAL equations in Listing 1 contain a state machine that resolves a MACE receive EOF issue. After the processor DMA reads the last word from the MACE's receive FIFO, the MACE doesn't deassert RDTREQ for up to four cycles-long enough to inadvertently latch one more DMA request.

If this DMA transfer is allowed to occur, the first of the receive status bytes will be read and placed into the DMA buffer. Software must then be aware of and recover from this event.

However, the PAL state-machine intercepts RDTREQ and disallows DRQO for seven cycles after the receive EOF. This feature prevents inadvertent DMA cycles, eliminating the need for a software workaround.

NONVOLATILE STORAGE

Many Ethernet applications require some type of writable nonvolatile memory, usually for IP addresses or other configuration-specific set-up information. Accesses to nonvolatile **memoryarelypically** infrequent, being done mostly at powerup or when the end user changes the configuration.

EEPROM is a popular method for storing such data. Parallel EEPROMs can be connected to the address and data bus and driven with one of the available chip selects.

For serial EEPROMs, it's relatively **easy** to use the PIOs to implement a serial interface. A driver for such a PIO-style serial interface is simple to write.

Of course, nonvolatile data can also be stored in

Vendor	Product	Protocols Supported
US Software	USNet	TCP/IP,ftp, telnet, ping,SNMP
EBS, Inc.	RT-IP	TCP/IP, ftp, telnet, PPP
Accelerated Technology	Nucleus Net	TCP/IP
Pacific Sottworks	Fusion TCP/IP	TCP/IP, SNMP, PPP, SMTP
XLNT	Stackware	TCP/IP, UDP, telnet, ping
Integrated Systems	Attache +	IP, UDP, TCP

Table ***1—Here*** is a list of the more well-known ***TCP/IP protocol stack*** vendors. These stacks are available in ***C,*** and all offer excellent support for porting to ***different*** targets.

flash. This technique has the advantage of not requiring another device, which can help keep costs down. Software is then required to manage the task of writing the saved data to the flash device.

BOARD LAYOUT

Once the design is on paper and you're ready to lay out the board, take time to learn about the recommended layout practices for Ethernet designs (see References).

In a nutshell, Ethernet interface devices contain both analog and digital circuitry. Typically, a device's analog portions are confined to an isolated part of the chip, which helps reduce digitally induced noise on the sensitive Ethernet-analog physical interface.

On the PCB, it's common practice to provide isolated analog power and ground planes. Separate powersupplydecoupling for the **analog** section is also recommended.

POWERUP AND DEBUG

When boards return from assembly, take a few common-sense steps to verify that your Ethernet design works properly.

First, get the microcontroller section running correctly. It should be fetching instructions from flash and correctly accessing SRAM.

Most designs use a monitor to aid in development. Typically, this monitor uses oneoftheserial ports built into the Am186ES and provides a user interface that can be displayed using an ANSI terminal or emulation program (e.g., Hyperterm).

Once the monitor is booting from flash and running properly, it's time to check out the Ethernet section. The design in Figure 1 has the MACE's register interface on Peripheral Chip Select 3.

Aftercorrectlyconfiguring the Am186ES PACS (offset A4h) and MPCS (offset A8h) for **proper operation,** you can try to read the MACE registers. Usually, the Am186ES registers are configured to place the MACE in I/O space at 300h.

If you're using this configuration, performing an I/O read at 320h using the monitor's IN command returns thecontents of the MACE chip-ID LSB register. It should be a 40h.

An I/O read from 322h should return x9, where x is dependent on the MACE version. Then, check out MACE writes by writing to a MACE read/write register and reading it back.

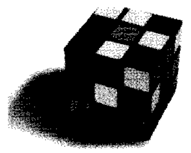
If you get these results correctly, it's time to get **your** MACE driver software working. After porting, the best testing method is to link a simple test application that includes the driver, TCP/IP stack, and ping application.

Usually, the main() program can do nothing more than initialize the stack and hardware interface and then just wait. As part of porting the stack and driver, you have to supply a unique Ethernet 802.3 MAC address, an IP address, and possibly a name domain address.

Hook your target to a simple network consisting of a UNIX or PC host, a hub, and possibly a network analyzer. Then, ping your target at the appropriate address.

If you don't get a reply, double check the networkconfiguration.Makesureyourdriver code is configuring the MACE correctly. Pay special attention to the address configuration since it's a common cause of problems.

Next, see if the MACE is asserting RDTREQ. If so, then it's receiving data and



Par•a•digm: your source for the most high-powered, comprehensive set of time-saving software and hardware development tools for embedded application development.

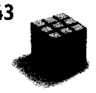
1: **Paradigm LOCATE** the most popular tool for creating embedded C/C++ applications with Borland and Microsoft compilers; **2: Paradigm DEBUG** the only x86 debugger with C++, RTOS, scripting language, and full in-circuit emulator support; **3: Paradigm SUPPORT** the best technical support in the industry 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 AMDx86 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



#116

asking the microcontroller to DMA it to memory.

After verifying that DMA is configured and operating correctly, check that the MACE is signaling the end of packet by asserting the interrupt to the Am186ES. If everything is happening correctly, begin debugging any software problems you might have.

By working upwards from the lowest levels of the hardware into the software, you can systematically find and eliminate problems.

By the time you're done, you should be getting responses to your ping requests. Setting breakpoints or debug messages in your ping application code verifies that your IP stack is working properly.

It's then time to try an ftp or other data-intensive test. Once it works correctly, you can proclaim the Ethernet section operational and move on to integrating the Ethernet with your embedded application.

You aren't quite done with the hardware, but the software tasks can now proceed. To be truly finished with the hardware, your Ethernet interface must meet the 802.3 specs.

SOFTWARE CONSIDERATIONS

In addition to application software, an embedded Ethernet design requires at least Ethernet drivers, almost always a network protocol stack, and probably a real-time operating system.

Choosing a protocol stack is a matter of matching design requirements and budget to available sources. Freeware stacks reduce initial costs, but they can take a lot of time to install correctly and may suffer from terrible performance.

At the other end of the spectrum, vendors such as Integrated Systems offer a wide range of high-level protocols (e.g., RMON) as well as porting and integration services (see Table 1). High-end stacks may cost more, but they're worth it if performance and time-to-market are critical.

All commercial protocol stacks have a few targetdependent modules that provide independence from hardware drivers and RTOS APIs. To interface to a new hardware driver or RTOS, only these interface modules need to be changed so that generic calls(e.g.,send_packetortask_wait) are replaced by the calls specific to the given driver and RTOS.

Therefore, you can use pretty much any protocol stack with any combination of hardware drivers and RTOS. Most stacks also come with drivers for the most common Ethernet chips and interface modules for the most popular RTOSs.

A base set typically includes basic protocols and applications like IP, UDP, ping, and telnet. You pay extra for extensions like TCP, ftp, PPP, SNMP, RMON, and other higher level protocols and applications.

There are also shareware and freeware RTOSs available. One system, Packet Driver, is not a TCP/IP stack itself, but if you unzipthefileandlookatsoftware.doc, you find a list of various protocol stacks (including several TCP/IP packages) and other applications supporting Packet Driver. Some are suitable for embedded applications.

NETWORK CONTROL

The design presented here is just a starting point. Adding Ethernet to an embedded design opens a whole new world of possibilities for the target system.

A recent AMD project included a port of US Software's USNet TCP/IP stack. After we ported the TCP/IP stack, engineers at US Software loaned us a copy of their recently developed HTTP server application.

We ported it to our demo board and wrote an HTML page to enable a Web browser to change the state of the micro's programmable I/O pins. We also wrote a CGI script for the board to return an HTML page showing the current state of the PIOs.

On this 3" x 3" demo board, the PIOs being controlled are connected to LEDs.

Listing I-These PAL Equations ***for*** the Am ***186ES/MACE*** design are ***written*** in ***PALASM*** for a ***PALCE22V10.***

```
 ——————————————Declaration Segment————
TITLE 186ES to MACE Glue Logic
DEVICE  22V10  PLCC
;——————————————PIN Declarations————————
PIN  2 clka       ; clk from 186EM
PIN  3 pcs2_      ; chip select from 186ES, DMA to/from MACE
PIN  4 psc3_      ; chip select from 186ES, regular cs to/from MACE
PIN  5 pio25      ; from 186ES, used to drive EOF on DMA writes
PIN  6 dtr_       ; data direction from 186ES, used to generate rw_
PIN  7 rdtreq_    ; receive DMA request from MACE
PIN  9 tdtreq_    ; transmit DMA request from MACE
PIN 10 int_       ; interrupt request from MACE
PIN 11 reset_     ; reset, for state machine
PIN 17 X0         ; state machine term
PIN 18 Xl         ; state machine term
PIN 19 X2         ; state machine term
PIN 20 int0       ; interrupt request to 186ES
PIN 21 drq0       ; receive DMA request to 186ES
PIN 23 drql       ; transmit DMA request to 186ES
PIN 24 fds_       ; FIFO data strobe for MACE
PIN 25 eof_       ; EOF for MACE
PIN 26 rw_        ; read/write for MACE
PIN 27 sclk       ; clka/2 for MACE

EQUATIONS         ; Boolean Equation Segment
int0 = /int_      ; invert interrupt pin
drql = /tdtreq_   ;  invert transmit dma request
fds_ = pcs2_      ; MACE FIFO data strobe is /pcs2
rw_ = /dtr_*(/pcs2+/pcs3)     ; build rw_ from chip selects and dtr_
eof_ = /pio25     ;eof follows sense of pio25 when enabled
eof_.TRST = /fds_ * dtr_      ; enable EOF as output on FIFO writes
sclk.clkf = clka
sclk := /sclk     ; divide 186 clock by 2 to get 20-MHz clock for MACE
drq0 = /rdtreq_ * /X0*/X1*/X2; allow DMA rec req only in state 0
XO.CLKF = CLK     ; use clka as the master clock for registered
X1.CLKF = CLK     ; outputs
X2.CLKF = CLK

; Downcounter state machine idles in state 0. When set to state 7,
; it counts down on each clock to state 0. The disable state machine
; gets set to state 7 when receive EOF occurs.
; drq0 to the Am186ES is allowed only in state 0.
GLOBAL.RSTF = RESET          ; reset configures to state zero
GLOBAL.SETF = (rdtreq_*eof_*pcs2_); receive EOF sets it to state 7
X0 := /X0 * X1 * x2 + /X0 * /Xl * x2 + /X0 * Xl * /X2
Xl := X0 * X1 * x2 + /X0 * /Xl * x2 + X0 * X1 * /x2
x2 := X0 * Xl * x2 + /X0 * Xl * x2 + X0 * /X1 * x2
```

So, any user with Netscape and the right URL can toggle the board's LEDs. Voilà, the world's fanciest blinking-light demo!

While this exercise might seem silly, it clearly shows how to provide a Web front end for any embedded application. Once you can control a PIO, you can control anything. EPC

Chip *Freitag is a* **marketing** engineer in *AMD's* embedded-processorgroup, specializing in networking **and telecommunications** applications. Previously, Chip was a software engineer *at Andrew/KM W* where **he** worked on 5250 terminal emulation and high-speedpageprinteremulationproducts *for* IO *years. You may* reach him at **chip.** *freitag@amd.com.*

SOFTWARE

For o list of shareware/freeware RTOSs, see <www.eg3.com/realxrto.htm>. For freeware protocol stacks, download PacketDriver from <www.crynwr.com/crynwr/>.

REFERENCES

Texts

AMD, "Embedded Network Applications De **sign Guide** Kit," PID 20397A, 1996.

AMD, "Am79C940 Media Access Controller for Ethernet," PID 16235C, 1994.

M.A. Miller, *Internetworking,* M&T Books, Redwood City, CA, 199 1.

W.R. Stevens, *TCP/IP Illustrated,* Addison-Wesley Publishing, Reading, MA, 1994.

Internet

www-cne.gmu.edu/modules/network/index. html

comp.arch.embedded

comp.protocols.tcp-ip

SOURCES

Am 186 controllers, **Am79C940** MACE
Advanced Micro Devices, Inc.
One AMD Pl.
Sunnyvale, CA 94088
(408) 732-2400
www.amd.com

USNet
US Software Corp.
142 15 NW Science Park Dr.
Portlond, OR 97229
(503) 64 l-8446
Fax: (503) 6462413

RT-IP
EBSNet, Inc.
P.O. Box 873
Groton, MA 0 1450
(508) 448-9340
Fax: (508) 448-6376
peter@etcbin.com
www.etcbin.com

Nucleus Net
Accelerated Technology, Inc.
P.O. Box 850245
Mobile, AL 36625
(205) 66 l-5770
Fax: (205) 6615788

Fusion **TCP/IP**
Pacific Softworks
4000 Via Pescador
Camarillo, CA 93012
(805) 484-2 128
Fax: (805) 484-3929
sales@pacificsw.com

Stackware
XLNT Designs
15050 Avenue of Science
San Diego, CA 92 128
(619) 487-9320
Fox: (6 19) 487-9768

Attache+
Integrated Systems
201 Moffett Park Dr.
Sunnyvale, CA 94089
(408) 542-l 500
Fax: (408) 542-l 961
www.isi.com

IRS

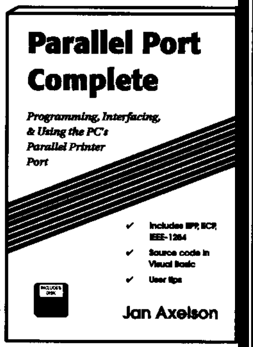
4 10 Very Useful
41 1 Moderately Useful
4 12 Not Useful

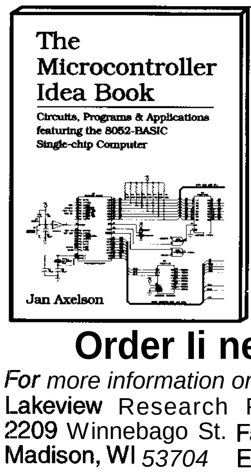
JUMP START YOUR DESIGNS with these practical guides

NEW! "A focused book that delivers what it promises: detailed technical information on the parallel port." - *Windows Developer's Journal*

"It's been a while since I've seen a book as practical as this one." *-Nuts & Volts*

ISBN O-9650819-1-5
$39.95
343 pages
Includes disk


Parallel Port Complete
Programming, Interfacing, & Using the PC's Parallel Printer Port
Includes EPP, ECP, IEEE-1284
Source code in Visual Basic
User tips
Jan Axelson


The Microcontroller Idea Book
Circuits, Programs & Applications featuring the 8052-BASIC Single-chip Computer
Jan Axelson

MICROCONTROLLERS!

"An ideal introduction to low-end embedded design." *- EDN*

"The writing is a model of clarity and conciseness."- *PC Magazine*

ISBN 0-9650819-O-7
273 pages. $31.95

Order li ne: l-800-247-6553

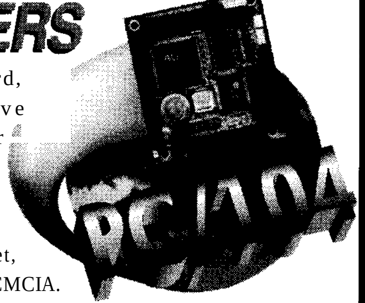
For more information or international orders:
Lakeview Research Phone 608-241-5824
2209 Winnebago St. Fax *608-241-5848*
Madison, WI *53704* Email jaxelson@lvr.com

Shipping: $5.00 *(U.S. & Canada)*

Read sample chapters on the web:
http://www.lvr.com

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 I%x/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

#117

David Feldman

A Formula for Winning

Product—Development Strategies

Sometimes, even *when* **opportunity** *stares us in the face, it's difficult to get started. David gives us all a jump start by guiding us through a design process* **thatmakessense for** INK*'s* **Embedded PC** *Design Contest and the day-to-day job.*

You entering Circuit Cellar *INK's* Embedded *PC* Design contest?

The goal of the contest is to encourage you to design unique applications that are both useful and creative. If you're successful and judged to be one of the winners, you profit (in cash).

Does this sound like your job? If you're the typical design engineer, this is exactly what you do every day.

Out in the real world, a winner's products get to market ahead of the competition, the company prospers, and you profit in cash and job security. Your company becomes a market leader.

COMPETE AGAINST TIME

But, how do winners stay leaders? What do they do that's so different?

Winners compete against time, so they don't have to worry about competitors. Numerous studies show that timebased companies consistently outperform their industry.

They move quickly and focus their resources on the items that provide the greatest value added in their market. They don't constantly reinvent the wheel by designing components or subsystems they can find cheaper. instead, corporate winners reinvent the market and leave everyone else playing catch up.

Advantages to accelerating product development? Getting there before the competition brings benefits which may not immediately come to mind but which have a dramatic impact on a company's ultimate success or failure.

But, most importantly, **the first** product to market is always in the enviable position of having 100% market share until competitors appear. It sets the standard.

The followers often have little choice but to give up margins. In effect, they have to buy market share with lost profits while suffering the added injury of having to claim "full compatibility with the leader."

Getting there first also extends your product's life. Every month of the develop ment cycle that's eliminated represents a month added to its sales and profit life. The leader thus achieves and maintains the greatest market share, which usually leads to the greatest return on investment.

Each customer added to the user base because of you getting to market first becomes a loyal user. They have a natural reluctance to switch to another product.

But to win, you need to take a step back and look at the big picture. Where will this product fit in the grand scheme?

Is it a one-of-a-kind, money-is-no-object research project? Is it a low volume, high-end product that can tolerate fairly high production costs?

Perhaps it's going to be a high-volume, cost-sensitive, market-sharegrabbing unit that will establish your company as the unchallenged leader in a segment.

'x86 AND THE EMBEDDED MARKET

Market pressures are propelling the growth of the 'x86 architecture in the embedded market. Although far from be ing the ideal solution for all embedded applications, its growth has been driven by product development time and cost constraints.

Most engineers know the 'x86 family because of using 'x86 desktop machines. This abundance of "humanware" added to the most cost-effective hardware and software available is the lure for embedded-system OEMs.

So far, 'x86 hardwareapplications have been limited only by the imagination! They appear in telecommunications equipment, process control, portable instruments, data logging, medical instruments, gaming machines, vending machines, and navigation systems-I could go on and on.

The market's rapid growth has increased demands on the design team to create state-of-the-art products, minimize costs, minimize risk, and shorten time-to-market.

