

www.circuitcellar.com

CIRCUIT CELLAR®

THE MAGAZINE FOR COMPUTER APPLICATIONS

#117 APRIL 2000

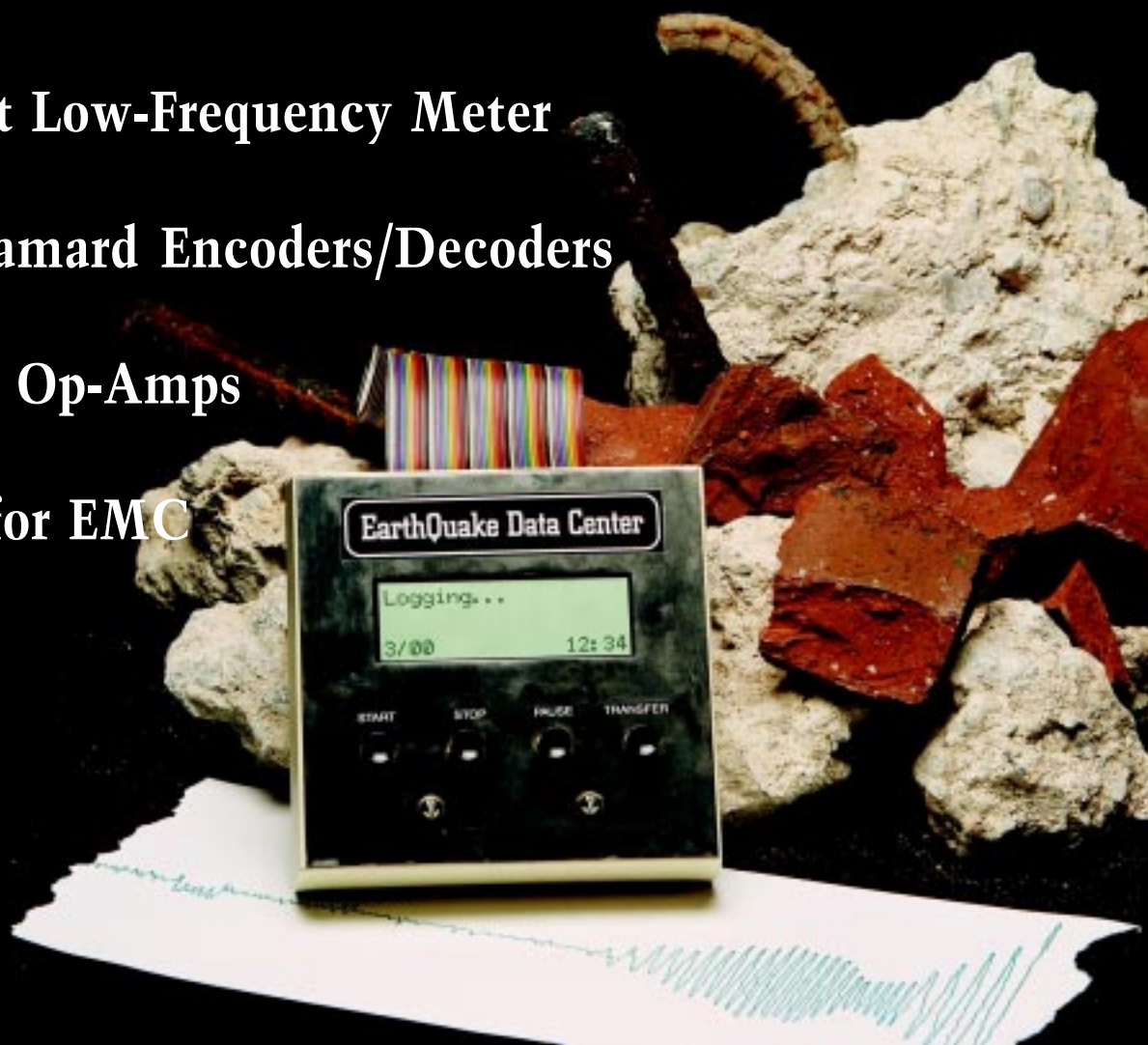
SIGNAL PROCESSING

Build a Fast Low-Frequency Meter

Walsh-Hadamard Encoders/Decoders

Know Your Op-Amps

Designing for EMC



CIRCUIT CELLAR **ONLINE**

Double your technical pleasure each month. After you read *Circuit Cellar* magazine, get a second shot of engineering adrenaline with *Circuit Cellar Online*, hosted by ChipCenter.

— FEATURES —

Roaming About

Wireless Connectivity for Mobile PCs

Vinit Nijhawan

With all of the advances in wireless technology, it's no wonder that wireless communications is one of the fastest growing industries on the planet. In this article, Vinit takes a look at some of the various public wireless network technologies available today.

A Well-Lit Sound Check

Build a MIDI Adapter with a LED Indicator

Stuart Ball

Is your soundcard MPU401 or MIDI compatible? If you think it is, but aren't sure what that means, then you definitely want to listen up. Stuart explains some of the details of working with and understanding MIDI, and then shows you how to build a MIDI adapter (complete with LEDs) for a soundcard.

87C51 Programmer Adapter

Noel Rios

Programming the 87C51 is easy when you have a dedicated or universal programmer, but such devices certainly aren't free. That's why Noel used an EPROM programmer, crystal, logic IC, and a few other parts to make this simple, inexpensive adapter for programming the 87C51.

Resource Links

- [System Safety](#)
- [Metglas](#) (Application of Amorphous Metals)

Bob Paddock

Test Your EQ

8 Additional Questions

— COLUMNS —

Learning the Ropes

Design Downloading and Debugging

Ingo Cylix

Last time, Ingo designed a simple bi-colored LED project. Using CPLDs and FPGAs is much easier when you understand how they work, so this month he gives us some information on flash-memory based CPLDs and SRAM-based FPGAs before explaining how to download and debug your design (even if it is just a blinky LED project) in a Xilinx FPGA environment.

Lessons from the Trenches

Embed This PC

Part 3: Emulator to Application Testing

George Martin

Now that George has explained some of the requirements for embedding a '486, it's time to get things going and start testing. Before he's done with Part 3, you'll have a simple design, some startup code for the emulator, and a project.

Silicon Update Online

Analog PLD Anyone?

Tom Cantrell

We're in the middle of the digital revolution, but according to Tom, 1s and 0s will never overthrow the world as we know it. After all, the world as we know it, is analog.

WWW.CIRCUITCELLAR.COM/ONLINE
Table of Contents for March 2000

**INTERNET
PIC[®] 2000
CONTEST**

www.circuitcellar.com/pic2000

Deadline is May 1, 2000

THE ENGINEERS TECH-HELP RESOURCE



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

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


12 **Getting Your Signals Straight**
Walsh-Hadamard Encoders and Decoders in MATLAB
Kizito Tshilumba Kasengulu


20 **Building a RISC System in an FPGA**
Part 2: Pipeline and Control Unit Design
Jan Gray

28 **The Shocking Truth about EMC**
Part 1: Design for Compatibility
George Novacek

52 **A Quick Meter Made**
Fast Frequency Meter for Low Frequencies
Markus Desgronte

58 **All About Correlation**
Ron Tipton

64  **MicroSeries**
Op-Amp Specifications
Part 1: Served Italian Style
Joe DiBartolomeo

70  **From the Bench**
Drop the Incredible Bulk
Using Capacitors as Isolation Components
Jeff Bachiochi

76  **Silicon Update**
A Winter Timer Tale
Tom Cantrell

Task Manager Rob Walker	6
Digital Becomes Analog	
New Product News edited by Harv Weiner	8
Test Your EQ	83
Advertiser's Index May Preview	95
Priority Interrupt Steve Ciarcia	96
All Things Considered	

INSIDE ISSUE 117

EMBEDDED PC

39 **Nouveau PC**
edited by Harv Weiner

42 **RPC Real-Time PCs**
POST It
Build a Power-On Self Test Card for PC/104
Ingo Cyliax

46 **APC Applied PCs**
Under the Covers
Part 1: Get Embed(ded) with Windows NT 4.0
Fred Eady

Digital Becomes Analog



There's no question, we are living in an increasingly digital era. In January, Mark Balch finished up his three-part series on high-definition digital television and brought us up to speed on the analog-to-digital conversion process. Jeff recently finished a series on digital filters in his From The Bench column, then there was Stuart Ball's digital sound board project, and the list goes on.

All of this emphasis on going digital has brought to attention some interesting issues. Namely, that digital engineers are being asked to fill in more and more analog gaps. As Joe DiBartolomeo found out when preparing this month's article on op-amp specifications, when digital designers find themselves faced with analog design issues, they focus on function, not form.

That mentality certainly doesn't make them inferior designers. It's just like taking a standardized test, you don't spend a lot of time on the questions you don't know or understand, you just circle the answer that looks like it might be right and go on. You take your time and make sure you ace the questions in your area of expertise.

Tom's on top of this issue, as you saw in his March Silicon Update Online article on the latest analog PLDs from Lattice Semiconductor. Like he said, the analog gap may be closing, but it certainly won't disappear because humans are analog.

Speaking of humans being analog, I had a chance to make a bit of a digital-to-analog conversion while at the ESC-Spring in Chicago last month. That is, instead of dashing off digital e-mail replies to readers and authors, I got to do some analog handshaking and spend some time talking to previous authors, future authors, and plenty of subscribers.

One of the things that came up while I was talking to different people at the show, was the issue of digital designers having to find their way through analog design. I heard some pretty interesting workarounds that digital designers had come up with. I also heard analog specialists discussing some of the bubble-gum-and-baling-wire solutions they had run across that were "obviously contrived by digital-oriented designers." Both sides have their politics and religion, that's for sure.

But, politics and religion is matter for the trade magazines, *Circuit Cellar* is about quality editorial and practical application articles. At least that's the impression I got from talking to readers while I was in Chicago. I talked to hardware designers who like the schematics, software designers who enjoy the code snippets and software downloads, and instructors who use *Circuit Cellar* in the classroom.

Sure, not everyone who came by the booth was familiar with *Circuit Cellar*, and not everyone who was familiar with us was 100% satisfied with everything we've done. But, I enjoyed talking to those people too, because it gave me an idea of the things we need to work on and improve. Try voicing your opinion to one of the trade magazines and you'll find out that the only input they want from you is green, with plenty of zeros.

I haven't seen any trade magazine demographics, but I've seen the stats on *Circuit Cellar* readers and I'd say that 100% of our readers are human. Until that changes, I guess we'll just keep shaking hands and taking your comments. After all, humans are analog.

rob.walker@circuitcellar.com

CIRCUIT CELLAR®

THE MAGAZINE FOR COMPUTER APPLICATIONS

EDITORIAL DIRECTOR/PUBLISHER

Steve Ciarcia

ASSOCIATE PUBLISHER

Sue Skolnick

MANAGING EDITOR

Rob Walker

CIRCULATION MANAGER

Rose Mansella

TECHNICAL EDITORS

Jennifer Belmonte Rachel Hill
Jennifer Huber

CHIEF FINANCIAL OFFICER

Jeannette Ciarcia

CUSTOMER SERVICE

Elaine Johnston

WEST COAST EDITOR

Tom Cantrell

ART DIRECTOR

KC Zienka

CONTRIBUTING EDITORS

Mike Baptiste Ingo Cyliax
Fred Eady George Martin
Bob Perrin

GRAPHIC DESIGNER

Mary Turek

NEW PRODUCTS EDITOR

Harv Weiner

STAFF ENGINEERS

Jeff Bachiochi John Gorsky

PROJECT EDITORS

Steve Bedford
Janice Hughes
Elizabeth Laurençot
James Soussounis
David Tweed

QUIZ MASTERS

Tak Auyeung Benjamin Day
Bob Perrin

EDITORIAL ADVISORY BOARD

Ingo Cyliax Norman Jackson
David Prutchi

Cover photograph Ron Meadows—Meadows Marketing

PRINTED IN THE UNITED STATES

ADVERTISING

ADVERTISING SALES MANAGER

Bobbi Yush
(860) 872-3064

Fax: (860) 871-0411
E-mail: bobbi.yush@circuitcellar.com

ADVERTISING COORDINATOR

Valerie Luster
(860) 875-2199

Fax: (860) 871-0411
E-mail: val.luster@circuitcellar.com

ADVERTISING CLERK

Sally Collins

CONTACTING CIRCUIT CELLAR

SUBSCRIPTIONS:

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

GENERAL INFORMATION:

TELEPHONE: (860) 875-2199 FAX: (860) 871-0411
INTERNET: info@circuitcellar.com, editor@circuitcellar.com, or www.circuitcellar.com
EDITORIAL OFFICES: Editor, Circuit Cellar, 4 Park St., Vernon, CT 06066

AUTHOR CONTACT:

E-MAIL: Author addresses (when available) included at the end of each article.

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

CIRCUIT CELLAR®, THE MAGAZINE FOR COMPUTER APPLICATIONS (ISSN 1528-0608) and Circuit Cellar Online are published monthly by Circuit Cellar Incorporated, 4 Park Street, Suite 20, Vernon, CT 06066 (860) 875-2751. Periodical rates paid at Vernon, CT and additional offices. One-year (12 issues) subscription rate USA and possessions \$21.95, Canada/Mexico \$31.95, all other countries \$49.95. Two-year (24 issues) subscription rate USA and possessions \$39, Canada/Mexico \$55, all other countries \$85. All subscription orders payable in U.S. funds only via VISA, MasterCard, international postal money order, or check drawn on U.S. bank.

Direct subscription orders and subscription-related questions to Circuit Cellar Subscriptions, P.O. Box 698, Holmes, PA 19043-9613 or call (800) 269-6301.

Postmaster: Send address changes to Circuit Cellar, Circulation Dept., P.O. Box 698, Holmes, PA 19043-9613.

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

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

Entire contents copyright © 2000 by Circuit Cellar Incorporated. All rights reserved. Circuit Cellar and Circuit Cellar INK are registered trademarks of Circuit Cellar Inc. Reproduction of this publication in whole or in part without written consent from Circuit Cellar Inc. is prohibited.

NEW PRODUCT NEWS

Edited by Harv Weiner

PIXEL-COUNTING SENSOR

The **PresencePLUS Pixel-Counting Sensor** is a camera-based sensor that consists of a 512 x 384 CMOS pixel array with a programmable microprocessor, controller, lens, lighting, mounting bracket, and cable. The sensor captures a 256-level grayscale image of a specified area, converts the image to black and white pixels, and compares the designated-color pixel count with user-programmed upper and lower threshold values to render a pass or fail judgment of the target.

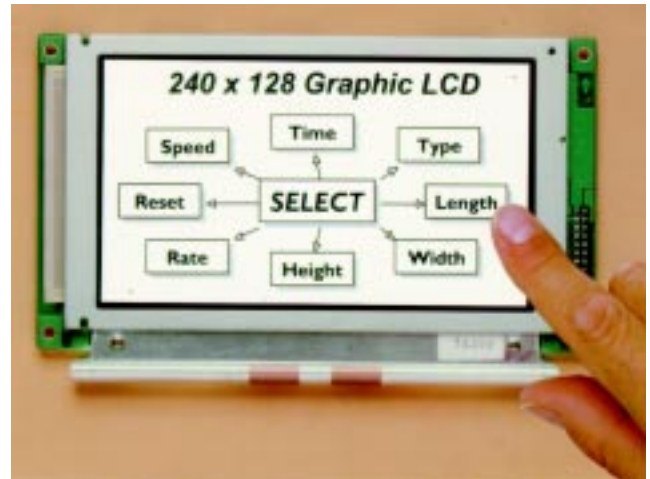
The PRC1 hand-held controller, used to program and monitor the PresencePLUS, displays programming options, monitoring options, and captured images on its LCD screen. Threshold values can be programmed, either manually or automatically, using one of the sensor's two TEACH modes. User programming designates which pixel color to count, white or black; the auto-exposure time, indexed or motion; and the light source behavior; off or strobed.

QUICK START is a one-step SETUP, TEACH, and RUN option for applications that do not require defining a specific region of interest, masking, or teaching bad product. In QUICK START, the pass and fail values are automatically set to 15% above maximum count and 15% below minimum count.

The sensor's three SPST solid-state output contacts may be individually programmed for either NPN (sinking) or PNP (sourcing). Integral circuitry protects the sensor against overload, short circuit, reverse polarity, and transient voltages. The device features an anodized aluminum housing with a painted finish and has an environmental rating of IP20, NEMA 1.

The PresencePLUS sells for under \$1700. Individual sensors list at \$995.

Banner Engineering Corp.
(612) 544-3164 • Fax: (612) 544-3213
www.baneng.com



LCD DISPLAY

Apollo Display Technologies Inc. has introduced a high-contrast line of 240 x 128 pixel monochrome, passive LCD's with onboard controllers and optional touchscreens. The line includes black and white FSTN transmissive and blue and white STN transmissive models, both with cold cathode backlighting. Also available is a transfective, sunlight readable version with LED backlighting.

These compact displays offer designers a simple and low-cost way to display icons, logos, bar charts, multiple fonts, and bit-map graphics with optional user input via an analog resistive touchscreen. Applications include bench top, rack or panel mounted instruments, POS terminals, and other applications requiring high information content in limited space.

The display includes 240 x 128 pixels with 0.50 x 0.50-mm (0.020" x .020") pitch in a 126.0 x 70.0-mm (4.96" x 2.76") viewing area and outline dimensions of 170.0 x 105.0 x 14.0 mm (6.7" x 4.1" x 0.55"). Each model features a 5.0-V, 8-bit ASCII display interface with onboard, industry standard controller T6963C and 64-KB SRAM.

The touch screen is designed for finger, stylus, and signature capture touch requirements. It features an 8-wire interface for accuracy, mounts directly to the display bezel, and offers 79% transmissivity with integral anti-glare film and hardcoat.

The displays price from \$99 and \$79 with and without touchscreen, respectively.

Apollo Display Technologies, Inc.
(631) 654-1143
Fax: (631) 654-1496
www.apollodisplays.com

NEW PRODUCT NEWS

MULTI-AXIS MOTION PROCESSOR

The Navigator **MC2800 Series of Multiple Motor Type Motion Processors** allows the control of both brushed and brushless servomotors in one chipset. The chipset supports sinusoidal commutation of 2- or 3-phase brushless motors, as well as positioning and velocity control of brushed servomotors. Users can select any axis to be brushed or brushless (for 2 and 4 axes). Applications include industrial automation and robotics, medical automation, materials handling, semiconductor manufacturing, test and laboratory equipment, and textiles.

The chipset consists of a 132-pin processor and a 100-pin logic and gives users the ability to off-load resource intensive motion control functions from the application's host processor. Its instruction set supports more than 130 commands to offer flexibility and versatility during application programming. User selectable profiling modes supported by the motion processor include S-curve, trapezoidal, velocity contouring, and electronic gearing. The MC2800 Series accepts input param-



eters such as position, velocity, and acceleration from the host, and generates a corresponding trajectory.

The MC2800 Series has a pre-programmed PID filter with feedforward velocity and acceleration, and a 32-bit position error. The motion processor outputs commutation signals based on either Hall sensors or the motor's encoder. Two or three separate motor signals per axis are sent by the motion processor, either in a 16-bit Digital-to-Analog Converter (DAC) or 10-bit, 20-kHz Pulse Width Modulation (PWM) compatible format. By di-

rectly outputting commutation signals from the chip, the MC2800 reduces torque ripple, oscillation, overshoot, and audible noise level.

The chipset is available in two (MC2820) and four axis (MC2840) configurations with prices starting at \$69 in OEM quantities.

Performance Motion Devices, Inc.
(781) 674-9860
Fax: (781) 674-9861
www.pmdcorp.com

NEW PRODUCT NEWS

SIGNAL PROCESSING SYSTEM

The **Model 6229** is the industry's first digital upconverter and D/A VIM-2 (Velocity Interface Mezzanine) module. It is ideal for radio transmission and can be attached to any VIM-compatible TMS320C6x digital signal processor (DSP) board and digital receiver to build a complete, low-cost transmit and receive software radio system. Applications include radar test signal generation, electronic countermeasures, transmit functions for advanced software radio and satellite communications systems, and synthesized signal generation for test and measurement.

The digital upconverter and D/A module contain two complete upconverter channels. Both channels use the AD9856 Quadrature Digital Upconverter, which includes half-band and CIC interpolation filters, a programmable local oscillator, a complex mixer, and a 12-bit D/A converter.

The VIM modules are daughter cards designed to maximize I/O transfer rates to match the performance and speed of the Pentek 'C6x DSP boards. VIM modules are offered in several different sizes and formats with an extensive array of functionality.

Pentek's ReadyFlow Board Support Libraries of C-callable device functions are supplied with all VIM modules. Third-party and Pentek software development tools are available for a variety of platforms including PC's running Windows 95/98/NT, Digital Alphas running D-UNIX, HP workstations running HP-UX, and Sun SpARCstations running SunOS or Solaris.

Pricing for the Model 6229 starts at \$3,995.

Pentek, Inc.
(201) 818-5900
Fax: (201) 818-5904
www.pentek.com



NEW PRODUCT NEWS

MOTOR CONTROL MODULE

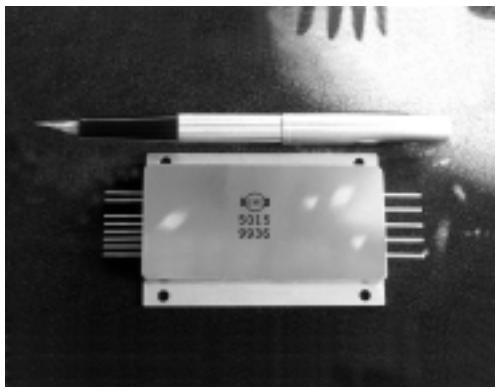
The CMI-5015-12 Motor Control Module is specifically designed for operating both fractional and integral horsepower three-phase motors. The module delivers currents of 30 A, with peaks up to 40 A off a single 4- to 12-V power supply. It is packaged in a miniature, hermetically sealed module, making it ideally suited for high performance battery operated three-phase Brushless Motor Control applications, such as robotics, electromechanical valve assemblies, battery-operated motors, positioning systems, fans, and blower motors. The module is ideally suited for harsh environments where shock, vibration, and extreme temperature are present, and operates over the full military temperature range of -55°C to 125°C.

The CMI-5015-12, plug and play module contains the latest ASIC technology for commutation, cross conduction protection, braking, under voltage lock out protection, temperature compensated reference, and cycle-by-cycle current limiting. Its high-efficient power stage is made up of high and low side drivers for driving the gates of low resistance N Channel power MOSFETS.

Coupled with a unique low loss current sensing circuit and internal bus capacitors, the need for large external components is eliminated. The CMI-5015-12 is designed with 60° electrical sensor phasing, but a 120° factory option is available. An internal DC/DC converter provides logic voltages allowing the module to operate off a single supply as low as 4 V while supplying maximum current to the output stage.

The CMI-5015-12 sells for \$375 per 1,000.

Composite Modules
(508) 226-6969
Fax: (508) 226-0938
www.cmodules.com



8051 DEVELOPMENT SUITE

The **8051 Development Suite** includes a full-featured ANSI C compiler, a relocatable cross-assembler, an advanced overlay linker, a state of the art source level simulator, and a debug monitor. All of these integrate into Crossware's Embedded Development Studio environment to create a single vendor tool set, which can be used to develop and fully debug 8051 programs both with and without target hardware.



The C compiler features smart pointers that free the programmer from the task of telling the compiler where variables are located. It supports memory banking with up to 256 banks of code and 256 banks of RAM. It also features customizable common code merging, allowing the programmer to adjust the size of the program to fit the memory available. In addition, the Embedded Development Studio is able to drive the parallel compilation of all of the source files in a program, allowing the compiler to carry out cross-module optimizations.

The linker also is cross-module aware. As well as performing an overlay analysis to minimize memory usage, it will check the integrity of the entire program ensuring that variables, functions, and other objects are used consistently across modules. The simulator, also known as the 8051 Virtual Workshop, allows full source level debugging without any hardware. It can be rapidly extended to simulate additional devices allowing the simulation of complete target systems.

The debug monitor turns the Virtual Workshop into a full source level remote debugger. The Virtual Workshop communicates with it via a PC serial port and controls execution of the user's program breakpoints, single stepping, and so on. The debug monitor consumes less than 2 KB of program space and easily fits beneath the 0800h address partition of the Dallas DS5000. It consumes no internal ram resources, swapping itself to external ram while the users program is active. Full C source code, using the Virtual Workshop, is supplied together with tools and instructions that allow it to be modified and debugged without hardware.

Crossware
+ 44 (0)1763 853500
Fax: + 44 (0)1763 853330
www.crossware.com

Getting Your Signals Straight

FEATURE ARTICLE

Kizito Tshilumba Kasengulu

Walsh-Hadamard Encoders and Decoders in MATLAB

Not up to speed on Walsh-Hadamard encoders and decoders? No problem, Kizito was kind enough to give us some Hadamard code background before explaining how to implement these helpful encoders and decoders in MATLAB.



named after the French mathematician Jacques Hadamard, Hadamard codes belong to the category of block codes, where the encoder takes the original stream of bits, divides it in short sequences of a certain length, and outputs code words with a content and a length defined by the properties of the code.[1] They have interesting properties, which is why they are used in DSP for communications, image processing, and so forth.

Despite the fact that they can be linear (i.e., each component of the code word is a linear combination of the ones in the message) or non-linear, in most applications only linear codes are used. They are easy to generate and provide simple properties that can be expressed through the algebra of matrix and Hadamard Transforms.

DEFINING A HADAMARD MATRIX

A mathematical definition is provided in this section (a demonstration of the theorem can be found in [2]). First, the definition: a Hadamard matrix of order n is a $n \times n$ matrix of 1s and -1s, such that any pair of distinct rows or columns is orthogonal (i.e., the inner product is zero). And now, the theorem: if a Hadamard matrix of

order n exists, then n is 1, 2, 4, or a multiple of 4.

The matrix H_8 is a Hadamard matrix of order 8 (the character “-” for “-1”). You can verify that any pair of rows or columns is orthogonal.

$$H_8 = \begin{bmatrix} +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 \\ +1 & - & +1 & - & +1 & - & +1 & - \\ +1 & +1 & - & - & +1 & +1 & - & - \\ +1 & - & - & +1 & +1 & - & - & +1 \\ +1 & +1 & +1 & +1 & - & - & - & - \\ +1 & - & +1 & - & - & +1 & - & +1 \\ +1 & +1 & - & - & - & - & +1 & +1 \\ +1 & - & - & +1 & - & +1 & +1 & - \end{bmatrix} \quad [1]$$

Two questions must now be answered. Does the arrangement of rows or columns (because it changes nothing, we will work only with rows for simplicity) make a difference in applications? And, how do we construct a matrix of a certain order (n) in agreement with the above theorem?

The answer to the first question is, yes. The order of rows leads to different matrices having different behaviors in applications. Many arrangements are possible [3], in this article I chose a widely used arrangement called natural order (H_8 is naturally ordered). H_n is in natural order if its rows are not ordered with respect to the number of sign changes in each row.