The typical product using embedded PC hardware and software does not rely solely on off-the-shelf items. The greatest value added should always come from the proprietary content developed by the OEM.

DESIGN CHOICES

Designing a leading-edge product often doesn't begin with leading-edge technology. In fact, the embedded market usuallyrelieson technologyabandoned by the desktop market.

You need to consider a number of design choices if you're going to arrive at the ideal solution to a given problem. Any project can encompass one or all of the options available, depending on market pressures and total anticipated manufacturing volume over the life of a product.

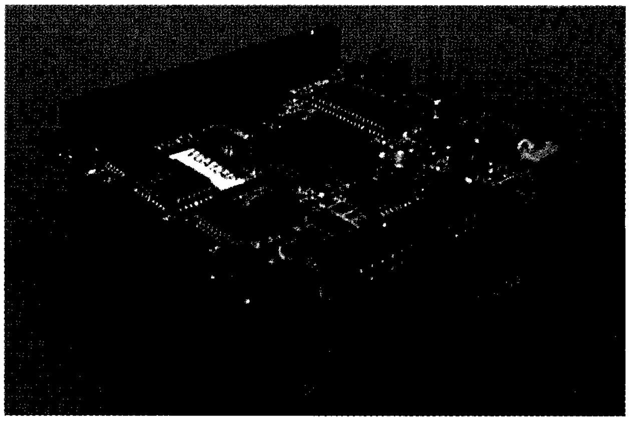


Photo 1—As you can see, PC/104 modules stack together quickly, providing rapid entry into markets for small medical instruments, data loggers, security systems, point- of-sale terminals, communications devices, and ***vehicu***lar navigation systems.

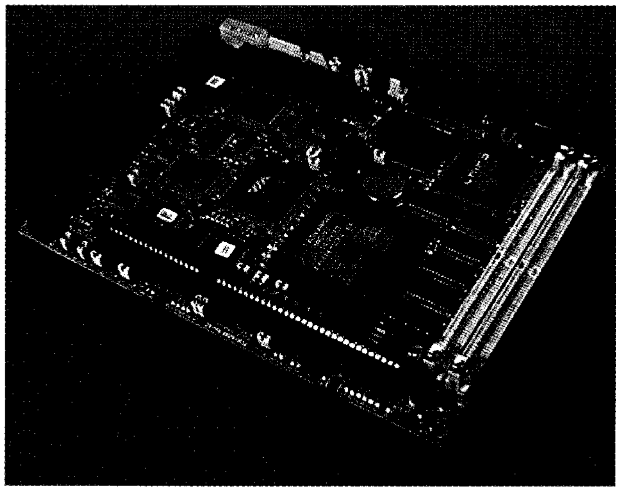


Photo ***2—This*** high-performance ***100-MHz 486DX4-based*** SBC incorporating all the functionality of a typical desktop system targets applications like medical imaging and high-speed test equipment.

Taking the time to make the right decisions about software and hardware is crucial. When you select off-the-shelf hardware and software, you enter a partnership. Your partner is the hardware or software supplier whose products you're incorporating.

Go past the specs to ensure there'll be adequate support both at the front end and long term.

SOFTWARE SELECTION

What do you want-real time, a DOS tailored for the embedded market, or plain-vanilla MS-DOS? A GUI?

Given that you're entering the Embedded PC contest, I assume your goal is a PC-compatible system. But, what's that mean?

In an embedded application, you're prob ably not concerned with running desktop word-processing or spreadsheet software. Games are also likely out of the question.

Usually, you want to write the custom embedded application on a desktop PC and possibly interpret the data collected by the product on a similar system.

Marketing may have also determined that it's important to have a familiar look and feel so the ultimate end user can use the product as intuitively as possible.

Therefore, you may also be considering a GUI that exhibits desktop characteristics. Pull-down menus and task-execution and -termination buttonsare the most recognizable features employed in interactive interfaces between humans and machines.

After marketing communicates the system's human-interface requirements, you have to select the right combination of OS and application software that most closely matches product specification.

You've already chosen the PC architecture because it has so much going for it in terms of user familiarity. The problem now: choosing from its abundant riches.

There are many excellent alternatives. Faced with the challenge of picking the "right" solution, you'll tend to opt for the most familiar. After all, less learning curve means shorter development time.

Although this possibility is very tempting, I'd like to suggest that you ask the providers of the various options out there.

Although none of us like to be sold on something we're unfamiliarwith, get all the facts before making a decision. A few phone calls or E-mail messages will give you a feel for the kind of support and responsiveness you can expect. Remember, a quick response from a helping hand can save the day when an important customer demo is just hours away.

Whether you need a real-time OS, an embedded DOS, flash file-management software, windowing software, debugging tools, or embedded kernels, the software sponsors of *INK*'s Embedded **PC** Design Contest offer a wide range of products and the support necessary to get products to market quickly. Taking advantage of their assistance is simply common sense.

So, you've compared the options and selected the right software for the project. Now comes the hardware.

HARDWARE CHOICES

Hardwaredesign methodologyfallsinto four principal categories depending on the size of the marketand stage in the product's life cycle.

Figure 1 illustrates typical product life cycles when the four most common design methodologies are used. A product intended for very high-volume production may actually progress through all of the architectures on its way to mass production.

However, time-to-market pressures often shorten total product market life to the point where it no longer makes sense to undertake wholly proprietary designs. Therefore, more in-house designs are only partially based on discrete components that reflect the unique value-added provided by the OEM.

Selecting the appropriate architecture depends in great measure on the ultimate market for which the product is intended.

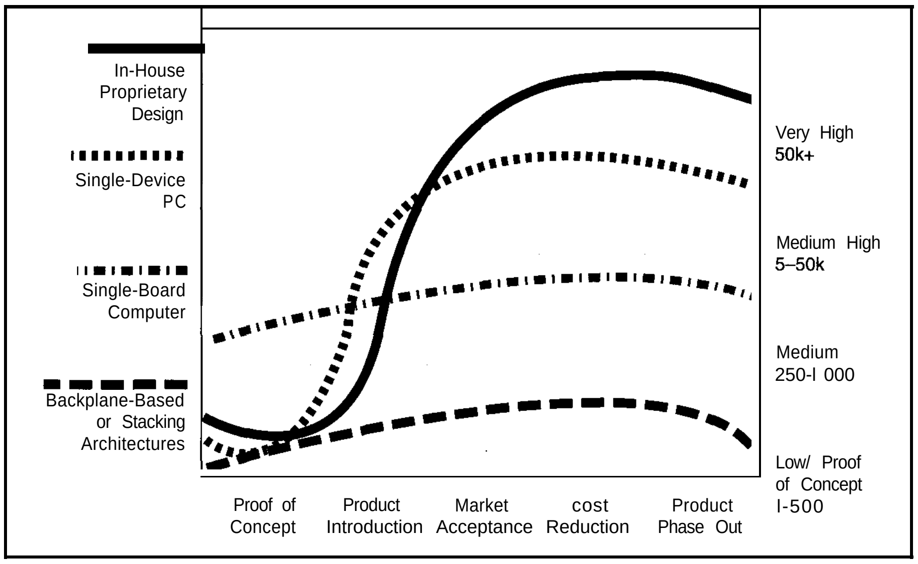


Figure I-Selection of one of the four most common design methodologies depends largely on the product's expected production volume and life cycle.

IN-HOUSE/DISCRETE COMPONENTS

This choice is usually best for very high-volume where 50,000+ systems per year will be produced. Here, the cost of goods is likely to be the greatest concern.

However, this approach often requires the highest front-end costs. The architecture must be determined, components selected, and prototypes built-all before the hardware and software can be integrated, a BIOS licensed and adapted, and an operating system ported.

The extremely short product life cycles inherent in the desktop mean that before selecting your 'x86-compatible components, ensure that those components will still be available when production begins.

A discrete component design often begins with a PC-on-a-chip device (e.g., the Elan 300 series from AMD). These devices must then be integrated with the other components to complete the design and provide the system's I/O and other functions.

A design based on one of these devices often includes as many as 50-I 00 additional components. Obviously, the effort to qualify, order, track, and inventory this many components can only be justified by extremely high-volume production.

Also, only these high-production volumes afford some measure of insurance against the risk of any single component in a design reaching its end of life and causing a complete system redesign.

BACKPLANE-BASED AND STACKING

Stackablecomponents (e.g., the PC/I 04 modules shown in Photo 1) help designs get to market quickly to prove market viability and gain a foothold ahead of competitors. If the design achieves the desired results and production volumes grow rapidly, a redesign is almost inevitable in order to reduce product costs and maintain or increase market share.

SINGLE-BOARD COMPUTERS

The extremely wide selection of single-board computers for embedded applications makes them ideal for products whose volumes are not expected to exceed 1 000–2000 units per year.

Typically, these boards represent half of the solution, as they only provide a PC-compatible "engine." A second proprietary design incorporates the unique features of the product.

From Backplane to SDPC-Based Design

A unique single-board embedded computer is the culmination of a progression through several of the design methodologies I discuss. The AKT/386 single-board transportation system controller shown in Photo i is part of a fare-box controller design developed by IBM Argentina's Systems and Solutions Group for use on public transportation systems.

The project began as an eight-board backplane-based design used to prove the concept and enter the market. The next iteration, a three-board PC/I 04 solution, reduced costs and increased reliability from the edge-card backplane design.

The decision to proceed with the final design was reached as prospects for higher volume developed. As well, field experience made it clear that even a ruggedized PC/I 04-based solution wasn't as reliable as a single-board design would be. Eliminating board interconnects and interboard cabling increased reliability while substantially reducing MTTR in the field.

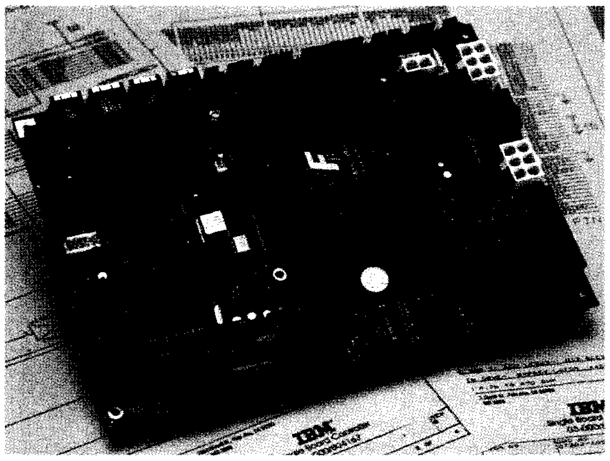


photo i—The AKT/386 CPU board developed by IBM Argentina and ZF Microsystems targets ***transportation*** applications. A single-device ***PC eliminates many cables and board interconnects-a plus in an environment where vibration*** is a ***major*** issue.

Generally, these boards are best suited to applications where cost is less of an issue than performance. As a result, most of the SBCs introduced currently tend to be premium priced, very high-performance Pentium- or '486-based designs like the 486DX4-based SBC in Photo 2.

SINGLE-DEVICE PCS

These component-h ke devices from Oki, S-MOS, or ZF Microsystems (e.g., the SMX/386 module shown in the sidebar and described in *INK* 69) let **you** include PC/AT motherboard functions in proprietary designs with minimal effort. They are usually best-suited to higher volume applications (e.g., 1000–50,000 systems per year).

Here, the development cost amortization is less significant, and the purchasing power is reasonably high. Therefore, these solutions must be highly cost competitive with in-house designs.

Single-device PCs have the advantage that project development can be done directly on the target processor that will be used in the final product. This feature enables shorter development cycles and faster product introductions with almost no redesign for high-volume production (seesidebar "From Backplane to SDPC-Based Design").

Single-board computersand backplane or stacking architectures (e.g., PC/l 04) lets **lower** volume products go from concept to product introduction without redesign. These architectures are most appropriate for production volumes of 250-l 000 units per year and for proof of concept.

Their primary disadvantages appear when market acceptance drives volume up and the added cost of interconnects between the SBC or stacking module and the proprietary technology makes cost reduction difficult without a system redesign.

Stacking modules can, ho&ever, be ideal peripherals for adding plug-in options to any of the architectures.

GOOD LUCK!

Successful companies lead their markets by getting products out before their competitors. They focus on their core competency and manage their resources effectively, thereby controlling their costs.

The brightest ideas are useless unless they get to market first. Few remember the second person to fly solo across the Atlantic.

Good luck with your entry in the **Embedded PC Design Contest!** EPC

David 1. Feldman is president and CEO of ZF MicroSystems in Palo Alto, California. The founder and former chief executive of Ampro Computers and the creator of the PC/104 concept, David has more than 25 years' experience in business management and the embedded systems market. You may reach him at dfeldman@zfmicro. corn.

SOURCES

PC/104 CPU, 1 00-MHz 486DX4, SMX/386, AKT/386
ZF Microsystems
1052 Elwell Ct.
Palo Alto, CA 94303
(415) 965-3800
Fax: (415) 9654050
www.zfmicro.com

PC/104 LCD controller
Apollo Display Technologies
194-22 Morris Ave.
Holtsville, NY 1 1742
(5 16) 654-l 143
Fax: (5 16) 654-l 496

Embedded PC Design Contest
Circuit Cellar INK
4 Park St.
Vernon, CT 06066
(860) 875-2 199
Fax: (860) 87 l-04 1 1
www.circuitcellar.com/public/cc/epcrules.html

I R S

4 13 Very Useful
414 Moderately Useful
4 15 Not Useful

Are you still wasting your money on

Run-Time Royalties?

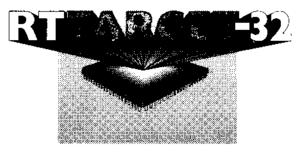
More than 4000 On Time customers aren't!

If you feel you've paid enough, check out our development tools for x86 embedded systems!

Windows NT compatible API!

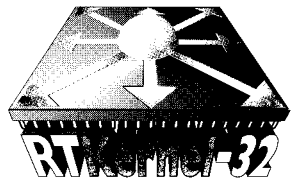
RTTarget-32

Cross development system
for 32-bit embedded systems.
Boot code, locator, cross debugger.
Supports Intel 386 and **higher, as little as 16k** P&I/ROM.
For Borland C/C++, Microsoft C/C++, Watcom C/C++, and Delphi.
Libraries: **$1** 700 Source Code: add **$1000**



RTKernel-3 2

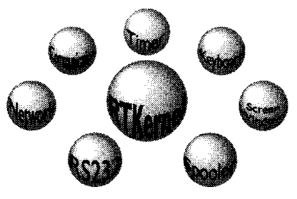
Professional, high-performance real-time multitasking kernel for RTTarget-32.
Supports Intel **386 and** higher.
for Borland C/C++, **Microsoft** C/C++, and **Watcom** C/C++.
Libraries: **$1** 950 Source Code: add $1650



Runs under DOS and Paradigm Tools!

RTKernel

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



Coming soon: TCP/IP stack and FAT file system!

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

International:
On Time Informatik GmbH
Hofweg 49
22085 Hamburg, GERMANY
Phone +49-40-2279405
Fax +49-40-2279263
email 102212.3101@compuserve.com

http://www.on-time.com

On Time
REAL-TIME AND SYSTEM SOFTWARE

Mike Justice
& Phil Marshall

Industrial I/O Networks

Although the desktop world essentially adheres to one network standard, the variation in industrial applications has led to the development *of* several networks. Mike and Phil suggest how to determine which standard you need.

As with any network, an industrial network is a common communications link between two or more devices. But, due to the unique requirements of manufacturing, separate networking technologies have developed.

These industrial networks-generally called "fieldbuses''-connect PCs, PLCs, industrial I/O, operator interfaces, drives, or any device needing to communicate with other pieces of a control system.

Industrial fieldbuses share several characteristics that distinguish them from their office relatives (e.g., Ethernet). For one, they are deterministic in nature, thus providing predictable performance.

And, while fieldbuses aren't as fast as office networks, data rate is probably an overused criteria for networking. After all, the message profile is different in the two applications.

Industrial networks also tend to be more robust (e.g., coax or fiber cable media) since they must provide noise immunity and dependability in harsh electrical environments. As well, the media used in the factory has different packaging and support hardware (e.g., hubs and connectors).

OFFICE VS. FACTORY

In **office** automation, the goal usually is to move as much data as possible between two points on an occasional basis. The transactions are often much larger than what's commonly found on an industrial network.

In industrial applications, you control I/O and continuously require the status of the I/O connected to, say, a PLC. While 1000 I/O points can be transferred in a single message, usually only a few bytes of data are required per transaction.

As well, you need the data continuously and in a predictable time frame. A conveyor application, for example, might be on every token rotation of the network to maintain synchronization, drive speed, and torque.

The cost per node differs significantly as well.

WHY SO MANY?

Given the myriad needs in manufacturing environments, there's a proliferation of industrial networkson themarket (see Table 1). tow cost and high speed with a deterministic architecture are only two requirements of networks for local I/O devices (e.g., DeviceNet and SDS).

Fieldbus and LonWorks are also designed for low-cost device-level implemen-

Network	Technology Developer	Year	Standard Organization
DeviceNet	Allen Bradley	1994	ISO 11898, 11519
InterBus-S	Phoenix Contact	1984	DIN 19258
LonWorks	Echolon	1991	ASHRAE / BACnet
IEC/ISA SP50	ISA/Fieldbus Foundation	1992	IEC 1158 / ANSI 850
Profibus			
SDS	Siemens/PITO Honeywell	1994 1994	IEC/ISO DIN 19245 11989

Table **1—As** you see, most network schemes were developed by different corporations and are maintained by different standard organizations.

Application	DeviceNet	InterBus-S	LonWorks	SP50	Profibus	SDS
Packaging	X	X			X	X
Conveyor	X	X			X	X
Building Control			X	X		
Process Control			X	X	X	

***2—Inevitably,** each network type meets the particular needs of a niche.*

tafions.
were originally designed for, greater distances are required, so t h esy crifice speed.

Profibus and ControlNet are for a higher level of communications (e.g., controller-to-controller communications). They let the

of manufacturing process, product quality, and plant productivity.

For long-term flexibility, use an open network. Thisstandard providescompatibility so that you can select vendors based on cost, availability, and functionality [1].

Some network protocols meet the specific needs of a particular application (e.g., Sercos for drive applications). Others are designed for use in single point I/O (e.g., DeviceNet and SDS).