Therefore, the response to the second question will concern naturally ordered matrix. The most used algorithms that allow the construction of a naturally ordered Hadamard matrix are named after Sylvester and Paley. At this stage, it's important to say that the Sylvester construction technique for a matrix of order $n = 2^m$ (m being a positive integer) always leads to a linear code, while the Paley technique for $n > 8$ always leads to a non-linear code.[2]

Because linear codes are used in this article, the Sylvester technique of order $n = 2^m$ is chosen. Using this

Message	Code word
(1,1,0)	(0,0,1,1,1,1,0,0)
(0,0,1)	(0,1,0,1,0,1,0,1)
(1,0,1)	(0,1,0,1,1,0,1,0)
(0,1,0)	(0,0,1,1,0,0,1,1)

Table 1—The encoder maps the message into a code word according to the Reed-Muller Algorithm.

Sylvester construction technique, the naturally ordered Hadamard matrix of order $n = 2^m$ is obtained through:

$$\begin{aligned} H_1 &= [1] \\ H_2 &= \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \\ H_{2n} &= \begin{bmatrix} H_n & H_n \\ H_n & -H_n \end{bmatrix} \end{aligned} \quad [2]$$

This article will deal with the naturally ordered Hadamard matrix of order $n = 2^m$, this leading to linear binary block codes of length $n = 2^m$ and to their decoding with Fast Hadamard Transforms. All of this is of course in agreement with the above theorem. In closing this section, note that MATLAB software provides the function `hadamard(n)` that allows the creation of a naturally ordered Hadamard matrix of order n .

WALSH FUNCTIONS

In the mathematical literature, two continuous functions $S1(t)$ and $S2(t)$ are said to be orthogonal over t_2-t_1 if:

$$\int_{t_1}^{t_2} S1(t)S2(t) = 0 \quad [3]$$

If we have discrete functions or sequences, the integral can be replaced by a summation, and the orthogonality expresses that the inner product (scalar product) is zero. Among orthogonal continuous func-

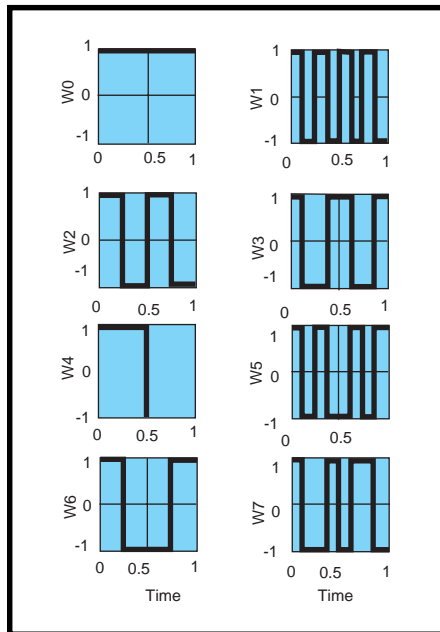


Figure 1—Here are the eight Walsh functions over 0 to 1, arranged in natural order from W_0 to W_7 .

tions, we can mention the well-known $\sin()$ and $\cos()$, Haar functions, Walsh functions, and so on.

Walsh functions were first published in 1924 by J.L. Walsh.[4] They are obtained by using square waveforms to produce the number of desired functions. Figure 1 shows eight Walsh functions over 0 to 1.

The Walsh functions in Figure 1 can be related to the rows of a Hadamard matrix of order 8 (see equation 1) because if you take eight samples (from 1/16 with interval 1/8) of each of the continuous Walsh func-

tions, the resulting sequences are simply rows of the Hadamard matrix, now called Walsh-Hadamard matrix. In other words, a Walsh-Hadamard matrix is a Hadamard matrix where its rows describe samples of Walsh functions.

If the Hadamard matrix is in natural order, as are the Walsh functions, its first row is the sample of W_0 , its second the sample of W_1 , and so on. Note that the matrix of equation 1 is a naturally ordered Walsh-Hadamard matrix, and its rows are noted by index starting from 0 for the first row to $7 = 8 - 1$ for the eighth row.

Naturally ordered Walsh-Hadamard matrices of order n have an index from 0 for the first row to $n - 1$ for the last. As it will be seen later, among the possible codes you can derive from them, you have linear binary block codes of length n . Remember that in this article $n = 2^m$ (m positive integer), in accordance with the Sylvester construction technique and the earlier-mentioned theorem.

FAST WALSH-HADAMARD TRANSFORMS

Just like a Fourier Transform is an expanded summation of the orthogonal functions $\sin()$ and $\cos()$, a Walsh-Hadamard Transform also is a summation of the Walsh functions of a Hadamard matrix. $f(n)$ being a sequence containing $M = 2^m$ real numbers, its naturally ordered Walsh-Hadamard Transform WHT is defined as:

$$WHT(k) = \sum_{n=0}^{M-1} f(n)WH(k,n) \quad [4]$$

where $k = 0,1,2,\dots$, and $M-1$ and WH is the naturally ordered Walsh-Hadamard matrix of order M .

We also can write (equation 4) in the general matrix form:

$$\begin{aligned} WHT &= (WHT_0, WHT_1, \dots, WHT_{2^m-1}) \\ &= f(n)WH_{2^m} \end{aligned} \quad [5]$$

From equation 4, observe that the Walsh-Hadamard Transform makes only sums and subtractions (they will be taken as sums) of the elements of the inputted vector. For an input

Listing 1—This is the code for the implementation of equations 6 and 7.

```
function FWHT_Vector = ...
FWHT_8 ( Input_Vector_8 )
%-----
% function : FWHT_8
% Description : This function executes a Fast Walsh-Hadamard
% Transform with a natural order, this for vectors of length 8.
%-----
H_2=[ 1 -1 ;
      1 -1 ;
      ];
%-- The three stages in FWHT_8
F_8_1=kron(kron(eye(4),H_2),1);
F_8_2=kron(kron(eye(2),H_2),eye(2));
F_8_3=kron(kron(eye(1),H_2),eye(4));
FWHT_Vector_8=Input_Vector_8*F_8_1*...
F_8_2*F_8_3 % performing the FWHT
end
```

vector of length M you have $M - 1$ additions in M steps, that is, $M(M-1)$ operations.

Fast Walsh-Hadamard Transforms allow a gain in time by having M additions in $\log_2 M$ steps (i.e., $M \log_2 M$ operations). Among the proposed algorithms, I chose the one of reference [5] because it leads to the result by using a simple method. This method uses a factorisation of the matrix WH so the application has fewer steps compared to the ordinary case. This factorization is:

$$WH_{2^m} = F_{2^m}^{(1)} F_{2^m}^{(2)} \dots F_{2^m}^{(m)} \quad [6]$$

where:

$$F_{2^m}^i = I_{2^{m-i}} \otimes WH_2 \otimes I_{2^{i-1}} \quad [7]$$

and $i = 1, 2, 3, \dots, m$, I_n is a $n \times n$ identity matrix, WH_2 is the matrix H_2 of equation 2 and multiplied by the Kronecker product. Therefore, for an input vector of length 2^m , you have m steps instead of 2^m . Listing 1 gives a short MATLAB function (*FWHT_8(Input_Vector_8)*) written to calculate the naturally ordered Fast Walsh-Hadamard Transform of length 8 by using the matrix factorisation method described here.

IN THE WORLD OF BITS

After defining the real matrix, in order to pass from the real world to the world of binary, you have the transformation:

$$\begin{aligned} +1 &\rightarrow 0 \\ -1 &\rightarrow 1 \end{aligned} \quad [8]$$

Listings 2 and 3 contain the functions *Conv(Binary_Sequence)* and *BinHM(Hadamard_Matrix)*, which allow you to convert a binary sequence into a real vector, and a real Hadamard matrix into a binary Hadamard matrix, respectively.

Despite the fact that there are many Walsh-Hadamard code definitions, in this article, the codes words are simply the rows of the binary Walsh-Hadamard matrix of order n and their complements.

So, in this article, a Walsh-Hadamard code has a length of n and comprises $2n$ code words. In addition,

Listing 2—This is how a test Walsh-Hadamard Transform is calculated in the real world. This function is used to cross from the binary world to the real world.

```
function Vector = Conv( Binary_Sequence )
%-----
% function : Conv
% Description : This function converts a Binary Sequence into a
%               Vector containing +1s or -1s
%-----
Vector=[];

for i=1:length(Binary_Sequence)
    if ( Binary_Sequence(i)==0 )
        %--- Converting 0 into +1
        Vector=[ Vector 1 ];

    elseif( Binary_Sequence(i)==1 )
        %--- Converting 1 into -1
        Vector=[ Vector -1 ];
    end ;
end;
end
```

this code is said to be non-cyclic because every code word does not have a shifted version inside the code. In other words, saying that the linear codes studied in this article are non-cyclic codes means that, for every code word, you can't find a right or left p -shifted version that also will be a code word.

There are two other code definitions for Walsh-Hadamard codes.[2] The first code is comprised of the rows of the binary Walsh-Hadamard matrix of order n left-most column (then this code has a length of $n - 1$ and comprises n code words). The second code is comprised of this first code along with the complements of the rows (then, here the code has a length of $n - 1$, but with $2n$ code words). The code definition here is commonly used and is sufficient for a good understanding of the subject.

Also, keep in mind that the Hamming distance (d) between two sequences of equal length is the number of coordinates in which they differ. The minimum distance (d_{\min}) of a block code is the minimum Hamming distance between all distinct pairs of code words.

A code with d_{\min} can detect all error patterns of weight (the number of nonzero coordinates in a sequence) less than or equal to, $d_{\min} - 1$. It can correct all error patterns of weight less than or equal to:

$$\text{floor_function}\left(\frac{d_{\min} - 1}{2}\right) \quad [9]$$

This correction capability must be taken as minimal because it can be proven that some block codes have a higher capability. You can verify that this Walsh-Hadamard code of length $n = 2^m$ ($2n$ codes words of length n) has a minimum distance of:

$$\frac{n}{2}$$

Assuming code words have the same probability to be transmitted, a maximum likelihood hard decision decoder (MLHDD) outputs the code word closest in Hamming distance to the received sequence.

Because the Hamming distance between a code word and its complement always has a value of n , encoders and decoders will work only with the n code words that are defined by the rows of the binary Walsh-Hadamard matrix. The MLHDD will output a code word having a minimal Hamming distance of d_x to the received sequence and $n + d_x$ between the complement of the code word and the received sequence.

WALSH-HADAMARD ENCODERS

The goal of the encoder is to map a message sequence into a code word of $n = 2^m$ bits. The reader must remember that this code word is a row or a

Listing 3—A MATLAB function is to convert a Hadamard matrix into a binary Hadamard matrix.

```
function Binary_Hadamard_Matrix = ...
BinHM ( Hadamard_Matrix )
%-----
% function : BinHM
% Description : This function converts a Hadamard Matrix into a
  Binary Hadamard Matrix
%-----
Binary_Hadamard_Matrix=ones(size(...
Hadamard_Matrix)); % initialization
Number_rows = size(Hadamard_Matrix,1);
Number_columns = size(Hadamard_Matrix,2);

for i=1: Number_rows
  for j=1: Number_columns
    if ( Hadamard_Matrix(i,j)==+1 )
      % Converting +1 into 0
      Binary_Hadamard_Matrix(i,j)=0;
    elseif( Hadamard_Matrix(i,j)==-1 )
      % Converting -1 into 1
      Binary_Hadamard_Matrix(i,j)=1;
    end ;
  end;
end;
end
```

complement of a row in the binary Walsh-Hadamard matrix of order $n = 2^m$. As is often the case, the encoder I used here will select only among the rows of the matrix. A message of length m will be encoded into a code word of length $n = 2^m$.

Among the algorithm used for a message of length m to select a Walsh-Hadamard code word of length $n = 2^m$, the Reed-Muller encoding algorithm is commonly used to give an efficient and easy-to-implement technique.

Because the goal here is not to study Reed-Muller codes, only the main steps are given (note that Reed-Muller codes are linear block codes). [6] The generator matrix for the first order Reed-Muller code $R(1,3)$ of length $8 = 2^3$ is defined as:

$$G = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} U \\ V_3 \\ V_2 \\ V_1 \end{bmatrix} \quad [10]$$

The above Reed-Muller matrix is comprised of a vector U of 1s in its first row and the remaining rows are constructed so that they describe a digital counter from left to right. This method can be generalized to obtain a greater matrix generator for first order Reed-Muller codes $R(1,m)$.

Considering a message $(m_0, m_1, m_2, \dots, m_{m-1})$ has to pass through an encoder, Reed-Muller codes and Walsh-Hadamard codes are related so the chosen code word c is:

$$c = m_{m-1}V_1 + m_{m-2}V_2 + \dots m_1V_{m-1} + m_0V_m \quad [11]$$

Therefore, the result obtained by the above linear combination of the V_i vectors is a row in the binary Walsh-Hadamard matrix of order $n = 2^m$.

Another way to avoid the direct use of Reed-Muller generator matrix is to observe that the index (remember that indexes start from zero) of the chosen row has a binary or radix-2 form $m_0m_1m_2\dots m_{m-1}$. Consequently, for a message $(m_0, m_1, m_2, \dots, m_{m-1})$, the index of the row to choose is the decimal form of $m_0m_1m_2\dots m_{m-1}$. It is important to note that this affirmation is true in the case of a naturally ordered matrix.

Listing 4 is a MATLAB program that implements a Walsh-Hadamard Encoder of length 8. This encoder use a first order Reed-Muller encoding algorithm. In the section Simulation Results, you can see that for a message $(0,0,1)$, the selected code word is $(0,1,0,1,0,1,0,1)$. You can verify that the index of this row is the decimal

form of 001 (i.e., 1). In the same way, the code word for the message (1,1,0) is the row having the index value 6 (6 is the decimal form of 110).

MAXIMUM LIKELIHOOD DECODERS

This article deals with hard decision decoding (i.e., the decoding that gives the closet code word in Hamming distance). By observing equation 4, we find that for each k , the right side of the equation is simply the correlation between the vectors $f(n)$ and $WH(k,:)$, meaning that the Hadamard Transform performs a correlation between the received sequence ($f(n)$) and all possible code words inside the matrix.

Under the assumption of equal probability of transmission, a Maximum Likelihood Decoder is related to a correlation decoder, because the code word that minimizes the Hamming distance maximizes the correlation (assuming that this correlation is positive).

So, the maximum likelihood code word is the row in WH that corresponds to the greatest value in the Fast Walsh-Hadamard Transform vector. If this value is negative, this row is simply the furthest code word, then its complement (addition with a sequence of 1s) is the maximum likelihood code word.

In relation to what I said earlier, you now see that it's sufficient to compute correlations with only m code words (the m code words inside WH). In brief, the encoder selects one of the m rows of WH , while the decoder outputs one of the $2m$ code words by using only the m rows of WH in its calculus.

LIKELIHOOD DECODER IN MATLAB

Listing 5 is a MATLAB implementation for the case of a code of length 8 ($16 = 8 \times 2$ code words). The function `FWHT_8(Input_Vector_8)` computes the Fast Walsh-Hadamard Transform with natural order, as described earlier, and puts the result in the vector `FWHT_Vector`. The MATLAB function `hadamard(n)` has been used for the creation of the naturally ordered Walsh-Hadamard matrix of order 8 (see the comments for more details). In the Simulation Results section, you can see that, for a corrupted received sequence (1,0,0,0,0,0,1,1), the maximum likelihood code word is (1,1,0,0,0,0,1,1).

SIMULATION RESULTS

This section gives some simulation results of the running of our MATLAB code and allows you to verify all of the theories I have covered. For clar-

Listing 4—The input message is taken from the user prompt and the code word is calculated by using the Reed-Muller Algorithm.

```
%-----
% Program : WH_Encoder_Length_8
% Description : This program implements a Walsh-Hadamard Encoder
% based on the first order Reed-Muller R(1,3)Encoding Algorithm.
% Author : Kizito Tshilumba Kasengulu
%-----
clear
m=input('Input the binary message (length 3) : ');
%--- The Reed-Muller Generator Matrix
Reed_Muller=[
    1 1 1 1 1 1 1 ;
    0 0 0 0 1 1 1 ;
    0 0 1 1 0 0 1 ;
    0 1 0 1 0 1 0 ;
];
%---The encoder outputs a code word c of length 8.
%---xor(x,y) is the Exclusive OR Operator acting as the binary
% addition of x and y
c=xor(xor(m(3)*Reed_Muller(4,:),...
m(2)*Reed_Muller(3,:)),...
m(1)*Reed_Muller(2,:))
```

Received sequence	Maximum likelihood code word
(0,1,0,1,0,1,0,0)	(0,1,0,1,0,1,0,1)
(1,0,0,0,0,0,1,1)	(1,1,0,0,0,0,1,1)
(0,0,1,1,1,1,1,1)*	(1,1,1,1,1,1,1,1)*
(1,1,0,0,0,0,0,0)*	(0,0,0,0,0,0,0,0)*

Table 2—As a result of noise, the received sequence is a corrupted version of the transmitted one. The decoder outputs the code word that is the most likely to have been transmitted.

ity, I chose to present simulations for the encoder and the decoder separately. Table 1 shows some simulations for the encoder and Table 2 shows the results of decoder simulations.

So, the decoding of a sequence and its complement agree with the theory

because the outputs are complementary (illustrated by the asterisks in Table 2).

END SIGNAL

This article has proven that it is possible to derive non-cyclic linear block codes from Walsh-Hadamard matrix. The derivation lead to codes with length $n = 2^m$ and $2n$ code words. Simulations with the MATLAB programs allow you to observe the working of encoders and decoders. It's also possible to verify the characteristics of the code mentioned in the text. In spite of the fact that the MATLAB programming was limited to $n = 8$ and $m = 3$, they can be generalized for any values of n and m with $n = 2^m$. ☐

Kizito Tshilumba Kasengulu is a TDMA software engineer at NSI Communications in Montreal, Canada. You may reach him at kkasengulu@nsicomm.com.

REFERENCES

- [1] J. Hadamard, *Résolution d'une Question Relative aux Determinants*, Bulletin on Science and Mathematics, vol. 17, pp. 240- 248, 1893.
- [2] F.J. MacWilliams and N.J.A. Sloane, *The Theory of Error Correcting Codes*, Amsterdam, North-Holland, 1977.
- [3] Y.A. Geadeh and M.J.G. Corinthios, *Natural, Dyadic, and Sequency Order Algorithms and Processors for the Walsh-Hadamard Transform*, IEEE Transactions on Computers; vol. C-26, pp. 435-442, May 1977.
- [4] J.L. Walsh, "A Closed Set of Orthogonal Functions", Am J. Math., vol. 55, pp. 5-24, January 1923.
- [5] M.H. Lee and M. Kaveh, *Fast Hadamard Transform Based on a Simple Matrix Factorization*, IEEE Transactions on Acoustic, Speech, Signal Processing; vol. ASSP-34, no. 6, pp. 1666-1667, 1986.
- [6] S.B. Wicker, *Error Control Systems for Digital Communications and Storage*, Prentice Hall, Englewood Cliffs, NJ, 1995.

Listing 5—This decoder takes the received sequence (from the point of view of the decoder) from a user prompt. It outputs the code word that is the closest in Hamming distance.

```
%-----
% Program : WH_Decoder_Length_8
% Description : This program implements a Walsh-Hadamard Maximum
Likelihood Decoder based on a naturally ordered Fast Walsh-
Hadamard Transform. This decoder is designed for codes of
length 8 and use Hard Decision.
% Author : Kizito Tshilumba Kasengulu
%-----

clear
Received_Sequence=input('Input the ...
received sequence (length 8) : ')
%--Converting the Received_Sequence ( 1s or 0s ) into a Vector
(+1s or -1s) in using our function Conv (see listing 4)
Received_Vector=Conv(Received_Sequence) ;
%--Performing the naturally ordered Fast Walsh-Hadamard Trans-
form of the Received_Vector of length 8 by using our function
FWHT_8 ( see listing 3 )
FWHT_Vector = FWHT_8( Received_Vector ) ;

%-- Calculating the greatest magnitude in FWHT_Vector and re-
taining its coordinate
[Greatest,Coordinate]=max(...
abs(FWHT_Vector ));
%-- Creating a Hadamard matrix of order 8 by calling the Matlab
function hadamard(8) and converting it into a Binary Hadamard
Matrix by using our function BinHM
Binary_Hadamard_Matrix_8=BinHM(...
hadamard(8));
%-- Calculating the Maximum Likelihood Code Word
if ( FHT_Vector(Coordinate) ) > 0
Maximum_Likelihood_Code_Word=...
Binary_Hadamard_Matrix_8(Coordinate,:);
else
%-- Sum of the two vectors by using xor(x,y)
Maximum_Likelihood_Code_Word=xor(...
ones(1,8),Binary_Hadamard_Matrix_8(...
Coordinate,:))
end ;
```

SOURCE

MATLAB

The MathWorks, Inc.
(508) 647-7000
Fax: (508) 647-7001
www.mathworks.com

HAVE YOU HEARD???
CIRCUIT CELLAR
ONLINE IS COMING
TO CD-ROM!

WATCH WWW.CIRCUITCELLAR.COM FOR DETAILS

Building a RISC System in an FPGA

FEATURE ARTICLE

Jan Gray

Part 2: Pipeline and Control Unit Design

In Part 1, Jan introduced his plan to build a pipelined 16-bit RISC processor and System-on-a-Chip in an FPGA. This month, he explores the CPU pipeline and designs the control unit. Listen up, because next month, he'll tie it all together.



Last month, I discussed the instruction set and the datapath of an xr16 16-bit RISC processor. Now, I'll explain how the control unit pushes the datapath's buttons.

Figure 2 in Part 1 (*Circuit Cellar*, 116) showed the CTRL16 control unit schematic symbol in context. Inputs include the RDY signal from the memory controller, the next instruction word $INSN_{15:0}$ from memory, and the zero, negative, carry, and overflow outputs from the datapath.

The control unit outputs manage the datapath. These outputs include pipeline control clock enables, register and operand selectors, ALU controls, and result multiplexer output enables. Before designing the control circuitry, first consider how the pipeline behaves in both good and bad times.

PIPELINED EXECUTION

To increase instruction throughput, the xr16 has a three-stage pipeline—instruction fetch (IF), decode and operand fetch (DC), and execute (EX).

In the IF stage, it reads memory at the current PC address, captures the resulting instruction word in the instruction register IR, and incre-

ments PC for the next cycle. In the DC stage, the instruction is decoded, and its operands are read from the register file or extracted from an immediate field in the IR. In the EX stage, the function units act upon the operands. One result is driven through three-state buffers onto the result bus and is written back into the register file as the cycle ends.

Consider executing a series of instructions, assume no memory wait states. In every pipeline cycle, fetch a new instruction and write back its result two cycles later. You simultaneously prepare the next instruction address $PC+2$, fetch instruction I_{PC} , decode instruction I_{PC+2} , and execute instruction I_{PC+4} .

Table 1 shows a normal pipelined execution of four instructions. That's the simple case, but there are several pipeline complications to consider—data hazards, memory wait states, load/store instructions, jumps and branches, interrupts, and direct memory access (DMA).

What happens when an instruction uses the result of the preceding instruction?

I_1 : andi r1,7
 I_2 : addi r2,r1,1

Referring to time t_3 of Table 1, EX_1 computes $r1=r1\&7$, while DC_2 fetches the old value of r1. In t_4 , EX_2 incorrectly adds 1 to this stale r1.

This is a data hazard, and there are several ways to address it. The assembler can reorder instructions or insert nops to avoid the problem. Or, the control unit can detect the hazard and stall the pipeline one cycle, in order to write-back the result to the register file before fetching it as a source register. However, these techniques hurt performance.

t_1	t_2	t_3	t_4	t_5
IF₁	DC₁	EX₁	EX₂	EX₃
	IF₂	DC₂	DC₃	DC₄
		IF₃	IF₄	

Table 1—Here the processor fetches instruction I_1 at time t_1 and computes its result in t_3 , while I_2 starts in t_2 and ends in t_4 . Memory accesses are in boldface.

Listing 1—This C code produces assembly code that includes a load I_L and a branch I_B . Each causes pipeline headaches.

```

if ((p->flags & 7) == 1)
    p->x = p->y;

I :    lw r6,2(r10)    ;load r6 with p->flags
IL :    andi r6,7      ;is (p->flags & 7)
I2 :    addi r0,r6,-1  ;==1?
I3 :    bne T
IB :    lw r6,6(r10)   ;yes: load r6 with p->y
5 :    ...

```

Instead, you do result forwarding, also known as register file bypass. The datapath DC stage includes FWD, a 16-bit 2-1 multiplexer (mux) of AREG (register file port A), and the result bus. Most of the time, FWD passes AREG to the A operand register, but when the control unit detects the hazard (DC source register equals EX destination register), it asserts its FWD output signal, and the A register receives the I_1 result just in time for EX_2 in t_4 .

Unlike most pipelined CPUs, the xr16 only forwards results to the A operand—a speed/area tradeoff. The assembler handles any rare port B data hazards by swapping A and B operands, if possible, or inserting nops if not.

MEMORY ACCESSES

The processor has a single memory port for reading instructions and loading and storing data. Most memory accesses are for fetching instructions. The processor is also the DMA engine, and a video refresh DMA cycle occurs once every eight clocks or so. Therefore, in any given clock cycle, the processor executes either an instruction fetch memory cycle, a DMA memory cycle, or a load/store memory cycle.

Memory transactions are pipelined. In each memory cycle, the processor drives the next memory cycle's address and control signals and awaits RDY, indicating the access has been completed. So, what happens when memory is not ready?

The simplest thing to do is to stop the pipeline for that cycle. CTRL deasserts all pipeline register clock enables PCE, ACE, and so forth. The

pipeline registers do not clock, and this extends all pipeline stages by one cycle. In Table 2, memory is not ready during the fetch of instruction I_3 in t_3 , and so t_4 repeats t_3 . (Repeated pipe stages are italicized.)

I_L in Listing 1 is a load word instruction. Loads and stores need a second memory access, causing pipeline havoc (see Table 3). In t_4 you must run a load data access instead of an instruction fetch. You must stall the pipeline to squeeze in this access.

Then, although you fetched I_3 in t_3 , you must not latch it into the instruction register (IR) as t_3 ends, because neither EX_L nor DC_2 are finished at this point. In particular, DC_2 must await the load result in order to forward it to A, because I_2 uses r6—the result of I_L !

Finally, if (in t_3) you don't save the just-fetched I_3 somewhere, you'll lose it, because in t_4 the memory port is busy with the load cycle. If you lose it, you'll have to re-fetch it no sooner than t_5 , with the result that even a no-wait load requires three cycles, which is unacceptable.

To fix this problem, the control unit has a 16-bit NEXTIR register and an IR source multiplexer (IRMUX). In t_3 , it captures I_3 in NEXTIR, and then in t_4 IR is loaded from NEXTIR instead of from the memory port (which is busy with the load). NEXTIR ensures a two-cycle load or store, at a cost of eight CLBs.

As with instruction fetch accesses, load/store memory accesses may have to wait on slow memory. For example, had RDY not been asserted during t_4 , the pipeline would have

stalled another cycle to wait for EX_L access to complete.