Obviously, the best fieldbus for one application is not best for another. In applications like packaging machinery, performance is key. Process control, on the other hand, cares about distance and cable length.

CHARACTERISTICS

Someofthe most importantdetailsabout an industrial network are its physical characteristics, speed, distance, cabling, and communication methods (see Table 3).

Frequently, the need for speed is balanced by distance issues. Maximum distance usually causes minimum speed.

Twisted pair is the lowest-cost network solution as well as the easiest to install and maintain. Its disadvantage is its lack of electrical noise immunity.

Fiber-optic cable is good for distance, conductivity, and electrical noise immunity. But, it must be cut to fixed lengths and uses specialconnectors, complicating installation.

Radio is used for remote areas, where wiring and distance is a major problem. installing radiocommunications is not difficult, but cost, speed, two-way communications, and radio interference complicate the process.

The two communication methods are multimaster and master/slave. In a multimaster network, you need to know how arbitration specifies priority and performance.

The number of online intelligent devices (e.g., PCs) can also be a factor. A multimaster network lets each master communicate with other masters and slave devices without working through a single device.

Although most networks are still mostly single master and multiple slaves, the software enabling multimaster capabilities is evolving to truly distributed systems.

APPLICATION NEEDS

Network design balances several fundamentally opposing issues-speed, distance, and cost. As speed and distance increase, so does design sophistication, resulting in higher cost. If low cost is the goal, in restricting **distance** and speed, you restrict the applications it works for.

In a typical packaging application, where a local machine puts a product into a box or package, the machine is in a small geographic area and has a large number of DIO points. While distance is not an issue, performance and cost of the many I/O points are important. DeviceNet,

Network	Speed (bps)	Distance (m)	Cabling	Comm. Type	Max. Devices
DeviceNet	125k–500k	500	TP, signal & power	MM, M/S	64 nodes
InterBus-S	500k–12.8k	400/segment	TP, fiber	M/S	256 nodes
LonWorks	up to 1.25M	2000	TP, fiber, powerline	MM	32k/domain
IEC/ISA SP50	31.25k–2.5M	1900	TP, fiber, radio	MM	128 nodes
Profibus	31.25k–12M	24k	TP, fiber	MM	127 nodes
SDS	125k–500k	500	TP/signal & power	MM	64 nodes

Table *3-Sometimes, you can determine which network is best for your application **merely** by looking at the specs. (**TP** stands for twisted* pair, while MM and M/S refer ta multimaster and master/slave, *respectively.)*

Table 4—*Seldom can you have your cake and eat it, too.* Each gain comes with a specific tradeoff.

Speed	:	Distance
Scan time	:	Cable speed
I/O density	:	Distance
Number of nodes	:	Density of system
Determinism	:	Predictable message speed

InterBus-S, Profibus, and SDS qualify as good choices since they provide I/O with good performance.

In a conveyor application, many DIO points are spread over a large area. Reliability of data packets, networking distance, and speed are most critical. Again, DeviceNet, InterBus-S, Profibus, and SDS are good choices.

In building automation, dataaboutdoors, windows, heating, and air conditioning creates a profile for energy and security control. The I/O requirements are digital and analog with single pointsover a widearea.

Performance isn't usually an issue with building-automation systems. A 5- or 1 O-s scan time is considered fast. LonWorks and SP50 are well-suited for building-automotion networks, since theysupporta huge number of I/O over very large distances.

Process-control applications vary widely-from large oil refineries to small chemical processes. In general, they use a lot of analog inputs and outputs and only a few digital I/O. Because it's mostly analog, the price is high. Data and network reliability is extremely important.

Performance of the scanning of I/O can be an issue, but most systems have only a few points requiring fast scans. LonWorks, SP50, and Profibus PA are preferred because they can handle a lot of analog I/O spread over large areas.

While it would be great to quantify the selection of a fieldbus, the process is closely related to the type of application. One application's need often has an opposing tradeoff (see Table 4).

For example, if you need high speed and high I/O density, your system is probably confined to a small area. If you have a large number of nodes, then system density is probably low and spread out over a large area.

MACHINE-CONTROL EXAMPLE

The real-time control-systems division at Advanced Technology and Research (ATR) was contracted by General Motors to provide an open-architecture controller to support work with NCMS on the Next Generation Inspection System Project (NGIS II).

This controller serves a test bed for developing machine monitoring and on-machine measurement techniques to improve the accuracy of milling operations.

ATR was to retrofita K&T 800 horizontal-axis milling machine in the GM Powertrain engine-prototype lab with an RCS2000 open-architecture CNC controller. Probes, machine monitors, and sensors interface to the controller through open APIs.

Profibus was chosen for several reasons. In addition to gaining international support, the speed and determinism of Profibus DP (12 Mb) was well within GM's requirements. A milling machine must scan many digital and analog I/O points quickly.

The bus needed to be **fast,** deterministic, and either open or a standard. The scan time for reading and writing data had to be in the millisecond range and be predictable. Profibus DP proved to be the best fit.

So far, NGIS II has been a success. ATR is presently testing the machine with Sercos drives and an RCS2000 controller.

LOGICAL

Versatility in device Programmers DOS, Windows 3.1 Windows 95 Windows NT Compatible

Start programming devices today with the lowest cost and highest performance CERTIFIED programmers. Enjoy a no hassle user interface for ALL versions of Windows and DOS. Works with any PC of any speed without a hitch. Device libraries added in less 2 hours to our Web customer support section. Unique programming head options for gang programming most microcontrollers and memory devices. Direct Docking Gang TSOP, QPF, PLCC, DIP... programming heads. Evaluate a unit today with 100% satisfaction guaranteed or YOUR MONEY BACK!

(no penalties or restocking fees if unit is returned).

Gang any Device, in any Socket Type

with PC based or Stand Alone unit

Call Today in USA **800-331-7766**
Fax: 303-733-6868 or Visit our Home Page:

www.logicaldevices.com

#120

COMMON INTERFACES

Standardizing the interface to the software and hardware enables users to select the industrial network that best fits the application, preserving the investment in software and core hardware.

Using the PC/ISA or PC/ 104 computer bus as an example, we developed a set of PC/ISA and PC/ 104 boards that share the same hardware and software interface with the user. Each industrial network is implemented on the coprocessor board with the network connection and all the network-specific software and firmware running on the adapter board.

The interface to the user is via dual-port memory and a selectable interrupt. The memory structure is the same for each adapter.

For example, the first 1 KB is used for an image of the inputs and outputs. The second 1 KB handlescommunication messages to and from the board. The third area holds a set of simple command and status bytes that provide commands and information about the adapter.

By providing the same interface to each industrial network and a set of standard drivers for DOS, QNX, and Windows 3.1 1, 95, and NT, the user can **easily** write one application and use it with different networks. The industrial network then becomes just another simple component of the application-not a major problem.

Manufacturers are faced with the challenge of supporting various networks while minimizing costs. The PC/l 04 architecture solves this problem by providing a mechanicallyrobustand stabledesign platform while leveraging a large existing design base of products.

Since the PC/104 and ISA bus are fundamentally electrically identical, we could port all our ISA-based products to the PC/l 04 system while maintaining compatibility with the existing base of developed drivers.

NOT "ONE FOR ALL"

Frequently, we're asked which network is best or which network is going to win the war.

Our answer: many different networks will survive, given the sheer number of applications. No single network can address all the needs of all applications.

So, use the information you can glean and select an open-architecture network that enables your application to use the best attributes of that network. PCQ.EPC

Mike Justice is *president of Synergetic Micro* Systems, a company specializing in providing industrial *network* adapters for PC, PC/*104,* and custom OEM products. He is the executive director of the *InterBus-S* Club and has been involved with industrial communications for 18 years. You may reach Mike *at mjustice@synergetic.com.*

Phil Marshall is *sales* manager at SMSI. He has marketed and sold industrial *communications* products for the last IO years.

REFERENCE

[1] General Motors, Ford, and Chrysler, OMAC Paper, www. lpelcktronic.com/omac.htm.

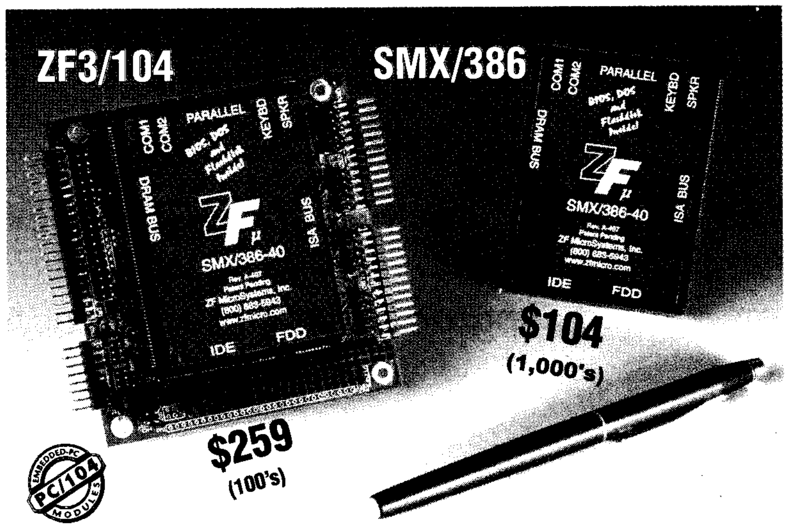
SOURCE

Network **interface** products
Synergetic Micro Systems, Inc.
2506 Wisconsin Ave.
Downers Grove, IL 605 15
(630) 434-l 770
Fax: (630) 434-l 987
www.synergetic.com

IRS

4 16 Very Useful
417 Moderately Useful
418 Not Useful

IF YOU'RE EMBEDDING A PC, TAKE A LOOK AT THESE



SMX/386 OEModule™

No, it's not a very large chip, it's a very small computer. A complete, 40MHz 386SX computer in a 240-pin surface-mount package just 2.20"x3.00"x.35" that mounts on your board like a chip.

Harsh Environment?

Forget unreliable cables and vibration problems. Everything you need is built into the OEModule. It boots from its internal Flash disk and even has 256K of user available program space. There are two COM ports, a parallel port, full core logic, keyboard & speaker ports, IDE and floppy disk support, and 2Mb of internal DRAM with support for up to 64Mb externally. Only 5V at 2W (power management & sleep modes available). Mounted directly on your proprietary board, connections are rock-solid even in harsh conditions.

A familiar platform.

The SMX/386 is PC/AT compatible, so all your standard development tools will work perfectly.

BIOS & DOS license included.

No licenses to negotiate. Power on the SMX/386 and you have a DOS prompt.

ZF3/104 CPUModule™

The creator of the PC/104 concept brings you more features per dollar than any PC/104 CPU on the market. 386SX, up to 18Mb DRAM &16Mb Flash, floppy, IDE, 4 serial, 2 parallel ports!

SBX/386 SlotCard™

The SBX/386 development kit comes with everything you need to put your project on the fast track to market: hardware, software, schematic and component libraries. It includes both AT and PC/104 expansion busses.

AKT/386 SystemCard™

Jointly developed by IBM @and and ZF for the transportation industry. Only 5.75" x 8", SMX/386, 8 serial ports, 2 parallel ports, J1708 interface, slave processor controls RS485 multidrop network, and a resident interface for GPS.

The Solid Solution in Embedded Control℠

ZFMicroSystems, Inc.
1052 Elwell Court
Palo Alto, CA 94303
Tel: l-800-683-5943
Fax: 415-965-4050
www.zfmicro.com

#121

Fred Eady

Internet Appliance Development

Part 1: From the GUI Up

And you thought you were unique? Nah! Even the embedded world wants to get on the Internet. Fred shows us some new tools that make it easier for companies to upgrade their clients' products remotely via the Internet.

I feel like Buckwheat in the *Little Rascals* movie when he chased down the duck and got a dollar bill tied to its leg. He was singing and dancing, "I've got a dollar, hey, hey, hey. Got a dollar today, hey, hey, hey."

Getting further into the duck story, Butch (the bully of the Rascals) had a problem. He wanted to steal the Heman Womun Hater's Club car for the big race, but Porky and Buckwheat were guarding it.

Butch's motto was, "Nothin' beats a duck and a buck." He tied a dollar to a duck's leg and got Porky and Buckwheat to chase the dollar-bill-totin' duck. Butch and his sidekick, Worm, got away with the car.

At first glance, *I'd* probably chase a duck tied to this eval kit.

OUTTA-THE-BOX INTERNET

The EXPLR2 Internet Appliance Developer's Toolkit is designed to aid theembedded design engineer in quickly getting an Internet Appliance to market. It has everything you need to get an Internet Appliance development system operational in minutes.

After reading the 0.634" of start-up documentation, I was up with a working demo in about 10 minutes. As Uh-Huh, the Rascals' single-word-speaking secretary would say, "Uh-huh."

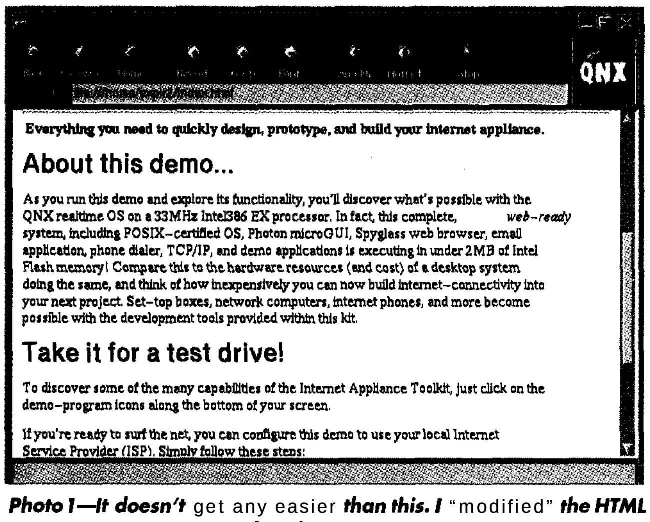
Just 10 minutes. Everything needed to run the initial demo is preloaded in the EXPLR2's flash.

You get a couple QNX Ethernet cards and a Rockwell 33.6 AccelerATor Modem kit with the EXPLR2 kit, too. And, all thecablesand what-nots are in the box.

I followed the instructions for setting the Ethernet card and modem jumpers and plugged the two cards into the open ISA slots on the EXPLR2 board. I added a keyboard, mouse, display, and standard PC power supply.

I fire it up, and blap! I'm looking at Photo 1. A phone line and some ISP (Internet Service Provider) info later, I was on the 'Net.

I'm jumping around in the kitchen, singing like Buckwheat, and you're shaking your head, "Fred got this neat embedded sys-



Everything you need to quickly design, prototype, and build your internet appliance.

About this demo...

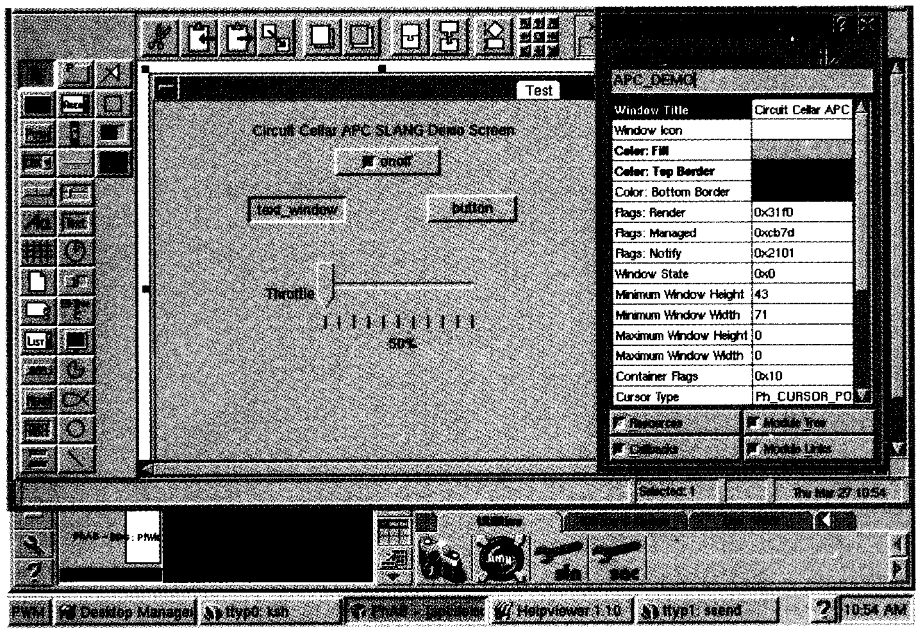
As you run this demo and explore its functionality, you'll discover what's possible with the QNX realtime OS on a 33MHz Intel386 EX processor. In fact, this complete, *web-ready* system, including POSIX-certified OS, Photon microGUI, Spyglass web browser, email application, phone dialer, TCP/IP, and demo applications is executing in under 2MB of Intel Flash memory! Compare this to the hardware resources (and cost) of a desktop system doing the same, and think of how inexpensively you can now build internet-connectivity into your next project. Set-top boxes, network computers, internet phones, and more become possible with the development tools provided within this kit.

Take it for a test drive!

To discover some of the many capabilities of the Internet Appliance Toolkit, just click on the demo-program icons along the bottom of your screen.

If you're ready to surf the net, you can configure this demo to use your local Internet Service Provider (ISP). Simply follow these steps:

Photo 1—It doesn't get any easier ***than this. I*** "modified" ***the HTML*** source ***to*** add an INK ***advertisement to the top of the*** window.



APC_DEMO

Window Title	Circuit Cellar APC
Window Icon	
Color: Fill	
Color: Top Border	
Color: Bottom Border	
Flags: Render	0x31f0
Flags: Managed	0xcb7d
Flags: Notify	0x2101
Window State	0x0
Minimum Window Height	43
Minimum Window Width	71
Maximum Window Height	0
Maximum Window Width	0
Container Flags	0x10
Cursor Type	Ph_CURSOR_PO

Photo 2-Who cares about making a real application. I could spend days here turning all those knobs1 Notice ***that the** APC_DEMO I* created is an instance of the Photon class *PtWindow.*

tern in the mail, and now he thinks he can get on the Internet with a toaster oven. I bet he braided his hair like Buckwheat, too."

Otay. Otay. I may be a singing, kitchen-dancing, mad-duck-chaser with braids, but I know what an "Internet Appliance" is.