BRANCHING OUT

Next, consider the effect of jumps (call and jal) and taken branches. By the time you execute the jump or taken branch I_j during EX_j (updating PC), you'll have decoded I_{j+1} and fetched I_{j+2} . These instructions in the branch shadow (and their side effects) must be annulled.

Continuing the Table 3 example from time t_5 , and assuming the branch is taken at t_7 , you must annul the EX_5 stage of I_5 , and the DC_6 and EX_6 stages of I_6 . (Annulled stages are struck through). Execution continues at instruction I_T . T_9 is not an EX_5 load cycle, because the I_5 load is annulled.

Because you always annul the two branch shadow instructions, jumps and taken branches take three cycles. Jumps also save the return address in the destination register. This return address is obtained from the datapath's RET register, which holds the address of the instruction in the DC pipeline stage.

INTERRUPTS

When an interrupt request occurs, you must jump to the interrupt handler, preserve the interrupt return address, retire the current pipeline, execute the handler, and later return to the interrupted instruction.

When INTREQ is asserted, you simply override the fetched instruction with *int*, that is, `jal r14,10(r0)` via the IRMUX. This jumps to the interrupt handler at 0x0010 and leaves the return address in r14, which is reserved for this purpose. When the handler has completed, it executes *iret*, (i.e., `jal r0,0(r14)`) and execution resumes with the interrupted instruction.

t_1	t_2	t_3	t_4	t_5
IF ₁	DC ₁	EX ₁	EX ₁	EX ₂
	IF ₂	DC ₂	DC ₂	DC ₃
		IF ₃	IF ₃	IF ₄

Table 2—During t_3 , the instruction fetch memory access of I_3 is not RDY, so the pipeline registers do not clock, and the pipeline stalls until RDY is asserted in t_4 . Repeated pipeline stages are italicized.

t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
IF_L	DC_L	EX_L	EX_L					
	IF_2	DC_2	DC_2	EX_2				
		IF_3	IF_3	DC_3	EX_3			
			IF_B	DC_B	EX_B			
				IF_5	DC_5	EX_5		
					IF_6	DC_6	EX_6	
						IF_T	DC_T	

Table 3—Pipelined execution of the load instruction I_L , I_2 , I_3 , the branch I_B , the annulled I_5 and I_6 , and the branch target I_T . During t_4 you stall the pipeline for the I_L load/store memory cycle. The branch I_B executed in t_7 causes I_5 and I_6 to be annulled in t_8 and t_9 . Annulled instructions are struck through.

There are two pipeline issues here. First, you must not interrupt an interlocked instruction sequence (any add, sub, shift, or imm followed by another instruction). If an interlocked instruction is in the DC stage, the interrupt is deferred one cycle.

Secondly, the int must not be inserted in a branch or jump shadow, lest it be annulled. If a branch or jump is in the DC stage, or if a taken branch or jump is in the EX stage, the interrupt is deferred.

The simplicity of the process pays off once again. The time to take an interrupt and then return from a null interrupt handler is only six cycles.

You might be wondering about the interrupt priorities, non-maskable interrupts, nested interrupts, and interrupt vectors. These artifacts of the fixed-pinout era need not be hardwired into our FPGA CPU.

They are best done by collaboration with an on-chip interrupt controller and the interrupt handler software.

The last pipeline issue is DMA. The PC/address unit doubles as a DMA engine. Using a 16×16 RAM as a PC register file, you can fetch either an instruction ($AN \leftarrow PC_0 \pm 2$) or a DMA word ($AN \leftarrow PC_1 \pm 2$) per memory cycle.

After an instruction is fetched, if DMAREQ has been asserted, you insert one DMA memory cycle.

This PC register file costs eight CLBs for the RAM, but saves 16 CLBs (otherwise necessary for a separate 16-bit DMA address counter and a 16-bit 2-1 address mux), and shaves a couple of nanoseconds from the system's critical path. It's a nice example of a problem-specific optimization you can build with a customizable processor.

To recap, each instruction takes three pipeline cycles to move through the instruction fetch, operand fetch and decode, and execute pipeline stages. Each pipeline cycle requires up to three memory access cycles (mandatory instruction fetch, optional DMA, and optional EX stage load or store). Each memory access cycle requires one or more clock cycles.

CONTROL UNIT DESIGN

Now that you understand the pipeline, you are ready to design the control unit. (For more information on RISC pipelines, see *Computer Organization and Design: The Hardware/*

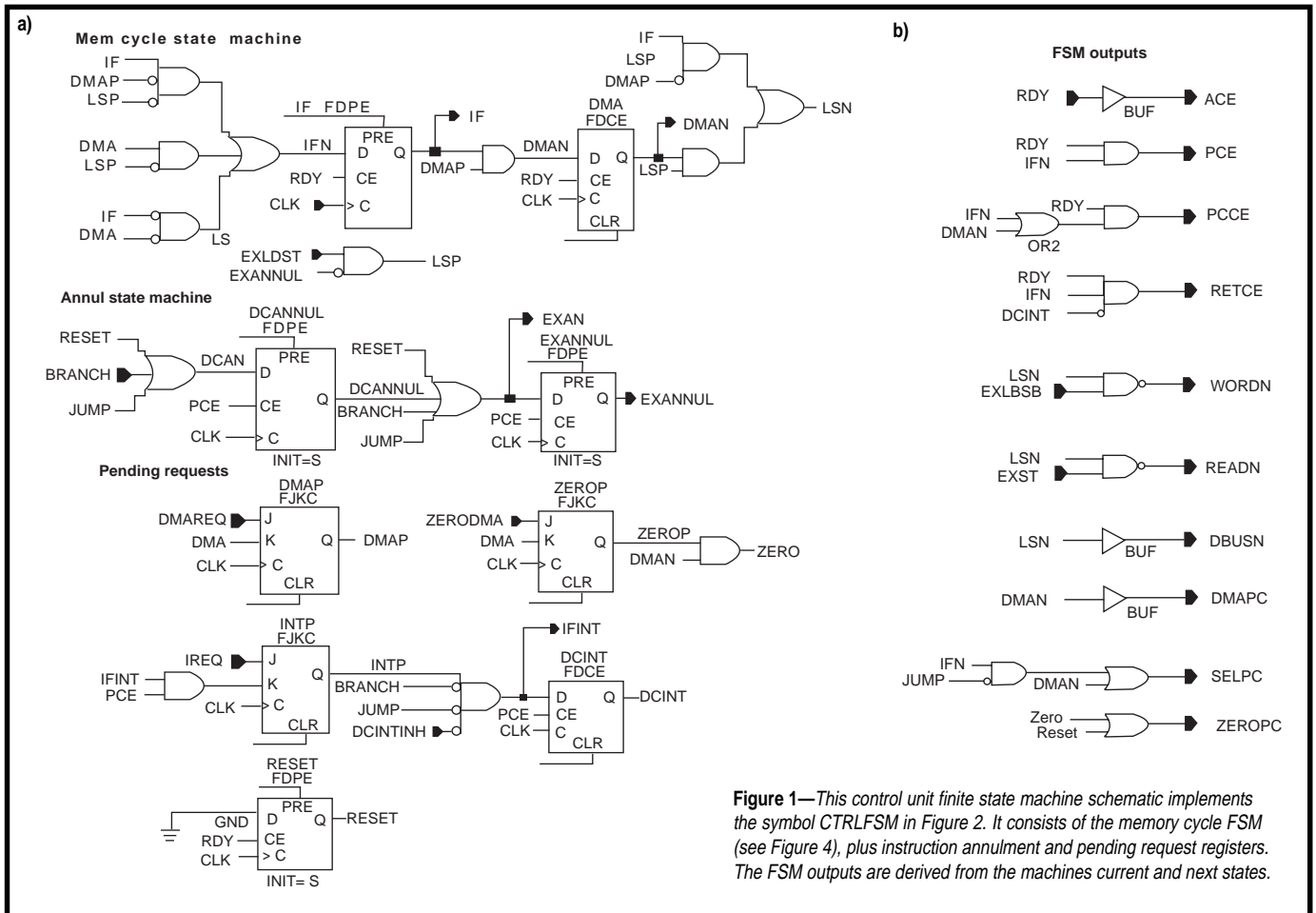


Figure 1—This control unit finite state machine schematic implements the symbol CTRLFSM in Figure 2. It consists of the memory cycle FSM (see Figure 4), plus instruction annulment and pending request registers. The FSM outputs are derived from the machines current and next states.

RNA	When
RA	DC: add sub addi lw lb sw sb jal
RD	DC: all rr, ri format
0	DC: call
EXRD	EX: all but call
15	EX: call

RNB	When
RB	DC: add sub, all rr fmt
RD	DC: sw sb
EXRD	EX: all but call
15	EX: call

Table 4—RNA and RNB control the A and B ports of the register file. While CLK is high, they select which registers to read, based upon register fields of the instruction in the DC stage. While CLK is low, they select which register to write, based upon the instruction in the EX stage.

Software Interface, by Patterson and Hennessy.) [1] First, some important naming conventions. Some control unit signal names have prefixes and suffixes to recognize their function or context (most signal names sans prefix are DC stage signals):

- Nsig: not signal—signal inverted
- DCsig: a DC stage signal
- EXsig: an EX stage signal
- sigN: signal in “next cycle”—input to a flip-flop whose output is sig
- sigCE: flip-flop clock enable
- sigT: active low 3-state buffer output enable

Each instruction flows through the three stages (IF, DC, and EX) of the control unit (see Figure 2) pipeline. In the IF stage, when the instruction fetch read completes, the new instruction at $INSN_{15:0}$ is latched into IR.

In the DC stage, DECODE decodes IR to derive internal control signals. In the first half clock cycle, CTRL drives $RNA_{3:0}$ and $RNB_{3:0}$ with the source registers to read, and drives FWD and $IMM_{5:0}$ to select the A and B operands. If the instruction is a branch, CTRL determines if it is taken. Then as the pipeline advances, the instruction passes into EXIR.

In the EX stage, CTRL drives ALU and result mux controls. If the instruction is a load/store, it inserts a

memory access. In the last half cycle, RNA and RNB both drive the destination register number to store the result into the register file.

Let’s consider each part of the control finite state machine (see Figure 1). The control FSM has three states:

- IF: current memory access is an instruction fetch cycle
- DMA: current access is a DMA cycle
- LS: current access is a load/store

Figure 4 shows the state transition diagram. The FSM clocks when one memory transaction completes and another begins (on RDY). CTRLFSM also has several other bits of state:

- DCANNUL: annul DC stage
- EXANNUL: annul EX stage
- DCINT: int in DC stage
- DMAP: DMA transfer pending
- INTP: interrupt pending

DCANNUL and EXANNUL are set after executing a jump or taken branch. They suppress any effects of

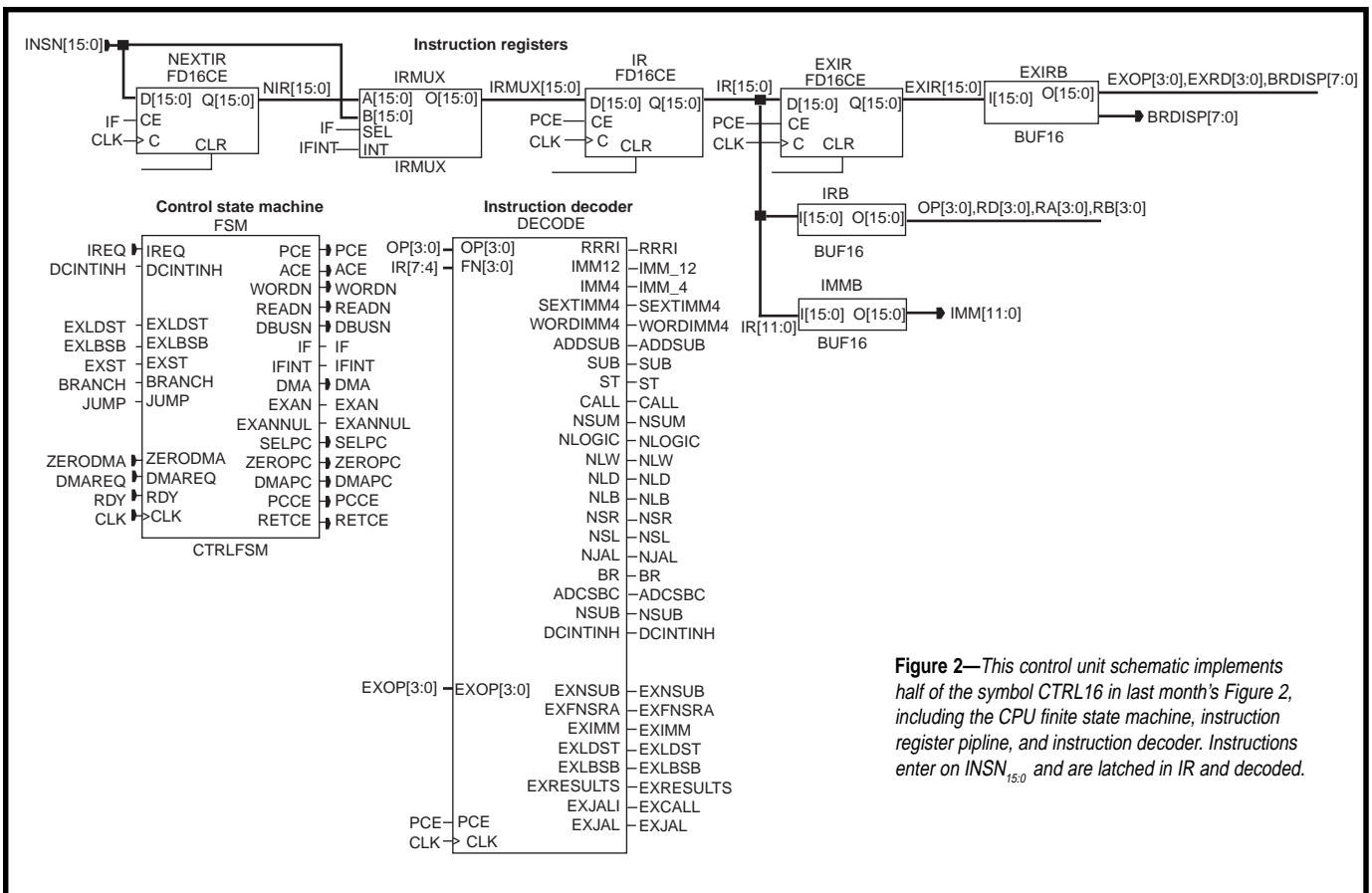


Figure 2—This control unit schematic implements half of the symbol CTRL16 in last month’s Figure 2, including the CPU finite state machine, instruction register pipeline, and instruction decoder. Instructions enter on $INSN_{15:0}$ and are latched in IR and decoded.

the two instructions in the branch shadow, including register file write-back and load/store memory accesses. So, an annulled add still fetches and adds its operands, but its results are not retired to the register file.

DCINT is set in the pipeline cycle following the insertion of the `int` instruction. It inhibits clocking of RET for one cycle, so that the `int` picks up the return address of the interrupted instruction rather than the instruction after that.

The highest fan-out control signal is PCE, the pipeline clock enable. Most datapath registers are enabled by PCE. It indicates that all pipe stages are ready and the pipeline can advance. PCE is asserted when RDY signals completion of the last memory cycle in the current pipeline cycle. If memory isn't ready, PCE isn't asserted, and the pipeline stalls for one cycle.

The control FSM also takes care of managing the memory interface via the following signals:

Enable	Instruction	Source
SUMT	add sub addi adc sbc addi sbci	SUM _{15:0}
LOGICT	and or xor andn andi ori xori andni	LOGIC _{15:0}
SLT	slli	A _{14:0} 0
SRT	srlr srar	SRI A _{15:1}
ZXT	lb	0 _{15:8}
RETADT	jal call	RETAD _{15:0}
none	sw sb br* imm	—

Table 5—Here's a look at the result multiplexer output enable controls. The instruction determines which enable is asserted and which function unit drives RESULT_{15:0}

- RDY: memory cycle complete (input from the memory controller)
- READN: next memory cycle is a read transaction—true except for stores
- WORDN: next cycle is 16-bit data—true except for byte loads/stores
- DBUSN: next cycle is a load/store, and it needs the on-chip data bus
- ACE (address clock enable): the next address AN_{15:0} (a datapath output) and the above control outputs are all valid, so start a new memory transaction in the next clock cycle.

ACE equals RDY, because if memory is ready, the CPU is always eager to start another memory transaction.

There are no IF stage control outputs. Internal to the control unit, three signals control IF stage resources. Those three signals are:

- PCE: enable IR and EXIR clocking
- IF: asserted in an instruction fetch memory cycle
- IFINT: force the next instruction to be `int = jal r14,10(r0) = 0xAE01`

If a DMA or load/store access is pending, IF enables NEXTIR to capture the previously fetched instruction (take a look back at time t_3 in Table 3). Otherwise, the instruction fetch is the only memory access in the pipe stage. So, IF is then asserted with PCE, and IRMUX selects the INSN_{15:0} input as the next instruction to complete.

DECODE STAGE

The greater part of the control unit operates in the DC stage. It must decode the new instruction, control the register file, the A and B operand multiplexers, and prepare most EX stage control signals.

The instruction register IR latches the new instruction word as the DC stage begins. The buffers IRB and IMMB break out the instruction fields OP, RD, and so forth—IR_{15:12} is re-named OP_{3:0} and so on (the tools optimize away these buffers).

The instruction decoder DECODE is simple. It is a set of 30 ROM 16x1s, gate expressions, and a handful of flip-flops. Each ROM inputs OP_{3:0} or EXOP_{3:0} and outputs some decoded signal. The decoder is relatively compact because

xl16 has a simple instruction set, and its 4-bit opcodes are a good match for the FPGA's 4 LUTs.

The register file control signals, shared by both the DC and EX stages, are RNA_{3:0}: port A register number; RNB_{3:0}: port B register number; and RFE: register file write enable.

With CLK high, CTRL drives RNA and RNB with the DC stage instruction's source register numbers. With CLK low, CTRL drives RNA and RNB with the EX stage destination register number.

RFE is asserted with PCE when there is a result to write back. It is false for instructions, which produces no result (immediate prefix, branch, or store) for annulled instructions, and for destination r0.

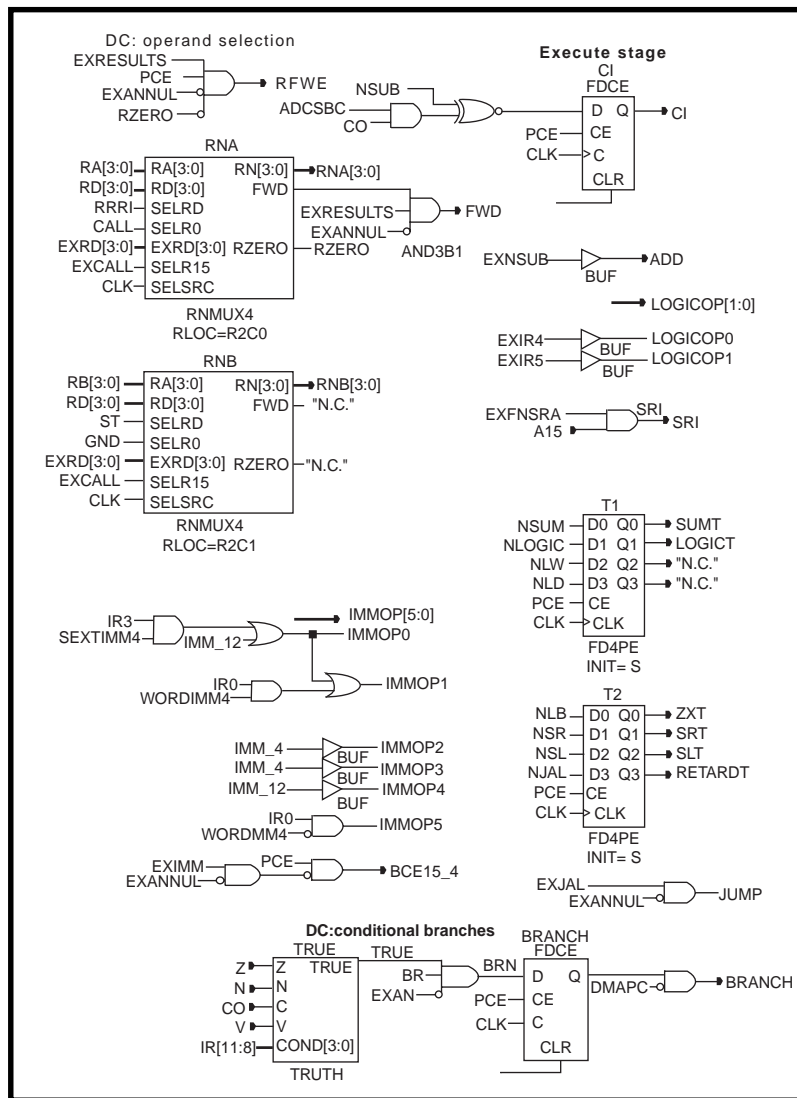


Figure 3—The remainder of the control unit schematic implements the DC stage operand selection logic including register file, immediate operand control, branch logic, EX stage ALU, and result mux controls.

The muxes RNA and RNB produce RNA_{3:0} and RNB_{3:0}, as shown in Table 4, as selected by decode outputs RRR1, CALL, ST, EXCALL, and CLK. Call is irregular. It computes r15 = pc, pc = r0 + imm12<<4, and the registers r15 and r0 are implicit.

The FWD signal causes RESULT to be forwarded into A, overriding AREG. CTRL asserts FWD when the EX stage destination register equals the DC stage source register A (detected within RNA), unless the EX stage instruction is annulled or its destination is r0.

Last month, I discussed IMMED, the BREG/immediate operand mux. IMMOP_{5:0} controls IMMED, based upon the decoder outputs WORDIMM, SEXTIMM4, IMM_12, and IMM_4.

B_{3:0} is clock enabled on PCE, but B_{15:4} uses B15_4CE. B15_4CE is PCE, unless the EX stage instruction is imm. Thus, the imm prefix establishes B_{15:4}, and the subsequent immediate operand instruction provides B_{3:0} only.

Now, turning to conditional branches, if the DC stage instruction is a branch, then the EX stage instruction must be add, sub, or addi, which drives the control unit's condition inputs Z (zero), N (negative), CO (carry-out), and V (overflow).

Late in the DC stage, the TRUE macro evaluates whether or not the branch condition COND is true with respect to the condition inputs. If so, and if the branch instruction is not annulled, the BRANCH flip-flop is set. Therefore, as the pipeline advances and the branch instruction

enters the EX stage, the BRANCH control output is asserted. This directs PCINCR to take the branch by adding 2xdisp8 to the PC.

THE EXECUTE STAGE

Now, let's discuss the EX stage ALU, result mux, and address unit controls. The ALU and shift control outputs are:

- ADD: set unless the instruction is sub or sbc
- CI: carry-in. 0 for add and 1 for sub, unless it's adc or sbc where we XOR in the previous carry-out
- LOGICOP_{1:0}: select and, or, xor, or andn. LOGICOP_{1:0} is simply EXIR_{5:4} (i.e., EX stage copy of FN_{1:0})
- SRI: shift right input—0 for srl i

Next cycle	Next address	Outputs
IF	$AN \leftarrow PC_0 + 2$	SELPC PCCE
IF branch	$AN \leftarrow PC_0 + 2 \times \text{disp8}$	BRANCH SELPC PCCE
IF jal call	$AN \leftarrow PC_0 = \text{SUM}$	PCCE
IFreset	$AN \leftarrow PC_0 = 0$	SELPC ZERO PCCE
LS load/store	$AN \leftarrow \text{SUM}$	—
DMA	$AN \leftarrow PC_1 + 2$	SELPC DMAPC PCCE
DMA reset	$AN \leftarrow PC_1 = 0$	SELPC ZERO PCCE

Table 6—Here's a look at the result multiplexer output enable controls. The instruction determines which enable to assert and thus determines which function unit drives the RESULT bus.

and A_{15} for srai (shift right arithmetic)

slix and srx (shift extended left/right for multi-word shift support) are not yet implemented. Be my guest!

The result mux control outputs SUMT, LOGICT, SLT, SRT, SXT, and RETADT are active low RESULT bus 3-state output enables. Each cycle, all EX stage function units produce results. One asserted T enables its unit's 3-state buffers to drive the RESULT bus, as shown in Table 5.

ZXT zeroes $\text{RESULT}_{15:8}$ during lb . As you'll see next month, the system drives $\text{RESULT}_{7:0}$ with the byte load result.

The following outputs control the address unit:

- BRANCH: if set, add $2 \times \text{disp8}$ to PC, otherwise add +2
- SELPC: if set, next address is $\text{PCNEXT}_{15:0}$, otherwise $\text{SUM}_{15:0}$
- ZERO PC: if set, next address is 0

- PCCE (PC clock enable): update PC_i
- DMAPC: if set, fetch and update PC_1 (DMA address), otherwise PC_0 (PC)

Depending on the next memory cycle and the current EX stage instruction, the control unit selects the next address by asserting certain combinations of control outputs (see Table 6).

WRAP-UP

This month, we considered pipelined processor design issues and explored the detailed implementation of our xr16 control unit—and lived! The CPU design is complete. The final article in this series tackles the design of this System-on-a-Chip. 📄

Jan Gray is a software developer whose products include a leading C++ compiler. He has been building FPGA processors and systems since 1994, and he now designs for Gray Research LLC. You may reach him at jan@fpgacpu.org.

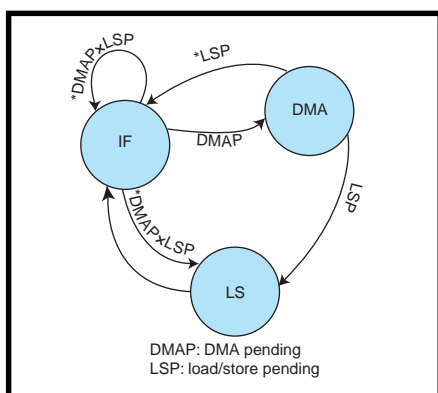


Figure 4—Each memory cycle is an instruction fetch unless there is a DMA transfer pending or the EX stage instruction is a load or store. The FSM clocks when one memory transaction completes and another begins (on RDY).

SOFTWARE

Visit the *Circuit Cellar* web site for more information, including specifications, source code, schematics, and links to related sites.

REFERENCE

- [1] D. Patterson and J. Hennessy, *Computer Organization and Design: The Hardware/Software Interface*, Morgan Kaufmann, San Mateo, CA, 1994.

The Shocking Truth about EMC

FEATURE ARTICLE

George Novacek

Part 1: Design for Compatibility

Designing for electromagnetic compatibility shouldn't just be an afterthought. As George shows us, if you wait until EMC causes a problem, you just might find yourself doing quite a bit of redesigning. Listen up and save yourself the trouble.



The tremendous growth in electromagnetic compatibility (EMC) importance has affected all of us. Aerospace engineers have seen their requirements skyrocket while designers of communications and commercial electronics, especially those in the European countries, have had dozens of stringent new requirements imposed on them.

With the greater sophistication of simple appliances, we increasingly rely on flawless performance of electronics. With the profusion of cellular phones, pagers, and ever-escalating computer clock speeds leading the pack, we generate a huge amount of electromagnetic garbage, all of which seems to choke and confuse every electronic device within its reach.