Do you? Well, you know what the Internet is, and you know what an appliance is. That's an Internet Appliance. It's your version of a toaster or dishwasher for the Internet-a specialized piece of embedded hardware and firmware you design to perform some particulartaskvia the Internet.

A 455 ROCKET

If you don't remember the Oldsmobile 455 Rocket V-8, you're youngggggg. In its time, it was the biggest displacement engine **you** could buy off the showroom floor. It was built for speed.

Just like the 455 Rocket was built to go real fast, the EXPLR2 Internet Appliance system board was designed to be both flexible and functional. It's based on the 33-MHz C-Step Intel '386EX processor and surrounded by a full set of peripherals, compliments of a RadiSysR380EX embedded system controller.

The R380EX is specifically designed to provide glueless support for the '386EX. It's got most everything-a DRAM controller, keyboard/mouse controller, real-time clock, enhanced IDE interface, and an ISA-bus controller. Of course, all the goodies are PC compatible to keep us 'x86 embedded bit shifters happy.

To jazz things up, the Intel and RadiSys folks threw in functional onboard Port 80 POST LED circuitry and a PCMCIA interface.

When all is said and done, the hardware consists of:

- an EXPLR2 system board with mouse, keyboard, standard PC power supply, and monitor
- a host PC ('486 or higher) loaded with QNX and the normal complement of peripherals
- a QNX-based Ethernet link between the host and EXPLR2
- a 33.6 Rockwell chipset modem on the EXPLR2 for Internet access

BODY BY FISCHER

All the power that 455 Rocket could muster wouldn't push it along the highway withoutsomesupportingcomponents (e.g., tires, a crankshaft, a frame, and a sturdy body). Remember, Butch stole the whole car, not just the engine.

I could power up the toolkit system board, drop it onto some hot embedded asphalt, and obtain minimal computational results. There's nothing special about the hardware functionality.

Or, I could add QNX's Photon MicroGUI and a "just-in-time-compiled-programon-the fly" programming language from Cogent Real-Time Systems.

In case you're not mechanically inclined, let me put this Photon/SLANG thing in perspective. You just designed and marketed a unique Internet Appliance. Some fancy marketing engineer named all the buttons.

Way down in the menu structure, a selection-button label is misspelled. "Appropo" is something you thought you ate with cornbread. Well, it **should** spell "Apropos," and you can't order it at the diner.

You have more of these devices with the misspelled label out there than you want to think about. And, you can't tell your clients to stop using their products for a couple hours while you download the fixes. Most of them rely on the appliance you parented to keep their businesses profitable.

And, by the way, your little invention was sold all over the world. That means different time zones. Someone somewhere is using it all the time! You're hosed! (Where did you put that resume?)

But, you created your GUI sibling with Photon and SLANG.

PHABULOUS PHOTON MICROGUI

Photon is a graphical user interface environment for the QNX RTOS and QNX-based applications. It's similar to any other GUI you may have encountered in your development travels.

Just like Bill's Visual stuff in the DOS world, Photon is an event-driven environment. User input via mouse or keyboard is processed as an event, and a corresponding output is produced.

Bill's GUIs tend to be largish because they're not intended for a classical embedded environment. Photon is inherently tiny but can be scaled to any proportion.

Thus, it can be used in embedded and not-so-embedded applications. It doesn't matter if you're running Photon on a stingy embedded design or a full-blown PC-the look, feel, and results remain the same.

Photon has the added advantage of being microkernel based, enabling it to work easily in network-intensive applications. Right there beside QNX.

GUIs aren't grown on trees and don't come preapproved in the mail. Somebody and something must be employed to make a pleasurable interface.

In the QNX/Photon world, that development tool is PhAB (Photon Application Builder). PhAB assembles and generates the GUI objects the user manipulates as well as the application interface C code that makes the interface come alive.

80251 Embedded **Controllers**

MMT-188EB
MMT-Z180
MMT-HC11
MMT-196
MMT-31
MMT-251
MMT-PIC44
& OTHERS

MMT-251
$199.00 (100 Quantity)

Midwest Micro-Tek is proud to offer its newest line of controllers based on the 8031/51/251 architecture. The 8031 comes in at a surprisingly low Cost of $89.00 (100 quantity). Custom work welcome.

MIDWEST MICRO-TEK
2308 East Sixth Street
Brookings, SD 57006
Phone 605.697.8521
Fax 605.692.5112
www.midwestmicro-tek.com
Guaranteed to be in your Future

#122

AMX® The Real-Time Multitasking Kernel

680x0, 683xx	PowerPC™ family
80386 protected mode	i960® family
80x86/88 real mode	R3000, LR33xxx

NEW: • PowerPC support • *KwikLook* AMX-aware debug tool

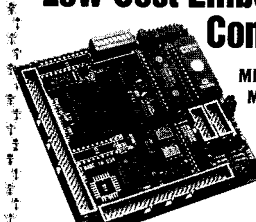
- Compact, ROMable, fast interrupt response
- Preemptive, priority based task scheduler
- Mailbox, semaphore, resource, event, list, buffer and memory managers
- Configuration Builder utility
- Comprehensive documentation
- No royalties, source code included

For a *sample of KwikLook and description of* AMX,
Phone: (604) 734-2796
Fax: *(604) 734-8114*
E-mail: *amxsales@kadak.com*
Web: *http://www.kadak.com*

KADAK Products Ltd.
206-1847 West Broadway
Vancouver, BC, Canada V6J 1Y5

#123

Low Cost Embedded Controllers



MMT-188EB
MMT-Z180
MMT-HC11
MMT-196
MMT-31
MMT-251
MMT-PIC44
& OTHERS

MMT-188EB
$199.00 (100 Quantity)

If you're interested in getting the most out of your project, put the most into it. Call or Fax us for complete data sheets and CPU options. Custom work welcome.

MIDWEST MICRO-TEK
2308 East Sixth Street
Brookings, SD 57006
Phone 605.697.8521
Fax 605.692.5112
www.midwestmicro.tek.com
Guaranteed to be in your Future

#124

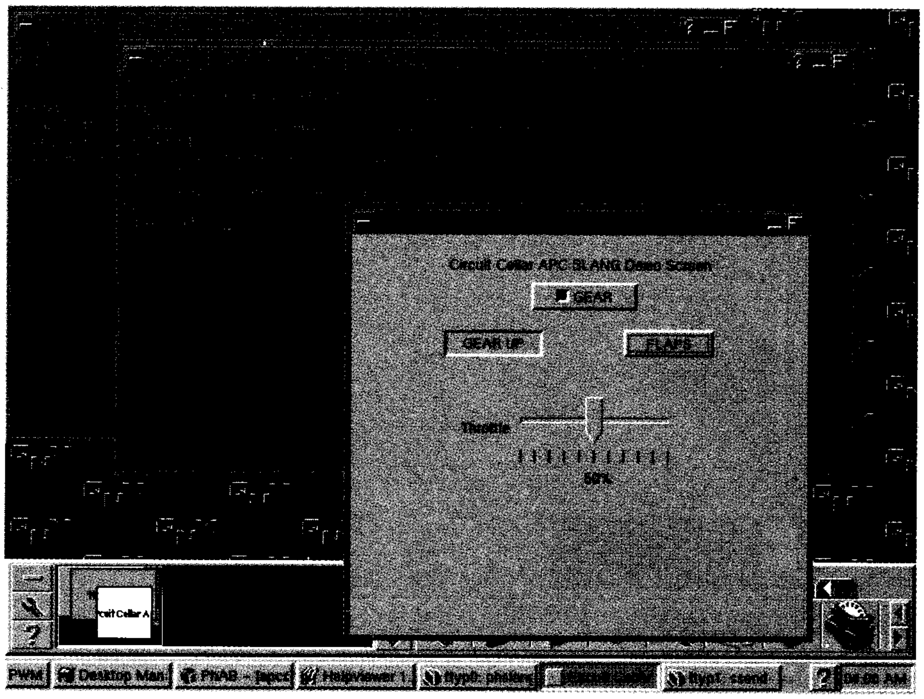


Photo ***3—Sorry,*** I didn't have to write any complicated code to *make any* of this happen.

GUIs consist of windows, menus, dialogs, icons, buttons, labels, and supporting user-written or machine-generated code. PhAB handles the design and creation of all these ingredients.

Its strength lies in the fact that you the developer can focus on the problem at hand, not the code. Buttons and labels are encapsulated into what Photon calls "widgets."

Using PhAB, widgets are created, moved, and processed by a simple mouse click. Each widget has a set of parameters or resources controlled within the PhAB framework atcreation. These resources can be varied on-the-fly in run time with SLANG.

In addition to the physical control of a widget, PhAB permits working code to be attached to a widget's event-generation properties. In other words, when a button within the GUI is selected, an event is triggered that invokes an application-specific call-back algorithm.

This callback is usually connected to a window, dialog, or menu. Most of the time, the callback is laced with user-written C code to perform a predetermined function.

The raw interface code is generated by PhAB, not the programmer. So, the interface can come alive without you writing any code. Think about it. You can test the GUI interface for look and feel before committing to any application code.

Let's face it. Most of the problems with creating GUI interfaces are called "end users." I don't care how pretty or functional the programmer thinks the GUI is, end users always have the final say. With PhAB, you can create a GUI for them while **they** watch!

Once you nail down the interface, you can fine-tune the GUI with your own C verses. While Photon (and **every** other GUI, for that matter) was designed to work this way, PhAB has a secretweapon-SLANG.

SLANG takes the place of C in the PhAB environment. Normally, you design your GUI interface and use PhAB to generate the C back-bone code that makes the widgets work at their minimal level.

With SLANG, you design the interface without generating the basic PhAB C code. You then load the codeless PhAB-created widget image into the SLANG environment.

Using SLANG syntax that's structured to closely resemble native PhAB command syntax, you logically implement the call-back structures just like you would with C. The developers at Cogent tell me they only use C for the down-and-dirty driver stuff.

If you're not intimately familiar with LISP and C, SLANG is going to be a little difficult to understand at first. Once you're high on the learning curve, here's what SLANG buys you. **For every** 1 O-I 5 lines of C you'd write in C call-back code, you write on average l-2 lines of SLANG code.

Also by using SLANG, you can modify the GUI appearanceand functionality without (yes, without) stopping or interrupting the running application. So, you can change the GUI's look and feel in real time.

With SLANG, you can debug the application-GUI and all-in real time. Since QNX is network natured, you can even debug and modify SLANG applications remote to your local machine!

Thus, all those bugs on your nifty Internet Appliance can be squashed without interrupting the user or the program. Program updates can be applied on-the-fly, totally transparent to the user.

The best way to fully describe this miracle is to show you how it's done.

SLANGING CODE

Remember those couple of Ethernet cards? Well, the other network card goes into a host PC running the QNX demo system that comes with the toolkit.

The idea is to link the host PC with the EXPLR2 board via Ethernet and use the power of QNX IPC (Interprocess Communication) and SLANG to rapidly debug and deploy your application. Over this IPC link, you can control the application running on the EXPLR2 from the host PC and vice versa,

The EXPLR2 QNX software suite consists of a 30-day license for full-blown QNX 4.23, Watcom C 10.6, Photon, and QNX TCP/IP on CD. A 30-day, full-function version of SLANG comes on another diskette.

The set includes an abundance of working Photon and SLANG programming examples bundled with their complete (and commented, I might add) source code.

For guys like me that need to show this stuff via the printed page and scatter it about the Internet, there's a snap-shot program for screen captures as well as a graphic viewer among the many utilities on the CD.

One handy utility, Jump Gate, enables me to transfer programs between the host and EXPLR2 for execution. I can also import a screen from the EXPLR2 and manipulate the application on the host PC as if it were physically on the EXPLR2.

Photo 1 was generated by sending the host-based snap-shot utility to the EXPLR2 and executing it there against the Web screen. I sent the resulting screen capture back to the host PC via the Ethernet link and QNX IPC.

The QNX Ethernet connection along with IPC features within the QNX OS lets the EXPLR2 be loaded directly from a specific directory on the host PC (/home/iat/explr2/root). The EXPLR2 board sees the host PC's EXPLR2 directory as if it were its own. Conversely, the host PC sees into the flash file system on the EXPLR2 board!

The EXPLR2 can be made to boot over the network or from its onboard flash. Booting from the network lets the developer quickly assemble an application and test it by putting the required code modules in the PC EXPLR2 directory and starting the application on the EXPLR2 to bring them online. Changes are implemented by modifying the required module on the host PC and restarting the application.

When debugging is done, bring your application to life from flash by loading the appropriate PhAB, C, and SLANG modules into the EXPLR2 host directory and downloading the resultant file image to the EXPLR2 board's flash. All the utilities to perform these tasks are included with the EXPLR2 toolkit.

SLANG lets you instantly see the changes you make. Imagine skipping the compilation and reboot steps and updating your application in real time.

Yep, you can enter a line of code and see the result while your application is running! At that point, you can either incorporate it or trash it. That's the power of SLANG.

LEFT WIDGETS

I'll show you how it works by creating and manipulating some widgets in a window with PhAB. Photo 2 looks into the PhAB application I call **apcdemo.**

Before I can manipulate the widgets, I must know how they're defined. The definitions are established in the initial PhAB session. For clarity, each widget's name is displayed as its initial text attribute.

After placing all the widgets in the window, I generate the application using PhAB's Generate function. Generate builds some backbone C code and a file called APC_DEMO.wgtw, which is what I'll load into SLANG for manipulation.

Normally, using PhAB only, I would Make the application that generates all the code necessary to run the application. Listing 1 is the SLANG load procedure saved on the host PC as **slangcode.**

This code invokes the Circuit Cellar APC window. Typing slangcode in the ttyp0 window starts SLANG and sets up the environment for loading an application window.

The Photon environment allows up to nine ttypx windows. To do on-the-fly changes to the Circuit Cellar APC application, I open ttyp1 and issue ssend slangcode.

The program ssend attaches to a running SLANG program (e.g., slangcode) and enables the user to send commands without exiting the event loop of the attached process. When slangcode> appears in ttyp1, I'm connected to the target app and ready to issue change orders.

*listing 1—Look at that Load statement. I might be a **writer** by trade, but I know how to widget around in QNX.*

```
#!/usr/cogent/bin/phslang

/* Initiate graphics session with Photon window manager */
PtInit(nil);

/* Declare a global name */
qnx_name_attach (0,"slangcode");

/* Load Photon Widget convenience functions */
require-lisp("PhotonWidgets");

/* Load support for loading windows created in PhAB */
require-lisp("PhabTemplate");

/* Load and convert window created in PhAB using wload function.
   Set first item returned (PtWindow) definition to be variable
   named win. */
win = car(wload("/usr/itk/apcdemo/wgt/APC_DEMO.wgtw"));

/* Normally, user-written call-back functions reside here */
/* Start an infinite event loop to handle Photon events. I can also
   use a call to PtMainLoop() here, but then I can't intervene
   after each event. */
while (t){
  next-event();
}
```

RIGHT WIDGETS

The application consists of a base window and five widgets. Ci rcui t Cel 1 ar APC SLANG Demo Screen is a label widget. The onof f widget is an on/off push button.

Let's assume we need a landing-gear push button in an aircraft application. To label this button GEAR, I enter onoff.text _string = "GEAR": in ttypl.

What happens? Yep, it says GEAR.

This button has an indicator that changes color when the button is pushed. This operation is inherent to the widget, but I can use SLANG to pick the color if I wish.

Follow the commands through in Photo 3, and note the results in the application window. That's how SLANG works.

I can do all sorts of things to the widgets from t t y p l. In fact, any widget resource that can be program controlled can be changed with this method. All that's left is to write and insert SLANG call-back code.

STILL RACING ON

While the software is solid and the development hardware well thought out, it takes time to navigate through the kit's sea of documentation. An inexperienced QNX user will undoubtedly need to call for help.

I managed to hose my server's Internet mailbox because the Web demo didn't log off cleanly. One thing led to another, and I found I'd also corrupted the SLANG demo and snap-shot application while trying to fix the mailbox problem. It wasn't a pretty sight, and I ended up reloading the entire development package-more than once!

I suggest you read all the supporting documentation before diving into this kit. And, if you're not familiar with QNX, learn some QNX basics and talk to some experienced folks before you begin.

I found the technical support from all parties involved to be very good. Use them as resources. It will save you time and frustration.

Next time, I'll mix SLANG and Photon into some hardware and cook up a useful QNX/EXPLR2 application. APC.EPC

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

SOURCES

SLANG
Cogent Real-Time Systems
168 Queen St. S, Ste. 205
Mississauga, ON
Canada L5M 1 K8
(905) 8 12-9628
Fax: (5 10) 472-6958
info@cogent.ca

Photon **MicroGUI,** QNX RTOS
QNX
175 Terence Matthews Crescent
Kanata, ON
Canada K2M 1 W8
(613) 591.0931
Fax: (613) 591-3579
www.qnx.com

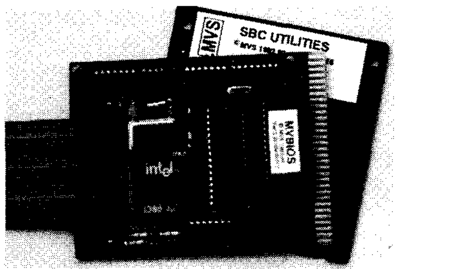
EXPLR2, EXPLR2 Internet Appliance Developer's Toolkit
RadiSys Corp.
15025 SW Koll Pkwy.
Beaverton, OR 97006.6056
(503) 646-I 800
Fax: (503) 646-I 850
BBS: (503) 646-8290
www.radisys.com

IRS

419 Very Useful
420 Moderately Useful
42 1 Not Useful

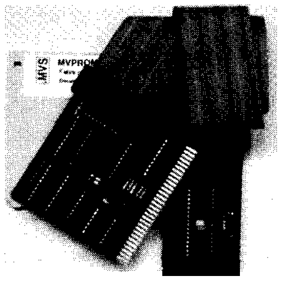
386 SBC $83 OEM (1K) PRICE INCLUDES:
- 5 SER (8250 USART)
- 3 PAR (32 BITS MAX)
- 32K RAM, EXP 64M
-STANDARD PC BUS
- LCD, KBD PORT
- BATT. BACK. RTC
- IRQO-15 (8259 X2)
- 0237 DMA 0253 TMR
-BUILT-IN LED DISP.
-UP TO 8 MEG ROM
-CMOS NVRAM

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

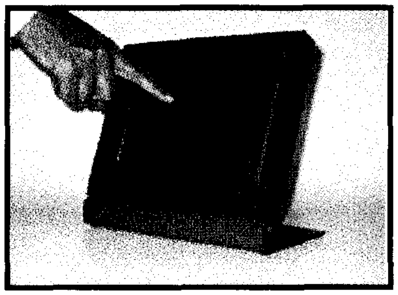
- EVAL KITS INCLUDE MANUAL BRACKET AND SOFTWARE.

5 YR LIMITED WARRANTY
FREE SHIPPING
HRS: MON-FRI 10AM-6PM EST

MVS
MVS BOX 850
MERRIMACK, NH
(508) 792 9507

#127

Touch The Future



LCD Touch Monitors
LCD Touch Screens
VGA LCD Displays
LCD Controllers
ISA, **PC104,** Analog, Video

EARTH Computer Technologies

lowest Prices on **Earth!**

27101 Aliso Creek Rd - # 154 - Aliso Viejo - CA - 92656
Ph: 714-448-9368 - Fax: 714-448-9316
Email: oemsales@flat-panel.com
FREE CATALOG available at http://www.flat-panel.com

#128

MICRO SERIES

Jan Axelson

Using Serial EEPROMs

Part 1: General Principles

Part 1 of 2

In need of nonvolatile, read/write memory? Perhaps, you should check out Jan's review of EEPROMs. She'll help you determine whether Microwire, SPI, or I^{2}C is the solution.

If your project needs a modest amount of nonvolatile. read/write memory, serial EEPROM may be the answer. These tiny, inexpensive devices are especially useful for minimizing the number of I/O lines, cost, or physical size.

Serial EEPROMs most commonly store user data-settings for remote-control devices, phone numbers, security codes, or anything that once was set with DIP switches. You can also store error codes, diagnostic information, usage records (e.g., times, dates, counts), and instrument readings.

In some cases, serial EEPROMs can even store program code. Parallax's BASIC Stamp and similar products use them to store user programs in the form of BASIC language tokens.

This series is a guide to choosing and using serial EEPROMs. In Part 1, I compare the three major interface types-Microwire, SPI, and I^{2}C. In Part 2, I'll show you how to program and read all three types from a PC standard parallel port.

THE BASICS

Serial EEPROMs use a synchronous interface. Both the EEPROM and the chip controlling it use a common clock, and clock transitions signal when to send and read each bit.

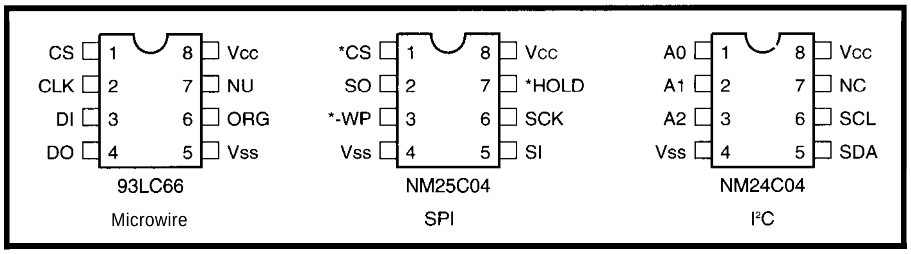


Figure I--Here are *pinouts for* three *512-byte* serial *EEPROMs. The SO/C version of the 93LC66 is available in this and an alternate pinout. On the NM24C03, pin* 7 is *Write* Protect.

Although synchronous serial chips require minimum clock frequencies, the clock for serial EEPROMs can be as slow as needed, and the clock signal doesn't need to be symmetrical. The controlling device can toggle the clock at its convenience, up to the maximum speed. There's no need for a fixed time-base.

Serial EEPROMs typically have just eight pins-power and ground, one or two data/address lines, and a clock input, plus up to three other control signals. Unlike parallel EEPROMs, which add pins as the number of address and data lines grows, a serial EEPROM's physical size doesn't have to increase with capacity.

Capacities begin at 128 bytes. As with other memory, maximum capacity has increased over time. Microchip's 24LC64, for example, is an 8-KB chip.

The EEPROMs use CMOS technology, so they consume very little power. Currents are as low as a few microamps in standby mode and a milliamp when active.

The synchronous interfaces aren't intended for use over long distances. For that, use RS-232 or RS-485. However, cables may be as long as 4 m in some cases-longer if you add stronger drivers and buffers.

Depending on the device, the maximum clock speed for accessing serial EEPROMs can be over 2 MHz. But because it takes eight clock cycles to transfer a byte and the master also has to send instructions and addresses, the maximum data-transfer rate is no more than -4 µs per byte.

Write operations take much longer because the EEPROM needs several milliseconds to program a byte into its memory array. During this time, the master can't read or write to the chip, but it can do other tasks not involving the EEPROM.

With use, EEPROMs eventually lose their ability to store data. So, they're not suited for applications where data changes constantly.

These days, most are rated for a minimum of 10 million erase/write cycles. That's fine for data that changes occasionally or even every few minutes. But, if you need unlimited read/write cycles, use battery-backed RAM.

Besides EEPROMs, other components with synchronous serial interfaces include ADCs and DACs, I/O expanders, clock/calendars, and display interfaces. Multiple devices connect to one set of lines, with each chip having its own Chip-Select line or firmware address.

There are three major types of interfaces for serial EEPROMs—Microwire, SPI, and I^{2}C. The different types vary in speed, number of signal lines, and other details.

To see how the different interfaces compare, I describe a 4-Kb EEPROM of each type. Table 1 summarizes the major features, and Figure 1 shows their pinouts.

All EEPROMs with the same interface behave in a similar way, though they may vary in the number of address bits and other details (e.g., whether there's a write-protect pin). Always read the datasheet for the chip you're using!

MICROWIRE

Microwire is the oldest of the three interfaces. National Semiconductor introduced it, and other manufacturers now support it as well.

National's COP888 is an example of a microcontroller with a Microwire interface built in. Though it's commonly called a three-wire interface, a complete link actually needs four signal lines plus a common ground.

Microchip's 93LC66 is a 4-Kb serial EEPROM with a Microwire interface. It has two data pins, DI (data in) and DO (data out), as well as a clock input (CLK) and a chip select (CS).

Additional inputs are for memory configuration (ORG), which determines data format as 8 or 16 bits, and program enable (PE), which must be high for programming. Setting ORG high saves time because you can program and read two bytes with one instruction.

The EEPROM understands seven instructions-Erase/Write Enable and Disable, Write, Read, Erase, Erase All (sets all bits to 1), and Write All (writes one byte to all locations).

Figure 2 shows the timing for byte read and write operations. Each instruction must begin with a Start condition, which occurs when CS and DI are both high on CLK's rising edge.

The master must bring CS low after each instruction except sequential reads. When CS is high, the EEPROM is in standby mode, ignoring all communications until it detects a new Start condition.

To write to the EEPROM, the master must first write an Erase/Write Enable instruction to DI, followed by a Write instruction, the address to write to (0–1FFh), and the byte or word to write.

Interface	Microwire	SPI	I^{2}C
Example device (512-byte cap.)	93LC66	NM25C04	NM24C04
Source	Microchip	National	National
Min. interface (excl. GND)	4	4	2
Data width (bits)	8 (2 16	8	8
Max. clock speed (MHz)	10	2.1	0.4
Write (busy) time (ms, max)		5	10
Max. bytes programmed in one operation	2	4	16
Writes bit on (clock state)	rising edge	rising edge	low level
Reads bit on (clock state)	rising edge	falling edge	low level
Output low current (min.)	2.1 mA at 0.4 V	1.6 mA at 0.4	3 mA at 0.4 V
Output high current (min.)	0.4 mA at 2.4 V	0.8 mA at Vcc-0.8 V	(open collector)
Chip-select method	hardware	hardware	software
Write-protect method	software	hardware & software	none*

Table I-A/though *Microwire, SPI, and PC are all synchronous serial interfaces, the hardware specifications and other aspects of each are unique. (*The'24C03 has hardware write protect for its upper half.)*

The master writes bits on CLK's falling edge, and the EEPROM latches each bit on the next rising edge.

After sending the final data bit in a programming operation, the master must bring CS low before the next rising edge of CLK. This action causes the EEPROM to begin its internal programming cycle.

Some devices, such as Microchip's 93C76, don't require CS to go low here. Instead, they begin programming on CLK's rising edge after DO.

The programming is self-timed, so it requires no clock cycles. If CS returns high before the programming cycle completes, DO indicates Ready/*Busy status. CS must then go low again to complete the write operation.

The master needs to send the Erase/Write Enable instruction just once per programming session. The device remains write enabled until it receives an Erase/Write Disable instruction or power is removed.

To read from the EEPROM, the master writes a Read instruction to DI, followed by the address to read. When the EEPROM receives the final address bit, it writes a dummy 0 to DO and then writes the requested data on CLK's rising edges.

If CS remains high after a Read operation, extra clock transitions cause the chip to continue to output data at sequential addresses. If CS goes low, the next read operation must begin with the Read instruction and an address.

A two-line interface is sometimes possible by connecting DO and DI. (An isolation resistor between the lines is recommended.)

However, there is a brief bus conflict in Read operations when the EEPROM outputs the dummy zero on receiving the final address bit, and the master's driver must be strong enough to pull the combined DO/DI line high when this occurs and the address bit is 1.

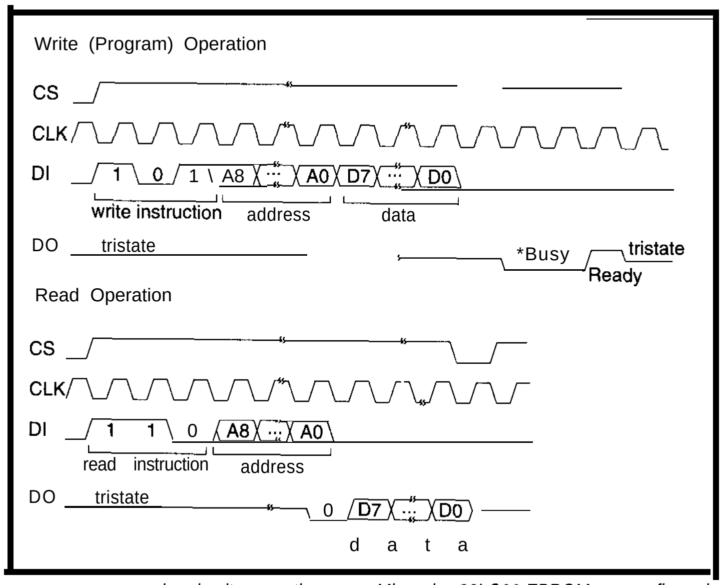


Figure 2—Byte read and write operations on a Microwire 93LC66 EPROM are configured for 8-bit organization. CLK's rising edges latch inputs at DI and clock data out at DO.

REMOTE POWER CARD!

3 VERSIONS:

RESET

WATCHDOG RESETS PC IF IT HANGS FOR HARDWARE OR SOFTWARE REASONS

P H O N E

TURN ON PC WITH PHONE, SHARE VOICE / MODEM LINE, CONTROL AC APPLIANCES

TIMER

WAKEUP OR SHUTDOWN PC, LATE NITE BACKUP / MODEM, CONTROL AC APPLIANCES

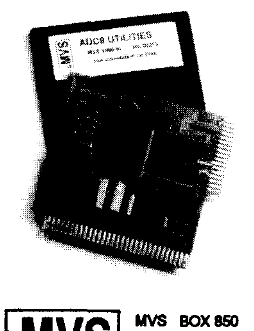
95$ 27$ OEM

ALL MVS CARDS INCLUDE ASM, BASIC, AND C SOURCE FOR PC OR SBC

8 CHAN ADC

DATA AQUISITION, 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) 792 9507

5 YEAR LIMITED WARRANTY

FREE S H I P P I N G IN USA

#129

oes your Big-Company marketing department come up with more ideas than the engineering department can cope with? Are you a sma ll 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.

hether 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.

CIARCIA DESIGN WORKS

FAX: (860) 871-8986

steve.ciarcia@circellar.com • www.circellar.com

SPI

The second interface type, SPI (Serial Peripheral Interface), originated at Motorola, and is included on their 68HC11 and other microcontrollers. SPI is much like Microwire, though the signal names, polarities, and other details vary.

Like Microwire, SPI is often dubbed a three-wire interface, although a read/write interface requires two data lines, a clock, a chip select, and a common ground.

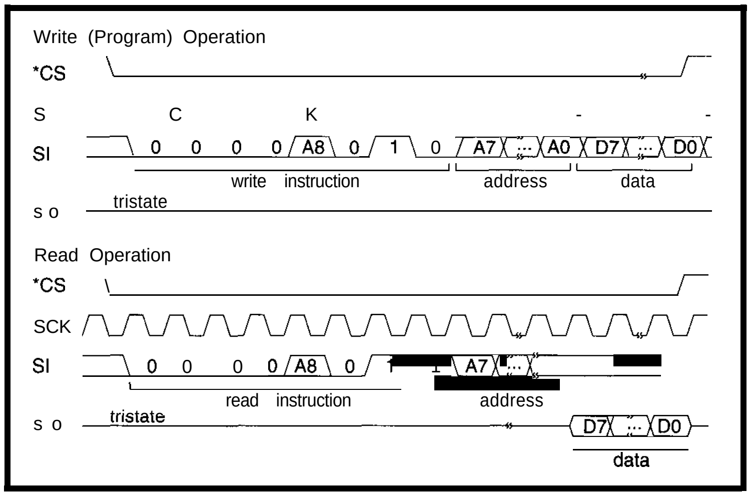


Figure 3—The NM25C04's SPI interface is similar to Microwire's. SCK's falling edges latch inputs at SI, while rising edges clock data out on SO.

The signal names differ on the master and slave devices. The data line MOSI (master out, slave in) on the master connects to SI on the slave, and MISO (master in, slave out) on the master connects to SO on the slave.

The clock is called SPICK at the master and SCK at the slave. A Master may also have four outputs (SS0–SS3), each connecting to a chip select (*CS) on up to four slave.

As with Microwire, SPI EEPROMs write bits on the clock's rising edge. But unlike Microwire, they latch input bits on the falling edge. (The SPI protocol allows two different clock polarities, but EEPROMs support only clock polarity equal to 0.)

Some SPI devices support two phases but the EEPROMs support only one clock phase equal to 1. Notice also that the polarity of *CS is opposite from Microwire's convention.

National's NM25C04 is a 4-Kb EEPROM with an SPI interface. In addition to the four lines mentioned above, the chip has two other inputs.

* WP must be high to program the device. For interfaces with multiple slaves, the *Hold input lets the master pause in the middle of a transfer to do something more urgent on the SPI link. The EEPROM ignores all activity on the SPI bus until *Hold returns high and both devices pick up where they left off.

The EEPROM understands six instructions-Set and Reset the Write Enable Latch, Read and Write to the Status Register, and Read and Write to the Memory Array.

The chip has several levels of write protection, which you can use to virtually guarantee there'll be no inadvertent writes to the device. If *WP is low, no changes to the data are allowed. If it's high, two nonvolatile bits in the chip's Status Register can block writes to all or a portion of the device.

If • WP is high, before you can write to the Status Register or the portion of memory enabled in the Status Register, the EEPROM must receive a Set Write Enable Latch instruction.

Figure 2 shows the timing for byte reads and writes for the '25C04. To write to the EEPROM, the master writes a Set Write Enable Latch instruction to SI, followed by a Write instruction (which contains address bit A8), the lower eight address bits, and the data to write.

The master may send up to four data bytes for sequential addresses in one operation. After clocking the final data bit, with SCK low, CS must go high to begin programming the byte into the EEPROM.

While the EEPROM writes the data, the master can read the EEPROM's Status register. When bit 0 of the Status Register is 0, the EEPROM has finished programming and the next write can begin. The chip is write protected after each programming operation, so each write must begin with a Set Write Enable Latch instruction.

To read the EEPROM, the master sends a Read instruction, which contains bit A8 of the address to read and then bits A7–A0. The EEPROM responds with the data bits in sequence on SO. As with Microwire, additional clocks cause the EEPROM to send additional data bytes in sequence.

For larger capacities, rather than embedding address bits in instructions, the master sends a 16-bit address.

I²C

The third interface type, I²C, originated with Philips. Their 8xC528 (8051 family) is an example of a microcontroller with built in I²C interface.

The I²C interface requires just two signal lines plus a common ground. Serial Data/Address (SDA) is a bidirectional line requiring open-collector or open-drain outputs. Serial Clock (SCL) is the clock. Instead of a chip-select line, the master sends a slave address on SDA.

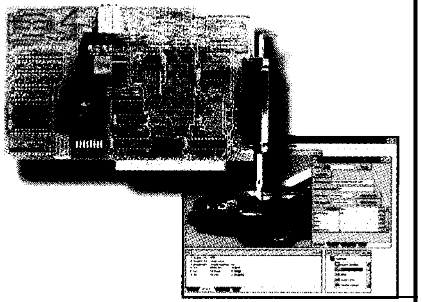
An I²C bus can have up to about 40 devices, with the limit determined by a maximum bus capacitance of 400 pF. Each device on the bus can have an address of up to 7 bits.

The open-collector/open-drain interface means that any logic-low output pulls SDA low. A device releases the SDA line by writing 1 to its output.

Unlike Microwire and SPI, which are edge sensitive, I²C is level sensitive. Data and address bits on SDA may change only while SCL is low, and the receiving device reads bits after SCL goes high.

There are two occasions when SDA changes state while SCL is high. A Start condition signals the beginning of an operation and occurs when the master brings SDA low with SCL high. A Stop condition signals the end of an operation and occurs when SDA goes high with SCL high.

After most transmissions (e.g., an instruction, address, or data byte), during the ninth clock cycle, the transmitting device releases SDA and the receiving device pulls SDA low to acknowledge that it received the bits. If the master doesn't see the acknow-



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 $195
8 channels 12-bit A/D,
16 digital I/O, Counter/Timer

High Speed
5508LC $245
8 channels 12-bit A/D,
100KHz, DMA

Multi-Functton DMA
5516DMA $295
16 channels 12-bit A/D,
DMA, 16 digital I/O

***High** Resolution*
5500HR $595
16 channels 16-bit A/D,
DMA, 8 digital I/O

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

#132

ledgment, it knows something isn't right. The receiving device releases SDA on SCL's next falling edge.