Electromagnetic compatibility has, for many years, been viewed as black magic. Designers shied away from it, hoping that by copying previous successful designs, the problems would go away. And, quite often they did.

Unfortunately, this approach no longer works. The quest for better performance, lower cost, smaller size, and shorter time to market forces designers to understand the problems before they can find the remedy. In this article, I'll look at taking control of and taming the EMC beast.

I'll discuss some fundamental principles of designing electronic equipment in compliance with the regulatory requirements. Even if no government certification is required, it is our duty as engineers to make sure our products work under adverse conditions. At the very least, it's a matter of professional pride.

It would be impractical and outside the scope of this article to write a cookbook to satisfy every conceivable situation. Rather, I'll concentrate on the principles, methods, and their application to critical level aerospace requirements as reflected in the DO-160D or MIL-STD-461D standards. It's up to you to transpose these principles to your design environments, which are probably less demanding (e.g., aerospace insists on 200 V/m immunity, but commercial equipment rarely encounters more than 20 V/m). Those interested in deeper technical details can consult the references.

FIRST THINGS FIRST

The first step in destroying its black-magic reputation, is to treat EMC as any other design problem. Once you define it, you need to form a plan of attack.

Many designers shy away from EMC issues and underestimate the importance of a plan. They often feel they have no time to waste on this obscure (and secretly feared) discipline. Yet, at this point, you either become proactive in controlling the EMC, or you allow the EMC to control you. If you consider the cost of redesigning to fix EMC problems after they become problems, can you afford not to spend a few days planning? Just the cost of an EMC test lab will run you in the range of \$1500 per day.

At the early stages of projects, military and aerospace organizations demand that their design teams prepare a formal document, usually called *EMC Control Plan*. There is no need for a commercial designer or an enthusiast to go to such lengths. However, every designer should understand as a minimum:

- definition of the EMC requirements (customer's and/or regulatory)

- analysis and method of satisfying the previous requirements
- method of formal verification of compliance, if needed

The informal plan may be merely a couple of pages to list and address the issues. The definition of the EMC requirements is usually issued by the customer and given as a reference to an applicable regulatory requirement or a standard. Even though you may be well acquainted with the standard, your team members may not be. So, it's a good idea to spell out what must be achieved, if for no other reason than as a reminder of the design goal.

The second section of the plan addresses the requirements specifically and how you plan to meet them through circuit design, shielding, grounding, packaging, filtering, and other methods to improve system immunity and keep emissions under control. Here you also define system interfaces. And, two additional subjects, although not strictly EMC but closely related, will be addressed here—the ESD protection and the power supply performance.

Finally, you must decide early in the process how you will prove to the customer and the certifying agencies that you have satisfied their requirements. This often overlooked detail can cause a lot of grief at the end. Some characteristics may have to be proven by analysis or by similarity. You can easily imagine the time and frustration it may take to reach a consensus of engineering opinions when presenting a similarity or a rationale report.

Nothing works better than tests to prove your point, but if an analysis is the only way, it's important to keep the customer and the agency advised and obtain their agreement with your verification plans well before you go past the point of no return.

THE PLAN OF ATTACK

Defining the plan needs to be nothing more than a simple matrix summarizing the design goals, their regulatory references, and specific objectives with supporting data for the designer.[1]

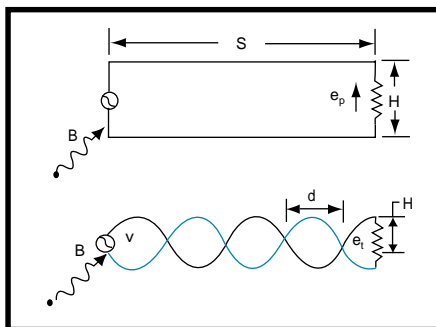


Figure 1—Wires connected to a load form a loop, whose transmission gain is proportional to its area formed by S (length) times H (the distance between the wires). By twisting the wires, many small loops result with two adjacent ones being always out-of-phase and therefore cancelling each other's effects.

Let's say you're about to design an embedded controller for a closed-loop position-control system that satisfies DO-160D standard. The electrical regulatory and operational requirements can be subdivided into five groups—magnetic effect, power supply considerations, susceptibility to interference, RF emissions, and lightning-induced susceptibility.

Magnetic effect is a requirement for equipment that generates a magnetic field that could potentially affect other equipment, such as a compass. It's generally not a problem with low-power microcontroller systems. Otherwise, you need to provide magnetic shielding of the offending components.

With power supply considerations, you define the characteristics of the external power feeding the controller to make sure it will continue working under all conditions. Let's assume power is supplied from a standard 28-VDC avionic power bus. It's not uncommon to call category Z per DO-160D. Our internal power supply circuit must ensure that:

- the controller is not upset by power interruptions up to 1 s long
- supply voltage may vary between 18 and 32.2 VDC
- the system withstands transients of 80 VDC for 100 ms, 48 VDC spikes for 1 s, and dropouts to 12 VDC for 30 ms

- the supply voltage dropping below the minimum 18 VDC for any period of time at any rate of change does not cause the system to lock up
- 50 transients of 600 V and 2 μ s duration, of each polarity and applied within 1 min. on top of the 28-VDC power don't upset the system

Every electronic instrument is potentially susceptible to interference. It is your responsibility to design the controller so it will not be affected by interference. Typically, you will have to make sure the controller's operation is not upset when:

- a ripple of the frequency and amplitude range is superimposed on the 28-VDC supply lines: 10Hz–200Hz at 0.2VRMS, 200Hz–1kHz at 0.55 VRMS, 1kHz–15kHz at 1.5 VRMS, and 15kHz–150kHz starting at 0.2 VRMS and sloping down at –40 dB per decade
- common mode interference is induced in the external interconnect wires and harnesses through a coil wrapped around them, producing a 30-A/m magnetic field at 400 Hz, gradually reducing with frequency to 0.8 A/m at 15 kHz. In this article, all references to interconnect wires and bundles mean external wiring unless noted otherwise.
- the interconnect wires are exposed to an e-field of 1800 V/m between 380 Hz and 400 Hz

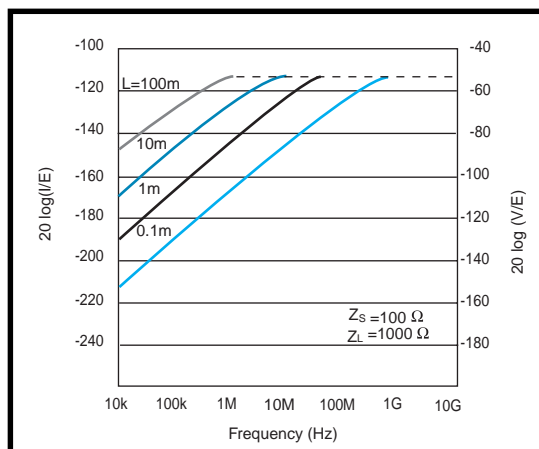


Figure 2—To find the voltage or current level induced into a wire in an E-M field, run a vertical line from the abscissa at the frequency of interest until it intersects with the curve appropriate for the length of the wire. A horizontal line through that point will show the induced voltage or current on the ordinate. Reference [3] shows diagrams for different impedance combinations.

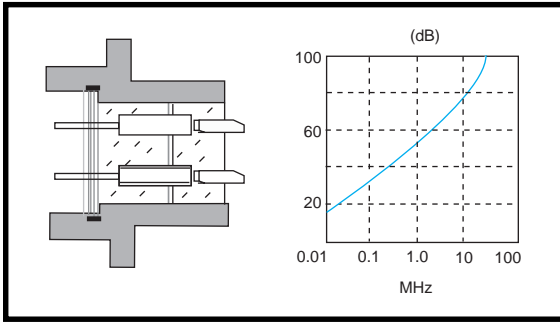


Figure 3—In a filter connector, pins act as low-pass π filters. The pins, carrying ferrite beads making them appear like inductors, are surrounded by a dielectric and a ground plate to create distributed capacitance. The attenuation shown in the plot to the right of the connector cross-section is usually limited to -60 dB due to impedance mismatch.

- a common mode burst of spikes of $600 V_{pp}$ spaced at 0.2 to $10 \mu s$, 50 to $1,000 \mu s$ long, is injected into the interconnect wires through a coil wrapped around the wire bundle
- $200 V/m$ field from 500 kHz to 400 MHz, decreasing at -20 dB per decade from 500 kHz– 10 kHz is injected into the interconnect wires through a current transformer
- the system with all its components is exposed to a modulated, as well as a continuous, wave e-field of $200 V/m$ from 100 kHz– 18 GHz
- electrostatic discharge of the test pulse to the cabinet does not damage or upset the system operation

RF emissions are quickly growing in importance. Not only is it your responsibility to ensure flawless operation in the midst of interfering signals, you are also expected to minimize EMI pollution. Believe me, between switching power supplies and digital circuits' clocks, you have enough to worry about. There are different standards and different categories within those standards, but let's assume DO-160D category M is applicable. The authorities will want to be satisfied that:

- when running the equipment's 28 -VDC power lines through a current probe, there are no signals up to 150 kHz exceeding 53 dB/mA, with this maximum allowed level sloping at -20 dB per decade to 2 MHz and not exceeding 20 dB/mA from 2 – 30 MHz
- using the same method, emissions from all the other interconnecting

wires do not exceed the 28 -VDC power line limits, as in the previous bullet, by more than 20 dB/ μA

- radiated emissions (received by an appropriate antenna) do not exceed $35 \mu V/m$ from 2 – 25 MHz. The maximum emissions are allowed to increase from 25 MHz to 6 GHz at $+20$ dB/ $\mu V/m$, with -10 dB notches at 100 – 150 MHz, 1020 – 1100 MHz, and 1525 – 1680 MHz

Like the famous bunny rabbit, our controller is expected to keep going and going without a hiccup when exposed to a Level 4 lightning strike. In practical terms, it means that there is no damage to the controller when every interface and power connection is injected with $1500V/60A$, $750V/150A$, and $750V/750A$ waveforms of up to $120 \mu s$, as defined in DO-160D.

Also, it's important that there be no functional upset when, using a current transformer, the interface cable bundle of the operating controller is injected with test pulses of $750V/1500A$, $1500V/300A$, and $750V/2000A$. The shape and timing of the pulses are also defined in DO-160D. The system can shut down (provided it's done in a safe, predictable manner), but it must automatically recover.

This is a tall order. The EMC requirements fall into two major groups—emissions and immunity. With emissions, the equipment will generate signals that may interfere with other equipment. The most common sources of interference would be switching power supplies and digital circuits, including clock generators. The interference travels from the unit by air (radiated emissions) and by wires through power lines and interfaces (conducted emissions). Immunity (also referred to as susceptibility) is divided into

several areas—High Intensity Radio Frequency (HIRF) susceptibility, audio frequency susceptibility (which in effect is a ripple superimposed on the power lines), indirect lightning effects, and electromagnetic pulse (EMP) susceptibility. Electrostatic discharge (ESD) is another concern, but if you have to design equipment to survive HIRF and indirect lightning effects, EMP and ESD will be automatically taken care of.

Susceptibility can be radiated or conducted through the wire interfaces. When it comes to the effects on the equipment, you need to consider damage tolerance (will the equipment survive a lightning strike or an ESD discharge?) and functional upset (will the equipment continue to operate properly when exposed to a strong e-field?). Here, the criticality of the application determines the proper behavior.

Less critical equipment may be allowed to stop operating during exposure to an interference, as long as it does not perform a forbidden action and recovers after the exposure has ended. Critical equipment must keep operating correctly no matter what.

A separate set of design constraints is set up to address the power supply. We must consider the ability of the equipment to function without upset or damage through a wide range of the power supply voltage, interruptions, and power transients. Before we delve into the practical activities of taming the EMC beast, we need to touch on the theory of electromagnetic interference.

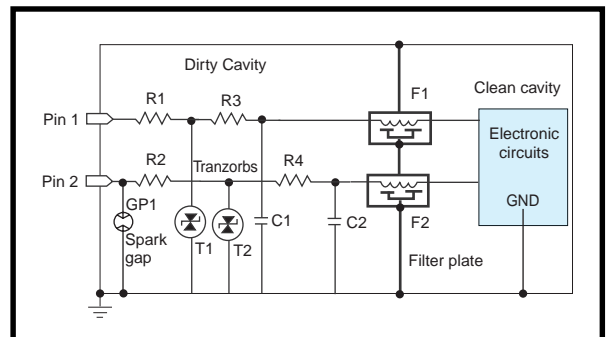


Figure 4—A typical two-cavity design has transient protection devices in the dirty cavity, which serves merely as a housing for environmental protection. From the EMC standpoint, full e-field exists there. The high frequency signals are filtered before reaching the clean cavity by feed-through filters installed in the filter plate.

LOW FREQUENCY WIRES

The electromagnetic energy gets in and out of the electronic circuits through the radiation (antenna) properties of conductors. If the wire length is less than 10% of the wavelength λ , the currents at any points on the wire can be considered in-phase—a condition generally referred to as a short antenna.

The typical examples are power leads with DC, 60 Hz, or 400 Hz current generating electric and magnetic fields. One type of field, magnetic (current) or electric (voltage), will usually predominate.

The magnetic field at low frequency will be:

$$|H| = \frac{I}{2\pi R}$$

where I represents the current and R is the distance from the wire. Often, two wires close to each other are used, where the current flows down one and returns through the other. The pair generates a magnetic field:

$$|H| = \frac{I \times d}{2\pi \times R^2}$$

assuming $d \ll R$. Although the current I and the distance from the wires R are design constraints, we can minimize the distance between the wires d to the insulation thickness and thus minimize the stray magnetic field.