An I²C bus can have multiple masters. If more than one master tries to control the bus at once, an arbitration protocol defined by the I²C standard determines which one wins.

National's NM24C04 is a 4-Kb EEPROM with an I²C interface. SDA, SCL, power, and ground use four of the eight pins. Three other pins are seldom-used page-address inputs (AO, Al, A2), which enable multiple low-density EEPROMs on a single interface.

On some EEPROMs (e.g., NM24-C03), the remaining pin is a hardware write protect for the upper half of the memory array.

To write a byte to the EEPROM, the master issues a Start condition and writes an 8-bit slave address. The address consists of a 4-bit type identifier (1010 for EEPROMs), followed by the selected page *(000* or 001) and a request to read (1) or write to (0) the device.

The type identifier is defined by the I²C standard. The page address specifies address bit A8 (0 or 1). The other two bits in the page address are unused unless there are multiple EEPROMs.

In the clock cycle following the slave address, the EEPROM pulls SDA low to acknowledge. The master then sends an 8-bit address, and the EEPROM acknowledges. (With larger capacities, the master can send two address bytes.)

The master sends the byte to write, waits for an Acknowledge, and issues a stop condition. The EEPROM then programs the data into its memory array. When programming is complete, the EEPROM acknowledges.

To write to up to 16 bytes to sequential addresses, instead of issuing a Stop condition after the first data byte, the master may continue to send data bytes, waiting for an Acknowledge after each. After sending all the bytes, the master issues the Stop condition and the EEPROM programs the bytes and acknowledges.

Some I²C EEPROMs (e.g., Microchip's 24LC04) can program all 16 bytes in parallel for much faster programming. On others, you can write *16* bytes in sequence, but the chip programs them one at a time.

To read a byte, the master begins as if doing a write operation, sending a slave address followed by a byte address. When the EEPROM acknowledges the bytk address, the master issues a new Start condition, followed by the slave address with the final bit set to 1 (read).

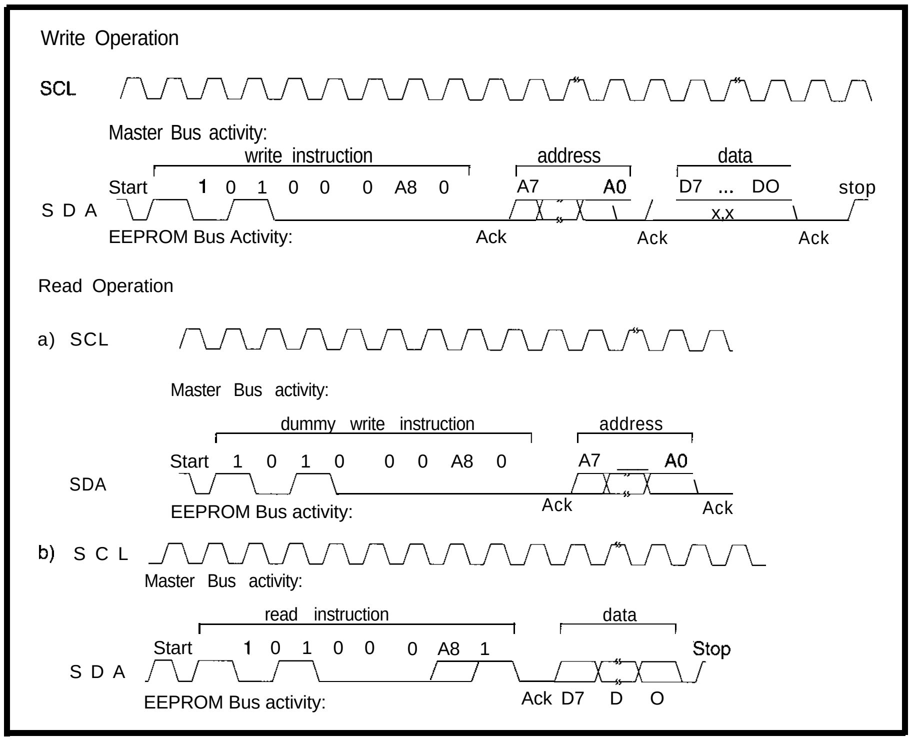


Figure 4—*The NM24C04's PC interface uses a single bidirectional signal line.*

The slave acknowledges, then writes the data to SDA. On receiving the data, the master doesn't acknowledge. Instead, it issues a Stop condition.

To read sequential addresses, the master acknowledges receiving the data byte, and the EEPROM responds by sending the next byte in sequence. The EEPROM continues to send bytes until it receives a Stop condition.

DECISIONS

Which EEPROM should you use?

When you're using a microcontroller with a built-in interface or you want to use a specific ADC in the link, the choice is obvious.

All three types are easily available and inexpensive. Digi-Key has 5 12-byte devices of each type for under $3 in single quantities.

I²C is best if you have just two signal lines to spare or if you have a cabled interface. (I²C has the strongest drivers.) If you want a clock faster than 400 kHz, choose Microwire or SPI.

For more on using serial EEPROMs, browse the manufacturers' Web pages.

In Part 2, I'll present the design of an EEPROM programmer that runs from a PC's parallel port with Visual Basic program code. ❑

Jan Axelson is the author of Parallel Port Complete *and* The Microcontroller Idea Book. *You may reach her at jaxelson@lvr.com or www.lvr.com.*

SOURCES

COP888, NM25C04, NM24C04, NM24C03
National Semiconductor
P.O. Box 58090
Santa Clara, CA 950528090
(408) 7215000
Fax: (408) 739-9803
www.national.com/design

68HCll
Motorola
MCU Information Line
P.O. Box 13026
Austin, TX 7871 l-3026
(5 12) 328-2268
Fax: (512) 891-4465
www.mcu.motsps.com/mc.html

Serial EEPROMs
Digi-Key Corp.
701 Brooks Ave. S
Thief Falls, MN 56701-0677
(218) 681-6674
Fax: (218) 681-3380

93LC66, 93C76, 24LC04
Microchip Technology, Inc.
2355 W. Chandler Blvd.
Chandler, AZ 85224-6199
(602) 786-7200
Fax: (602) 786-7277
www.microchip2.com/appnotes/appnotes.htm

8xC528
Philips Semiconductor
811 E. Arques Ave.
Sunnyvale, CA 94088-3409
(408) 9915207
Fax: (408) 991-3773
www.semiconductors.philips.com

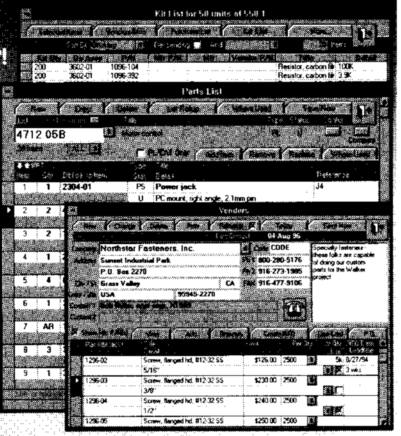
I R S

422 Very Useful
423 Moderately Useful
424 Not Useful

Parts List Manager

For Engineers, Product Designers, Prototypers

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



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

Keep track of:

- **Part Specs & Dwgs**
- **Suppliers & Mfrs**
- **Boms and Kit Lists**
- **Product Costs**
- **Engineering Stock**

Parts & Vendors™ Version 2.0

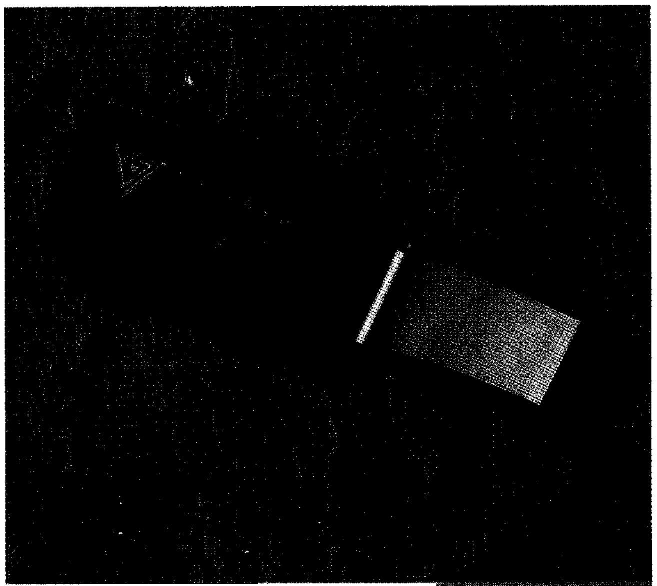
...for independent and departmental projects. Use alone or in a workgroup.

Version SE: $99 + s/h EXtended: $299 + s/h

Call 800-280-5176

Trilogy DESIGN
916-273-1985
fax 916-477-9106
P.O. Box 2270, Grass Valley, CA 95945
http://www.trilogydesign.com

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



E-Series

EPROM - FLASH - SRAM emulation and LIVE editing, 1 to 8 Mbit, 70 ns access time. 3V options.

[illegible]
$149 [illegible]

Scanlon Design Inc.
Tel (902) 425 3938 Fax (902) 425 4098
Sales & Info (800) 352 9770

#133

It Can't Be A Robot

Part 2: It Doesn't Talk

Jeff takes an RF transmitter/ receiver and sticks it on the robot he introduced last month. But, to get his robot talking, he first has to filter out the idle noise, deal with pre-amble data, and solve the problems of one-way communications.

FROM THE BENCH

Jeff Bachiochi

Every project takes much longer to complete than you anticipate. At least, that's what Murphy claims. And, my experience backs it up.

It started with my wife Beverly standing before me holding a deformed picture frame. Her "Can you fix it?" launched a thousand ships.

I assumed the Black and Decker Workmate could clamp the frame squarely while the glue set. But, I noticed it was wobbly. A bolt had worked its way loose and was now lost.

After searching for another bolt for a good 15 minutes, my best match was a bit too big for the hole in the workmate. I decided to open the hole so I could use the scavenged hardware.

Although the reversible drill was handy, it took me a few minutes to find the chuck. Once found and the bit tightened, I squeezed the trigger.

Nothing. Fatigue had reared its ugly head. The plug was gone.

Did I have a spare plug? You bet. However, it needed to be detached from an old lamp.

With a flat-bladed screwdriver, I loosened its screws and tried to attach it to the drill's cord. But, the screws wouldn't tighten. The shaft of the screwdriver was spinning in the handle. And, of course, I couldn't locate another flat-blade screwdriver.

I could have fixed the loose handle (and guaranteed myself at least 5 minutes of free time while the epoxy dried), but too many higher priority interrupts were overflowing my stack.

So, with a pair of vice grips, I grabbed the flat-blade screwdriver by the shaft and tightened the screws.

DETOUR

Similarly, this month, I intended to quickly show you how to add an inexpensive RF transmitter/receiver pair to the robotic platform I introduced last month (see Photo 1, *INK 83).*

But, when I examined the receiver's output, it contained random nqise while no carrier was being transmitted. Although a hardware UART rejects data that doesn't follow bit-timing minimums and maximums, a software UART like the one in the PicStic is more likely to accept any noise as data.

I needed to filter out as much noise as possible. So, I started by looking at what changing carrier rates were acceptable to the receiver. I used a signal generator to gate the transmitter on and off. Figure 1 shows my test results.

The receiver couldn't stay locked whenever the carrier was modulated slower than -10 Hz (this explained the noise I was seeing). Also, at the high end, it had trouble slewing above a 10-kHz modulation rate.

It looked as though I might be able to squeak through data rates of up to 9600. To prevent extraneous noise

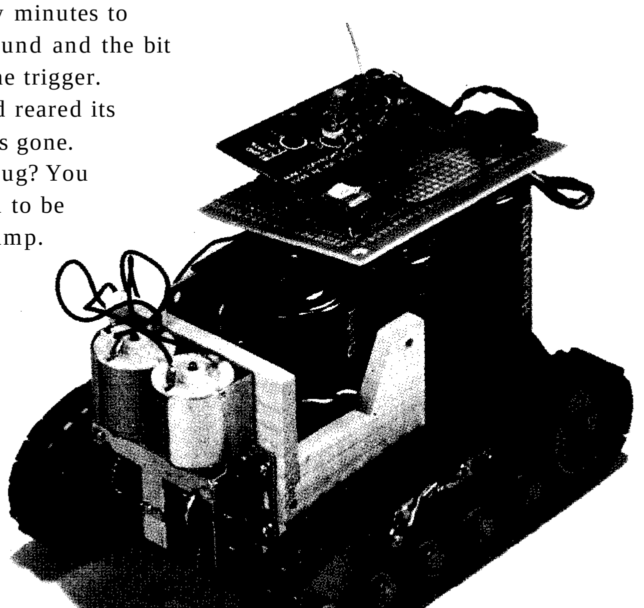


Photo 1—*Three D-cells power this robotic platform. The filter and RF receiver are mounted atop the platform.*

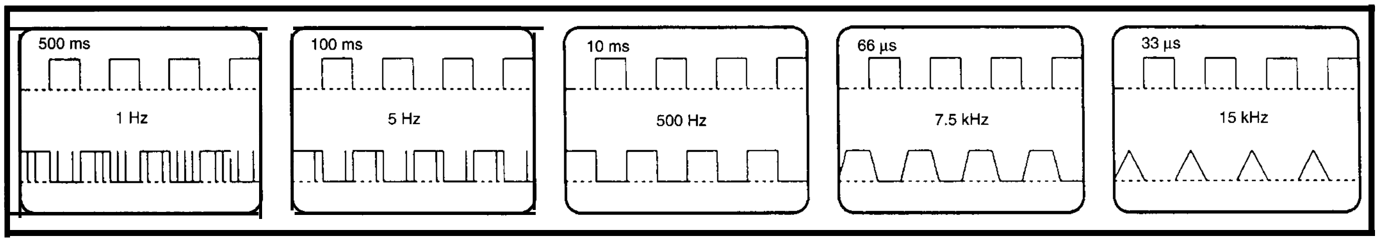


Figure 1--The upper *traces show the* **frequency generator's** *input to the RF* **transmitter, while** *the lower traces* **indicate** *the output of the RF* **receiver. Notice** *how the noise* **increases when** *the* **modulation is slow** *and how the signal's slew rate is* **limited** *at the* **high** *end.*

from creating havoc with the PicStic's software UART, I wanted to create as narrow a passband as possible.

To be safe, I chose a minimum time slot of ½ a bit time and a maximum time slot of 12 bit times. When data transmits, the ½-bit-time minimum assures that worst-case data (alternating 1s and 0s) passes without interference.

On data that's all 1s, there are at least 9 bits (8 data bits and a stop bit) in which the carrier doesn't change (plus the intercharacter spacing), hence the choice of 12 bit times as a maximum time slot.

My challenge: to do the filtering digitally by using a micro with few-or better yet, absolutely no-external components. By sampling the incoming data -10 times per bit time, I hoped to reduce the phase and resolution error as much as practically possible. (The internal RC oscillator of the PIC12C-508 I planned to use for the filter has about 10% accuracy over voltage and temperature.)

The pseudocode in Listing 1 shows my thought process for this digital filter. At the beginning of each sample, a timer is set and the input data is sampled and stored as the in_flag bit. By comparing the in_flag bit with the last_flag bit, a potential change of state can be determined.

If a change took place, good_flag is checked. This flag indicates whether the last logic state was completed within the time slot allotted.

If it was good, a good-bit counter is incremented (but prevented from exceeding 255). If it was bad, the good-bit counter is cleared along with the data output bit and the carrier-detect output bit.

Next, the 0-state counter is cleared if data_in is 0. Otherwise, the 1-state counter is cleared.

Finally, last_flag is updated with in_flag in preparation for the next sample. The timer is polled for an overflow (there are no interrupts here] before beginning a new sample.

If no change of state is detected, checking in_flag routes the program flow to one of two identical routines. A 0 steers the flow into the 0-state routine, and a 1 jumps to the 1-state routine.

These program paths check to see how long the sampled data has been unchanged. If my specifications call for 10 samples per bit time and the minimum time slot is ½ bit, then the minimum number of consecutive samples must be 5 to be considered good.

At the other end, the maximum time slot is **12** bit times. Therefore, the maximum samples must be 120 (10 x 12 bits) to be considered good.

If the consecutive sample counter CNTRx (where x = 0 for 0-state and 1 for 1-state) is within the good range, good_flag is set. Otherwise, it is cleared. (This flag was checked in the change-of-state routine to determine whether to clear or increment the good-bit counter **CNTRG.)**

If good_flag isset,then CNTRG is compared to the value 8 (an arbitrary value), which indicates the number of consecutive good bits that must be received before the carrier detect output is set and the input data is allowed to pass through the filter.

CNT Rx is now incremented but held at a maximum of 255. Finally, the timer is polled for an overflow (no interrupts here) before beginning a new sample.

TIME TRIALS

Up to now, everything was done without knowing the exact execution times and thus the maximum serial data rate (throughput). The longest execution path for this code requires 35 cycles or 35 µs with a 4-MHz internal clock. Now for some calculations.

At 4 MHz, the routine takes at least 35 µs to execute. Table 1 shows that 2400 bps is the fastest rate the routine can handle and still have time to complete [that's with only 7 µs to spare).

Since the timer has no interrupt, it must be polled. The polling loop requires 4 µs to execute (i.e., grab the timer count, test for zero, and if necessary, jump back to poll again).

After the timer is grabbed, it continues to count during the next three instructions. It could pass through zero while one instruction executes if the timer doesn't have a prescale divisor.

Choose divide by 4 to ensure the timer remains at each count for those four instruction cycles of the loop. Therefore, the timer should be reloaded with a number that's actually ¼ of the count you're looking for.

Now, we know what to expect from the filter and can fill in a few blanks. If the loop time set by the timer is to be -42 s (actually since it must be divisible by 4, we need to choose either 40 or 44 µs), the minimum and maximum number of loops necessary can be defined.

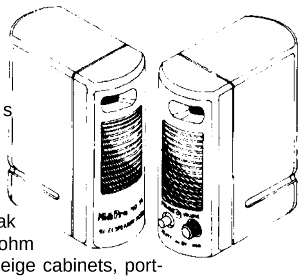
The minimum time slot equals 5 loops (208 µs) or 5/10 of a bit time. The maximum time slot equals 120 loops (5040 µs) or 12 bit times.

Baud Rate	Bit Time (µs)	1/10-Bit Time (µs)	Execution Loop > 35 µs
19200	52	5	no
9600	104	10	no
4800	208	20	no
2400	416	42	yes
1200	832	83	yes
600	1663	166	yes
300	3326	333	yes

Table 1—*With an execution loop of 35 ms, this chart shows that 2400 bps is the fastest data rate this filter can accept.*

ALL ELECTRONICS
CORPORATION

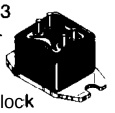
AMPLIFIED MULTIMEDIA SPEAKERS



Midipro # MD58 Amplified stereo speakers for computer/multimedia use. 3 watts RMS/ 12 watts peak full range 8 ohm speakers. Beige cabinets, ported for extra response. 7.24" x 2.95" x 4.75". Operates on 6 volts. Includes cables. Requires either 4 C cell batteries or 6 to 12 volt power adapter (not Included).
CAT # SK-58 **$14.95 pair**

Above speaker set including DC POWER ***ADAPTER/BATTERY*** *ELIMINATOR*
CAT# SK-58A $17.95 per **set**

5 amp Solid State Relav



C.P. CLARE /Theta J # JTA2405-3 Compact, TTL compatible, optically isolated solid state relay for loads up to 5 amps @ 240 vac. 0.8" x 0.82" x 0.56" high epoxy block with a 1.4" long metal mounting flange. 1 .19" mounting centers. 0.062" dia. x 0.175" high pins. Pins can be pc mounted wrapped and soldered UL and CSA listed. Input: 4 - 8 Vdc Load: 5 amps @ 240 Vac
$6.00 each CAT# SSRLY-2405
10 for $55.00

470 UF, 450 VOLT SNAP-IN CAPACITOR

Nichicon LGQ2W471 MHSC 1.375" diameter x 2" high. 0.4" lead spacing.
$4.50 each CAT# EC-4745
10 for $40.00

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 $3.00 postage.

MAIL ORDERS TO:
ALL ELECTRONICS CORPORATION
P.O. Box 567
Van Nuys, CA 91408
FAX (818)781-2653
E-Mail allcorp@allcorp.com
Internet - http://www.allcorp.com/

#134

Listing I--Here's *pseudo code showing data sampling and comparisons of min, max, and good bits for a legal output.*

```
begin
  initialize chip;                        /* 1 cnt = 42 us */
  min:=5; max:=24; good_bits:=8;
top :
  tmr0:=cnt;
  in_flag:=data_in;                       /* GP2 (Data in) */
  if in_flag<>1 then
    got0 in0;
in1:
  if last_flag=1 then
    got0 samp1;
  else
    got0 cos:
in0:
  if last_flag=1 then
    got0 cos;
  else
    got0 samp0;
cos:
  if good_flag=0 then
    cntrg:=0;
    cd:=0;                                /* GP5 (CD output LED) */
    data_out:=0;                          /* GP4 (Data out) */
  else
    cntrg:=cntrg+1;
  if cntrg=0 then
    cntrg:=256;
  if in_flag=0 then
    cntr0:=0;
  else
    cntr1:=0;
  last_flag:=in_flag
  goto bottom:

samp1:
  if cntr1<min or cntr1>max then
    good_flag:=0;
  else
    good_flag:=1;
    if cntrg>=good_bits then
      data_out:=1;                        /* GP4 */
      cd:=1;                              /* GP5 */
    else
      cd:=0;                              /* GP5 */
      data_out:=0;                        /* GP4 */
  cntr1:=cntr1+1;
  if cntr1=0 then
    cntr1:=255;
  goto bottom:

samp0:
  if cntr0<min or cntr0>max then
    good_flag:=0;
  else
    good_flag:=1:
    if cntrg>=good_bits then
      data_out:=0;                        /* GP4 */
      cd:=1;                              /* GP5 */
    else
      cd:=0;                              /* GP5 */
      data_out:=0;                        /* GP4 */
  cntr0:=cntr0+1;
  if cntr0=0 then
    cntr0:=255;
  goto bottom:

bottom:
  if tmr0=0 then
    got0 top
  else
    goto bottom
end
```

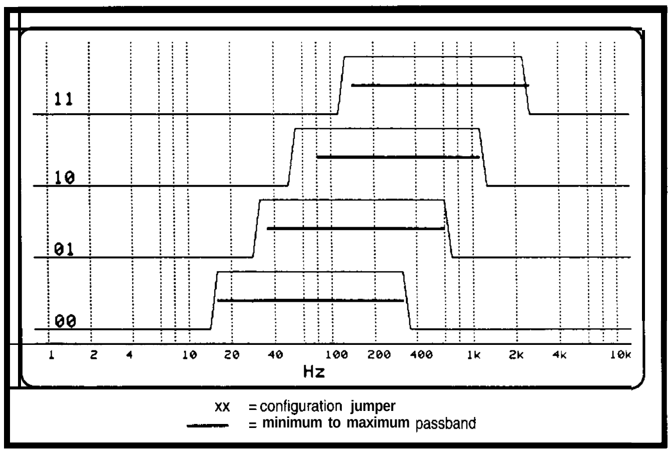


Figure 2—*Two* **configura-***tion* **bits choose different timer reload values, thereby shifting this digital** *filter's bandpass region.*

By simply changing the timer reload value, the execution loop can be increased. Spare input pins let the user select one of four possible reload values, adjusting the filter to pass either 300, 600, 1200, or 2400 bps.

Again, using the frequency generator, I ran test inputs through the filter. The results are in Figure 2.

PREAMBLE

Transmission of data presumes a few things. First, it takes 8 good bits (an arbitrary number] before anything passes through the filter. This isn't a problem since the receiver requires a certain time to get synced-up.

Some kind of preamble data wakes up the filter and the receiver's front end. If some of the first transmitted bits are lost, how can we know the UART will sync up on a true start bit?

Sending capital **Us** using an 8N1 data format looks like a stream of 1s and 0s. No matter where a UART starts receiving data, it will look like it's receiving a U.

If these **Us** are followed by a character 2 5 5 (the sync character), the UART will finish a byte sometime during this character, and there's only a small chance it will be a **U.**

Since there will be no additional bit transitions during this character, the UART will be ready for a new start bit when the actual data begins immediately following this sync character.

So at the transmission end, a few extra bytes of data must be appended to each new message transmitted (a new message consists of any transmission which begins after more than one stop-bit delay).

At the receiving end, the UART reception is ignored until an **M is re**ceived (i.e., the first legal character in a command). So, the preamble is essentially invisible.

RETURN FROM INTERRUPT

I think we're about back to where I expected to be at the beginning of this article-cutting the robot's umbilical cord without giving it much of a purpose in life.

A Ming transmitter/receiver pair works nicely (once this month's filter

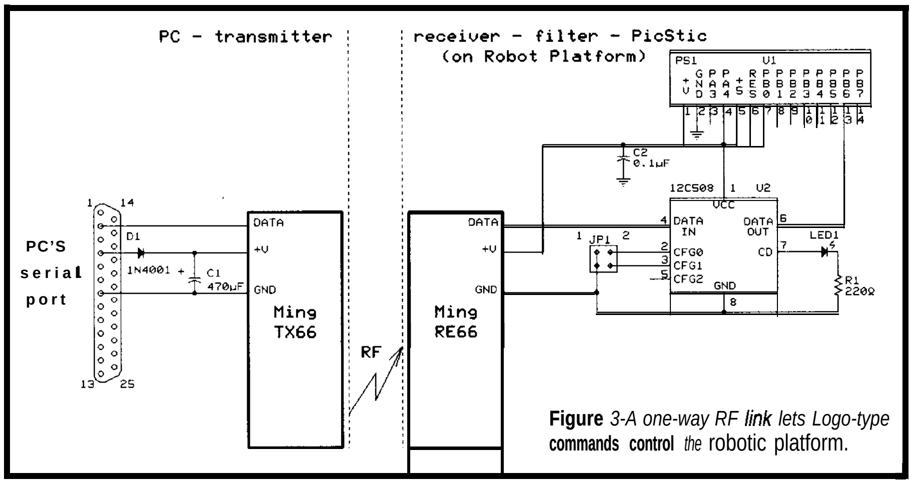


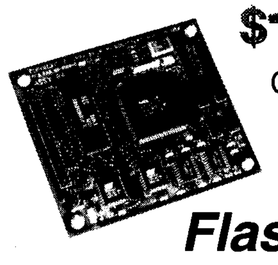
Figure *3-A one-way RF* ***link*** *lets Logo-type* **commands control** *the* robotic platform.

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**

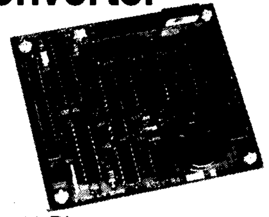
✔ 10 Mhz/8 Mhz CPU ✔ 2 Timers
✔ 512 k bytes RAM ✔ 4 Interrupt Lines
✔ 512 k/256 k FLASH ✔ 8 Analog Inputs
✔ 2 Serial Ports ✔ X-Modem File Transfer
✔ 24 Parallel I/O Lines

INCLUDES DOS & Utilities

USE YOUR TURBO C COMPILER OR QUICKBASIC COMPILER
SAVE TIME, MONEY AND HEADACHES

A/D Converter

$95 Qty 1 Price

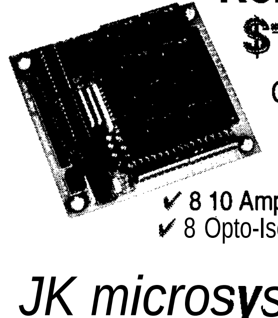


✔ 8 Channels, **12 Bits**
✔ 6 µs. Conversion Time
✔ Clock/Calendar Option
✔ Includes Drivers & Apps.

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

#135

Tight Budget '51 Emulation

8051 Family Emulator is truly Low Cost!

The DryICE Plus is a modular emulator designed to get maximum flexibility and functionality for your hard earned dollar. The common base unit supports numerous 805 1 family processor pods that are low in price. Features include: Execute to breakpoint, Line-by-Line Assembler, Disassembler, SFR access, Fill, Set and Dump Internal or External RAM and Code, Dump Registers, and more. The DryICE Flus base unit is priced at a meager $299, and most pods run only an additional $149. Pods are available to support the 803 1/2, 8751/2, 80C154, 80C45 1, 80C535, 80C537, 80C550, 80C552/62, 80C652, 80C85 1, 80C320 and more. Interface through your serial port and a comm program. Call for a brochure or use INTERNET. We're at **info@hte.com** or www.hte.com.

Tighter budget? How about *$149* emulation?

Our $149 DryICE model is what you're looking for. Not an evaluation board - much more powerful. Same features as the DryICEPlus, but limited to just the 8031/32 processor.

You can afford it!

So, if you're still doing the U V Waltz (Burn-2-3, Erase-2-3). or debugging through the limited window ROM emulators give, ***call us now*** for relief! Our customers say our products are still be Performance/Price emulators available!

Look into our Single Board Computer solutions, too!

HiTech Equipment Corp.
9672 Via Excelencia
San Diego, CA 92126
[Fax: (619) 530-1458]

Since 1983

(619) 566-I 892

Internet e-mail: info@hte.com
World Wide Web: **www.hte.com**

#136

```
Choose Command
    A=add step to program
    V=view program steps
    E=edit program steps
    C=clear program steps
    P=play program steps
    S=save a program to a file
    G=get a program file
    X=exit program

Viewing program steps
1   MF 10
2   MR 45
3   MF 10
4   MR 45
5   MF 10
6   MR 45
7   MF 10
8   MR 45
9   MF 10
10  MR 45
11  MF 10
12  MR 45
13  MF 10
14  MR 45
15  MF 10
16  MR 45
Finished
```

Figure 4-A logo-style program written in GWBASIC lets the robotic platform execute a number of programmable steps or functions (i.e., move forward 10").

is added to get rid of the idle noise) for sending commands to the platform using RF (see Figure 3).

It led me to write a GWBASIC program on my PC that enables a routine of commands to be entered, listed, edited, saved to a file, loaded from a file, and sent to the robot over the RF link. The robot becomes a Logo type operator as shown in Figure 4.

I showed four robot commands last month-Forward, Backward, Left Turn, and Right Turn. My program lets the user choose any of these and prompts for a distance in inches or degrees.

Through trial runs, I determined the actual motions and added multiplier constants for the distance and degrees. It's more user friendly than asking for the distance and direction in units.

You can add commands to create a lengthy repertoire of actions. Since the program lets you save and retrieve, you can use a simple text editor to create a movement command list.

There is a potential problem with one-way communications. There's always the possibility that a command could be jumbled or lost, and you'd never know it until the robot crashes into something unexpected.

With the umbilical cord, there was an echo of the command--the hand-shaking--to assure you the command was done. This isn't possible with only one transmitter and one receiver.

So, again I needed to add a multiplier constant. This time, and I do mean time, I needed a way to pause between commands just long enough to be sure the last command had completed.

I used the number entered either for distance or degrees and multiplied it by the constant to get a pause duration in fractions of a second. The BASIC Timer command let me easily and accurately wait an appropriate time, based directly on how far the platform was going to move.

LIFE BEYOND THE KEYBOARD

If I can pry this little robot away from my kids long enough, I'll add some sensors next month. Supposing this thing is ever to roam on its own, I need to add some collision avoidance.

Speaking of collision avoidance, maybe I should design some for the picture too since that's how it got broken. Perhaps its glue is dry by now. Time to get it back on the wall. ❑

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

SOURCES

PIC12C508
Microchip Technology, Inc.
2355 W. Chandler Blvd.
Chandler, AZ 85224-6199
(602) 786-7200
Fax: (602) 786-7277
www.microchip.com

Ming-RX66 transmitter and **RE66 receiver**
Digi-Key Corp.
701 Brooks Ave. S
Thief Falls, MN 56701-0677
(218) 681-6674
Fax: (218) 681-3380

I R S

425 Very Useful
426 Moderately Useful
427 Not Useful

SILICON UPDATE

Tom Cantrell

Cruise the Funchips

Often, Tom toys with the latest monoliths, and he's hard-pressed to get all the gory details down before the real-estate runs out. Today, he looks at a box full of gadgets that take care of time, temperature, and voice.

I understand others actually experience something called "vacations." I'm not one of 'em, though you could argue my entire life qualifies since I enjoy my work. If I was an MIS administrator or insurance salesman, it might be another story.

There seem to be two schools of thought when it comes to vacations. One is to try to pack as many locations and sights as possible into a whirlwind tour. The other is to go bury your head in the sand on a beach somewhere.

When it comes to ICs, the latest mega-monolith microchips are so complicated, it's easy to get a headache just trying to explain them. Like the overcaffeinated vacation, there's so much ground to cover, you can't enjoy the sights.

So, why don't we take a nice calming cruise on the Funchips? These gadgets are economical and fun, and your tour guide doesn't have to be a Ph.D.

MY KINGDOM FOR A CLOCK

If there's one hassle all of today's fast-paced designs run into, it's the need to generate and control high-speed clocks.

Traditional embedded setups running at O-20 MHz or so have had it easy. This range is where the ubiquitous fundamental overtone crystal rules and manufacturers have really got their oscillator designs perfected.

Short of adding a couple of caps and obeying common-sense layout rules, there's little hassle for the designer. Wire it up, and it works.

Ah, but what if your system needs a rev-up to 30, 40, 50 MHz and beyond? Turns out, it's tough to fabricate such high-speed fundamental crystals-at least ones that aren't ridiculously fragile. A number of alternatives have emerged over the years, but all of them come with irritating consequences.

From way back when, one approach has been to extract an overtone of the fundamental crystal. Unfortunately, it seems there are just too many start-up and reliability problems.

I'm not enough of an expert on oscillator design (a rather black art) to know exactly why this is the case. But, you don't have to be a guru to understand that if the overtone trick worked, its use would be much more widespread.

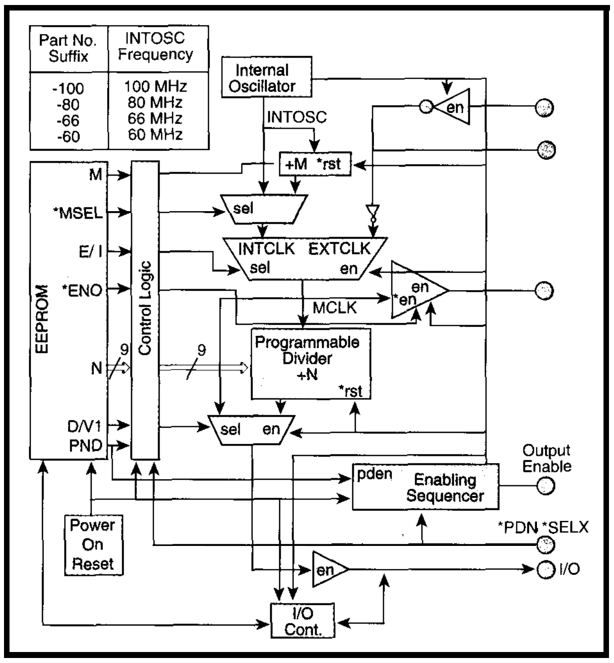


Figure 1--The *Dallas* **Semiconductor DS107.5** *EconOscillator* **is a** *200-kHz to 100-MHz* **clocking solution.** *EEPROM stores the* **divide ratios for automatic operation** *at powerup.*

Instead, most designers just punt and design in the stalwart DIP-can hybrid-TTL oscillator. These devices are OK, but they're pricey, bulky, and high power. The simplest ones also lack any means of controlling the clock, short of switching power on and off and grappling with the resulting glitches.

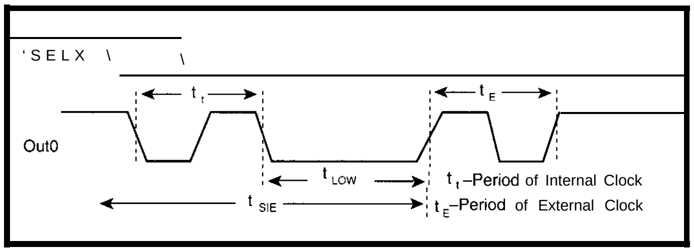

'SELX
OutO
t_I
t_E
t_{LOW}
t_{SIE}
t_I–Period of Internal Clock
t_E–Period of External Clock

Figure 2—*The* **DS107.5** *allows* **switching** *between* **external and internal clocks.** *The output is he/d low during the transition, and full cycles on both* **sides** *are* **guaranteed.**

From a system designer's perspective, the best thing to do is fingerpoint the chip designer and demand they design in a PLL clock multiplier. While they're at it, tell 'em to make sure it can be turned off (high-frequency PLLs use a lot of power) yet lock up real quick when called into action. Of course, it would be nice if they didn't increase the chip's price either.

Enter the Dallas DS1075 Econ-Oscillator. It handles high-frequency clocking with quite a bit of panache.

As shown in Figure 1, this puppy (8-pin DIP or SOIC package) starts with a fixed rate (60, 66, 80, or 100 MHz) internal oscillator feeding a divide by 1/2/4 prescaler and 9-bit divider.

That works out to 1536 different choices (though many overlap), covering a range from 100 MHz all the way down to 200 kHz (the specified minimum output frequency). Between the four speed grades, prescaler, and divider, it should be possible to come quite close to the frequency you need. Do note the ±0.5% accuracy and ±1% temp variation isn't as good as a crystal oscillator (typically about ±50 ppm or ±0.005%).

If all the choices on chip aren't enough, the chip also accommodates an external clock input-either TTL on the OSCIN pin or a crystal between OSCIN and XTAL. Assuming the PDN/ * SELX pin is configured for the * SELX (select) function, it can be used to switch dynamically between the internal and external timebases.

Notably, the switch is glitchless (the primary output, IN/OUT, is held low during the changeover) as you see in Figure 2. A secondary output, OUTO, grabs the output of the internal/external selection mux.

The '1075 also features an OE (output enable) pin that uses similar enable and disable sequencing to eliminate any truncated clocks or variable phasing (i.e., the divider chain is reset by OE transitions). Note that OE controls the main (IN/OUT), but not the secondary (OUTO), clock output.

Though drawing up to 50 mA when running, the *PDN/ • SELX pin invokes power-down mode. This action shuts off the oscillator and both clock outputs, thereby cutting power consumption to the bone (< 1 µA).

There are a couple compromises that come with the package. In other words, eight pins don't go a long way.

While internal/external selection, output enabling, and powerdown are all intended for dynamic operation, the actual divide ratios aren't. Instead, they and other key mode selections (e.g., which function the ● PDN/*SELX pin performs) are preprogrammed into EEPROM for no-programming startup.

Programming is accomplished by connecting a pullup to the IN/OUT pin. When power is applied, the chip detects the high input as a signal to use the IN/OUT pin for programming rather than as the primary clock output.

The programming scheme itself is based on the unique Dallas one-wire LAN protocol (see "The Little LAN That Could," INK 71), in which self-timed devices achieve bidirectional communication over a single wire. Once the divide and mode bits are written.

power must be removed, the pullup disconnected, and power restored for any changes to take effect.

The need to power cycle the chip and the dual use of the single pin as both a programming port and the clock out hinder self-clocking schemes.

For instance, it would be neat if the chip relied on EEPROM to boot up a micro at a default clock rate but then allowed the micro to dynamically change it (admittedly, a scheme fraught with risk, but intriguing nevertheless).

Putting aside wild and crazy ideas (at least for now), it seems clear the high speed and configurability of the DS 1750 means it's something a well-traveled designer shouldn't forget to pack.

EZ TEMP

Another chip with a wire-miser serial interface (in this case, I²C) is the National LM75 digital temperature sensor. As depicted in Figure 3, the chip combines a raw temp sensor, signal conditioning, ADC, set-point comparator, and I²C interface in its tiny S-pin SOP package.

Key specs include 3.0-5.5-V operation and low operating (< 1 mA typical) and quiescent (1 µA typical) currents. As well, it has decent accuracy (±3°C), considering the very wide temperature range of -55" to 125°C. Indeed, accuracy improves to ±2°C over the more temperate -25" to +100°C range.

The pinout is blessedly simple. There's power (3-5.5 V) and ground, the two-wire (SDA and SCL) I²C interface, three address lines specifying the least significant bits of the I²C address (i.e.,

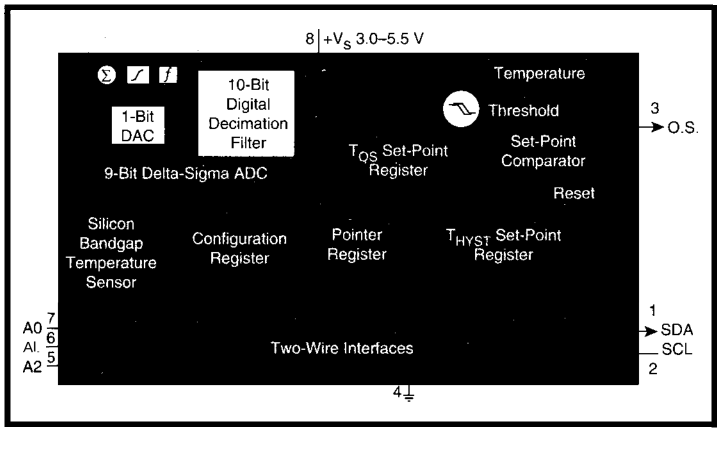


Figure 3—*One* **of the most recent digital** *temp sensors, the LM75, features an PC interface.*

up to eight LM75s can be connected), and an overtemperature shutdown (OS) output.

This output can be programmed to operate in either comparator or interrupt mode. Figure 4 illustrates how the former is like a typical thermostat, in which the output directly reflects the comparison result subject to programmed hysteresis.

By contrast, interrupt mode generates a pulse (programmably low or high) on each comparator transition. The interrupt is cleared by reading any of the four on-chip registers or placing the LM75 in low-power shutdown mode.

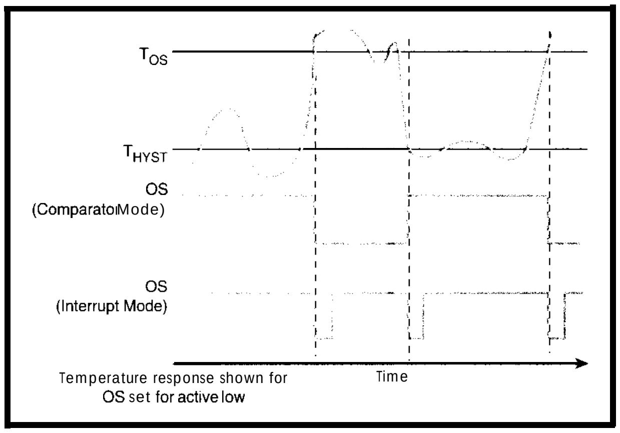


Figure 4—*The* **OS** *(Overtemperature Shutdown) pin can be programmed to operate in either comparator (i.e., thermostat) or interrupt mode.*

The OS output is open collector without an internal pullup, enabling it to be wire-ORed with other active-low sources. The datasheet cautions to use a weak (e.g., 30 kΩ) pullup to minimize self-heating. Remember, the most direct thermal connection is from die to pins to PCB.

To minimize I²C address consumption, the LM75 is programmed via a single pointer register that directs access to the temperature, setpoint, hysteresis, and configuration registers as shown in Figure 5.

The configuration register controls the previously mentioned features (i.e., low-power shutdown, compare and interrupt modes, and OS pin polarity). In addition, two fault-queue bits function as a low-pass filter by specifying that 1, 2, 4, or 8 consecutive samples must pass inspection before allowing an OS transition.

There's no need to explore the details of I²C since it's been well covered in *INK* and elsewhere. The LM75 is an I²C slave (i.e., the host provides the clock). Do note that the pointer and configuration registers are 8 bits wide, while the temp, setpoint, and hysteresis registers are 16 bits wide.

Watch out for inadvertently trying to perform an 8-bit read from a 16-bit register. If D7 (i.e., the ninth bit shifted) is 0, things can deadlock with both the CPU and LM75 waiting for the other to do something.

The fixup requires the CPU to issue nine additional clocks to get things back in sync.

Easy-to-use, not to mention reasonably priced ($1.59 in 100s) digital temp chips like the LM75 inspire a few moments of nostalgia for the good old days of diodes and op-amps.

Oops, no time to mourn. There's one more stop on our cruise.

BLABBER CHIP

Ever get stuck on a plane, train, or automobile next to somebody who just wouldn't shut up? Subtle hints like burying your nose in a magazine, putting on headphones, or speaking a foreign language seem to blow right by those determined to chat.

Better hope it's not a long trip if the TriTech TR83 100CF voice storage controller (see Figure 6) plops down next to you. Handling up to 14 min. of speech, this chip can talk your head off.

Of course, another strategy for dealing with a seatmate who won't shut up is to drink until either what they're saying sounds interesting or, better yet, you feel like sharing your own story.

Fortunately, the TR83 100CF is a good listener as well (i.e., it's both a recorder and player). It's quite similar in concept to the single-chip solutions from ISD (see "Talking Chips," *INK 36)* except storage capacity is much higher, thanks to the TriTech chip's reliance on external flash memory.

Wiring the '83 100 starts with connecting a crystal (20 MHz), microphone

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

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." - P. 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

- ❏
- ❏ Instant Boot, Console Redirection, & Much More
- ❏ Expert Support with Guaranteed Response Time
- ❏ We Work Closely With Acer, AMD, 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

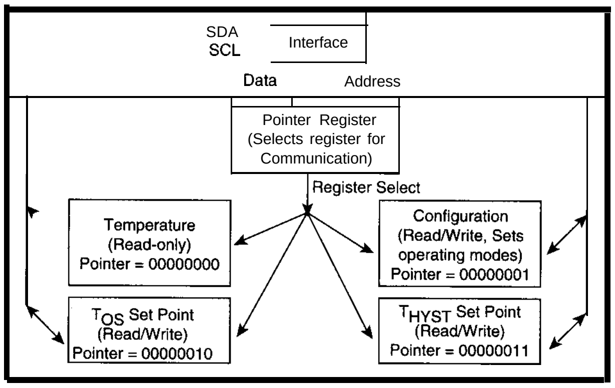
For Details, Call 1-800-850-5755

GENERAL GS SOFTWARE

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

#138



SDA
SCL
Interface
Data
Address
Pointer Register (Selects register for Communication)
Register Select
Temperature (Read-only) Pointer = 00000000
Configuration (Read/Write, Sets operating modes) Pointer = 00000001
T_{OS} Set Point (Read/Write) Pointer = 00000010
T_{HYST} Set Point (Read/Write) Pointer = 00000011

Figure B-Access *to* **the four LM75 on-chip control registers is via a pointer register.**

(electret, AC coupled), and speaker (32 Ω). The analog section includes practically everything (i.e., amps, AGC, and converters), though the active filter (shared between input and output) requires tuning with a handful of external Rs and Cs.

The '83 100 direct connects to a variety of parallel (8-bit data) flash chips. Up to two memory chips can be used (i.e., two chip-select pins), but both must be of the same type.

With a 19-bit address bus, l-, 2-, and 4-Mb flash chips are supported, so the maximum storage is 8 Mb [i.e., two 512 K x 8 chips). With selectable 10- or 20-kbps ADPCM compression, that translates to 14 or 7 min., respectively.

Like other chips used in digital answering machines, cellular phones, toys, and voice memo gadgets, the '83100 features a rather self-explanatory push-button interface [e.g., Play, Rec, etc.). There are also a couple LED outputs giving operationalfeedback (e.g., LED2 goes on during recording).

Since the unit is intended for push-button use, the inputs are not only debounced (30 ms), but they perform special functions when held. So, if you push the Next button during playback, it goes to the next message, but if you hold it down for more than 0.5 s, it plays back the current message fast (150%) until released. Similarly, the Erase and Erase All inputs require 0.5 and 1 s of convincing, respectively, before they do their dirty deeds.

Power consumption during playback (77 mA typical at 5 V) is dominated by the amplifier. However, when otherwise not busy for 3 s, the '83100 automatically enters a 10-µA power-down mode.

A falling edge on any of the button inputs automatically awakens the chip and executes the appropriate command, making powerdown completely transparent to the user.

Naturally, it's no problem to coerce a micro into dealing with the '83 100 push-button interface. However, the company mentions that since the chip is based on an 8-bit micro, they can modify it (e.g., with a serial interface) for different applications.

As it stands, the '83100 is really best suited for push-button-based designs. Keep in mind that there's no way to directly access a particular message out of order or string arbitrary sequences together.

If you just need a playback-only chip with a more micro-accessible interface, check out the IQ Systems IQS800. Other than lacking the record feature (audio data is prepared ahead of time on a PC), it's otherwise quite similar to the TriTech chip.

BACK TO REALITY

I look forward to more relaxing visits with chips that harken from a simpler time. Nothing like a vacation to recharge the old batteries.

But, back home in Silicon Valley, things move at a faster pace. Refreshed by the R&R, I'm a little better prepared to deal with the next zillion transistor wunderchips that come down the pike. Bring 'em on! ❑

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@circuitcellar.corn, by telephone at (510) 657-0264, or by fax at (510) 657-5441.

SOURCES

DS1075 **EconOscillator**
Dallas Semiconductor
4401 S. Beltwood Pkwy.
Dallas, TX 75244-3292
(214) 778-6824
Fax: (214) 778-6004
www.dalsemi.com

IQS800
IQ Systems, Inc.
75 Glen Rd.
Sandy Hook, CT 06482
(203) 270-9064
Fax: (203) 270-9064
www.iqsystemsinc.com

LM75
National Semiconductor Corp.
1111 W. Bardin Rd.
Arlington, TX 76017
(408) 721-5000
Fax: (817) 468-6935
www.national.com

TR83100CF
Tritech Microelectronics Intl.
1400 McCandless Dr.
Milpitas, CA 95025-1900
(408) 894-1900
Fax: (408) 941-1301
www.tritech-sg.com

I R S

428 Very Useful
429 Moderately Useful
430 Not Useful

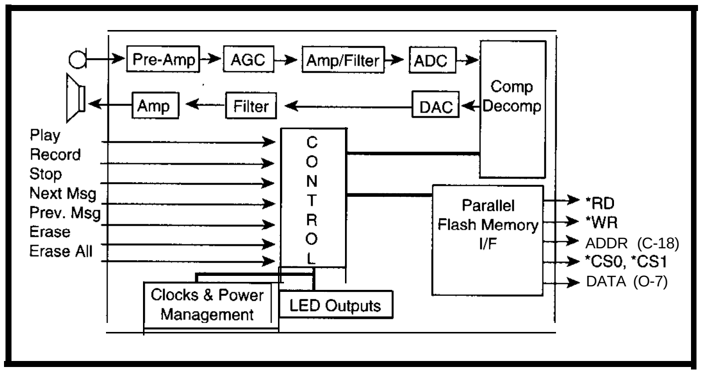
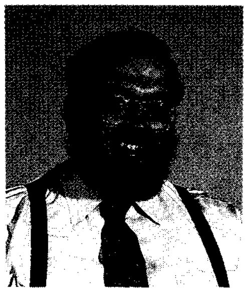


Figure 6—*The TriTech TR83100CF* **is a voice recorder that works with standard** *external* **flash chips to provide** *up to* **14 min. of storage.**

PRIORITY INTERRUPT

Don't Lose Your Head



While I certainly don't think executions are anything to joke about, there is a gag about the French Revolution that has a message buried in the humor. Perhaps you've heard the one about the engineer and the guillotine?

Apparently, a number of different professionals were about to be executed. A lawyer was led up the stairs and placed in the guillotine. When the executioner pulled the rope to drop the blade, it stopped just above the lawyers neck. The crowd gasped. Astonishment gave way to exuberant cheers at the obvious display of divine intervention. Consequently, the lawyer was released.

Next, a doctor was marched up the stairs and placed in the guillotine. Miraculously, the blade stopped short again as he too must have been divinely blessed. Astoundingly, the magic continued as an accountant and a clergyman were presented to the headsman. Each smiled, bowed in appreciation to the crowd, and then walked back down the stairs.

Finally, an engineer was placed in the guillotine. As the blade was about to be released, he interrupted the headsman, "Wait a second. Turn me over so I can get a better look at the blade guides. I've been watching this pretty closely and I think I know what your problem is. If you release my hands for a minute, I'm sure I can fix it."

The moral of this joke should be obvious. I'd also bet that I'm not alone in having done exactly what the big laugh is about. Perhaps it's something about the breed that makes us focus so much on problem solving that we often miss seeing the forest for the trees.

Without divulging how ancient I must really be, let me just say that I was there at the birth of the computer revolution. I'm not merely referring to having existed during the same chronological period. I mean that I was actually present at many of the important events and contributed a few myself. I remember having dinner with people who are now considered the famous and fabulous in Fortune. I sat though discussions about forming little startups that have become the megacompanies of today. I was there at the first stock offerings of Lotus, Compac, Microsoft, etc., etc.

Did I take optimum advantage of being in the right place at the right time? In retrospect, it's certainly true that I could have capitalized on several opportunities that I didn't. I was more concerned about the engineering than the business challenge.

Don't get me wrong. I'm not complaining. I'm just making a gross generalization based on a little personal experience. Perhaps it's the nature of the person that selects this profession, but problem solving for engineers often becomes so consuming that there's little time to view the big picture. In my case, I was fixated on discovery. Having a magazine pay me to write about whatever technical adventure I chose made it a fantasy avocation.

Technology continues its evolution. The inventions and innovations today in communications, biotechnology, and software are equivalent in magnitude to the discoveries of the past. While it can be argued that I surely haven't suffered from not owning treasury stock in Microsoft or Lotus, or from not patenting the numerous ideas in my articles that are now public domain, I regret that I sometimes failed to take advantage of many opportunities simply because I was too busy building the invention rather than thinking about its business impact.

The key is remembering the marketing end of things as well as the engineering solution. We are called on to create inventions which solve problems'for others. No engineering school prepares you to think about the business possibilities. But, recognizing that one of your projects or the technology involved is a big deal may not require all that much thought.

Concentrating solely on engineering solutions and not taking into account your own financial potential may not be so different than repairing the guillotine.

Steve

steve.ciarcia@circuitcellar.com