There's another way to look at it (see Figure 1). The voltage induced in, or a field emitted from, the parallel wire run (loop) is proportional to the area of that loop if both S (wire length) and d (wire separation) are much smaller than the wavelength λ . Then the induced voltage is:

$$e = A \frac{dB}{dt}$$

where A is the loop area and B represents magnetic flux density. We can rewrite that equation as:

$$E_p \propto S \times d$$

If you twist the two wires, you not only reduce the area of the loop, but you also cause only one loop to be active. All loop pairs' voltages will cancel out as they are out of phase as

a result of the twist. The theoretical reduction of EMI is:

$$\frac{e_p}{e_t} = N$$

where N represents the number of twists. In practical terms, it will be limited to about 1,000 (60 dB) for frequencies up to 100 kHz. As the frequency increases, the loops are no longer 180° out of phase, and eventually, the twisting effect will be lost.

At this point, the effect of coaxial cable should be mentioned. It can be viewed as a wire pair, with d being represented by the eccentricity of the conductor within the shield. Another common approach, especially in the automotive industry, uses one wire connection and the return through the chassis. Such a system behaves as if there was a mirror image of the second wire below the ground plane. In other words, the d in this case is double the distance from the ground.

To maximize the effect of twisted wires, they must form a balanced pair (i.e., one carries the current, the other

is the return, and no other path is involved). The maximum, 60 dB, attenuation can be approached up to the maximum frequency between 100 kHz and 2 MHz. It also needs to be reiterated that twisting has little effect on electric fields—these must be suppressed by shielding.

HIGH FREQUENCY WIRES

When the wire can no longer be considered short compared to the wavelength, it becomes a more efficient radiator. Such sources of radiation are customarily divided into two basic types—a small magnetic and a small electric dipole (small, in this case, referring to the size of the dipole compared to the wavelength λ).

An example would be a PCB trace connecting logic gates. With a CMOS gate, which has high impedance input, the dominant signal is voltage, producing dominant electric field. The same circuit driving a TTL gate, which has low impedance, creates a loop with predominantly current signal and the resulting magnetic emission.

Each type of dipole produces both fields, but the electric dipole is characterized by $C \times V \times L$, where C represents the capacitance between the leads, V the voltage, and L the distance between the dipole elements. The magnetic dipole, on the other hand, is characterized by $I \times A$, where A represents the area of the loop and I the current through the dipole.

Your main concern is that the efficiency of dipoles increases with the frequency, reaching maximum at:

$$\frac{\lambda}{4}$$

That puts most of your interface cables into the category of efficient radiators in the frequency range of 20–200 MHz—the band tightly controlled by all regulatory agencies.

DESIGN ANALYSIS

So, what does all this mean in practical terms? To simplify your analysis, look for the worst-case scenario and design for it. Let's assume the maximum length of the interconnecting cables in our system is 3 m (about 10 ft). The cables will be connected to the peripherals with internal resistance $Z_s = 100 \text{ W}$ and terminated inside the controller with 1000 W . When exposed to an e-field of 200 V/m intensity, there will be about 0.5 VRMS induced on the cables above 30 MHz, falling off at a rate of -20 dB per decade below 30 MHz.

The easiest way to establish the levels is to use graphs like the one in Figure 2.[3] Assuming the cable will never run more than 5 cm above the ground (chassis), a correction factor of 26 dB (also published in [3], in a graph form) must be applied, bringing the induced voltage to 10 VRMS.

You can appreciate how important cable routing is and how much the picked-up interference grows when the cables hang in the air! Just moving the cable 4.5" above ground will result in a 60-dB correction, bringing the voltage picked up to 500 VRMS.

Cable shielding effectiveness of -40 to -60 dB can be achieved. Consequently, you can assume that by properly shielding the wiring and using the above scenario, the maximum

interfering voltages will be somewhere around 100-mV RMS. You can achieve the same results with the magnetically induced interference, as we have seen previously, by using twisted and shielded wire pairs for all interfaces. Thus, the same attenuation can be achieved for the magnetically induced/radiated interference, as well as electrically induced signals.

You still need to low-pass filter the incoming signal. Although 100 mV may not seem like much, especially when it is way outside the operating bandwidth of the embedded controller, you want some design margin. The interconnect cable may be more than 5 cm (2") above ground, the shielding could get damaged, or some other unexpected occurrence may happen. Because the interfering signals are often modulated, once their induced level reaches about 400 mV, semiconductor junction's nonlinearity may effectively demodulate the interfering signal, and the result may fall within the operating spectrum, affecting the operation.

Commonly, you bring the connections from the interface connector(s) to the PCB, where they feed directly into low-pass filters. Unfortunately, those connector-to-PCB wires act like antennas, invalidating the shielding effectiveness of the cabinet at high frequencies. This is often acceptable at lower e-field levels, but not at all satisfactory at 200 V/m, which is our example design requirement.

In situations where lightning strike or a high voltage pulse is not a concern, filter connectors present an excellent option. The connector pins are in fact miniature, often π configuration, low-pass filters with attenuation, as shown in Figure 3. The filter pins provide effective attenuation of the interfering signal, but in my view -60 dB is, as a result of the impedance mismatch, the maximum you can practically expect.

The major disadvantage of the filter connectors is that the dielectric strength of the material forming the capacitor between the pin and the shell (ground) is often limited to 50 VDC operating voltage. This makes the filter connector unsuitable

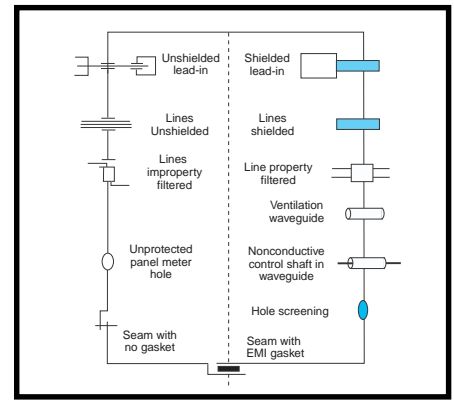


Figure 5—Common mistakes (shown on the left side) will degrade shielding properties to nothing. Whether it is a standalone package or one that works together with a dirty cavity, maintaining EMI integrity (as shown on the right side) will make the difference between a successful design and a failure.

for designs where power transients and lightning strike effects far exceed the 50 WVDC rating of the pin's distributed capacitor. Connectors rated for higher operating voltage, even with built-in clamping devices to limit the transients, are available, but their selection is limited and their cost is prohibitive for most applications.

In a typical system, transient protection devices (which I address later) are in immediate vicinity of the connector pins, followed by low-pass filters. This brings the interfering signal right into the cabinet, invalidating its shielding effect.

In the worst-case scenario, you must conservatively assume the full e-field appears inside the cabinet as a result of the radiation from the incoming wiring, the waveguide effect of the apertures, and resonance in the enclosure at high megahertz and, certainly, gigahertz frequencies. But, we also assume that we have a good PCB design with SMT components, no loop larger than 1 cm^2 is formed by the signal and its return, and that chassis grounding of filters retains its low impedance even at the highest frequencies.

Based on the above assumptions and using the same type of graphs as shown in Figure 2, you can determine that the loop can pick up about 80 mV of interference. Symmetrical inputs will not be completely immune to this interference either. The common mode rejection ratio (CMRR) usually

deteriorates to about -26 dB as a result of tolerances, leaving about 4 mV of interference in differential mode. The problem is that transient protection devices introduce fairly large trace lengths and current loops. A 20-dB increase in the calculated pickup is reasonable, leaving you with 800-mV common mode and 40-mV differential mode interference to deal with.

The solution to this problem is to use a dual-cavity design, like the arrangement shown in Figure 4. First, bring the interfaces through connectors to the electrically dirty cavity, where you clamp the transients, using transzorbors, MOVs, spark arresters, or a combination of these devices in a multistage configuration (addressed in detail in Part 2) to maintain the signal amplitudes within safe limits. Then, using feed-through low-pass filters, you feed the signal through the metal wall into the electrically clean cavity.

Figure 4 illustrates the concept of the dual-cavity design and depicts transient protection and filtering of typical input lines. The transzorbors in this arrangement can handle Level 4 lightning effects, as required for the example system. Resistors R1 and R2 are small, 20-Ω resistors that limit the maximum current through the transzorbors. They are followed by R3C1 and R4C2 low-pass filters, to provide contact debouncing and bandwidth limiting. For larger currents, a spark gap GP1 may have to be added, and/or R1 and R2 replaced with chokes and R3,R4 omitted, as we'll see next month. The filters also provide ESD protection.

Finally, the feed-through π filters provide high-frequency attenuation of about -60 dB up to the highest frequencies. Feed-through filters are available at critical frequencies as low as a few kilohertz, but with the push for smaller and smaller devices, most of the filters today start cutting off at about 1 MHz. When I discuss the design of specific interface circuits in next month's article, I'll take a closer look at protecting interface circuits.

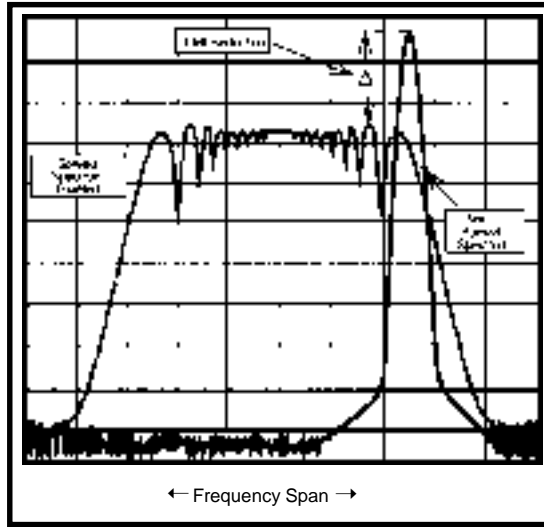


Figure 6—Spread-spectrum oscillator is a new technology. By spreading the signal spectrum as opposed to concentrating it in a precise, narrow band, we can significantly reduce unwanted emissions without the need for shielding.

UNDER CONTROL

In many respects, EMC immunity and emission control issues can be looked at as mirror images of each other and, if you've taken good care of the immunity, emissions should be taken care of automatically. This, however, is not always true. You still need to be careful, or some unexpected emission problem will pop up and bite you.

It's easy to become a little complacent about EMC emissions until a problem arises. We tend to concentrate on the susceptibility first, because it is defined in numbers, while we really don't know what kind of emission our equipment will generate. Bringing the immunity under control is also a little more forgiving than the emissions.

There are workarounds for some susceptibility problems that don't require you to beef up the packaging. Sometimes a ferrite bead threaded on a wire, a minor circuitry fix, or a software routine will compensate for inadequate EMC protection that allowed interference inside the cabinet. On the other hand, there are few simple circuit fixes that can reduce emissions. Using spread spectrum oscillators is one of the few options that can be easily implemented. Usually, the fix is largely mechanical and, therefore, complicated and often expensive to implement in later stages.

For example, a flight controller with the same EMC specifications as in our design example was undergoing qualification and passing all the tests. Once the immunity was completed, we expected smooth sailing. But then emission testing showed an excess at 60 kHz and its harmonics up to 1 MHz. Investigation revealed that the problem was related to the internal SPI interface of data acquisition ICs running at a 60-kHz bit rate.

The problem arose in the final version when the design had to be squeezed into a smaller envelope than originally intended.

The small size of the feed-through filters resulted in their critical frequency being pushed from 10 kHz to about 1 MHz. Only two options existed.

Pushing the communications frequency above 1 MHz would have been too radical a change to implement and re-certify in time, its impact on the budget notwithstanding. To use filters with lower critical frequency was equally impractical because there was no way these could be squeezed into the existing envelope. Changing the envelope was out of the question.

Fortunately, EMC standards provide a hefty margin between the maximum allowed emissions and the minimum susceptibility levels, so minor emission excesses can be often accepted. Because our exceedance was no more than 4 dB above the allowed emission level, the customer was able to accept it, but the fact is, we failed to deliver expected performance.

Ideally, you should strive to package equipment in a well-designed, shielded cabinet. The control of the effects of the external energy fields on the circuits inside the cabinet, as well as the containment of the internally generated fields, can then be reduced to the control of the EMI pickup and emissions through the wires and cables. If you follow the fundamental packaging principles as shown in Figure 5, the only way remaining for interference to get in and out of the cabinet is through the peripherals and their wire interfaces.

Just as you want to make sure minimum energy couples to the external wiring, you must follow the same philosophy with the internal design. There's little use in heavy filtering an I/O line to kill externally conducted interference if it picks up the internal system clock through radiation. Successful control of EMC depends on good overall design from the cabinet through the PCB layout to the electrical circuits and software.

Up to now, I have concentrated largely on what can be called a classic EMC design—a two-cavity, metal cabinet with feed-through filters, efficient shielding, and so on. But, continuous pressure for small low-cost and lightweight designs makes the classic approach often impractical.

As you have seen, the interference the circuits pick up is a function of their physical size. As Figure 2 shows, keeping connections short and close to the ground plane, while properly shielding external interfaces, reduces the interference levels to millivolts, and such interference can be easily rejected by digital circuits. (A situation almost inherent to the contemporary designs with SMT components and multilayer PCBs.)

Sensitive analog circuits (or our own interference emitters) can be shielded separately by small "sardine can" covers. You can design PCBs with solid ground planes on the outside and place them in a metal frame to effectively behave like a solid metal box. In high-volume production, a metallic coated plastic enclosure is often used.

It is the lightning strike transient protection that introduces physically large connections prone to the interference pick up and radiation. When combined with the need to operate in high-energy fields (HIRF), the classic two-cavity, metal cabinet design remains the only choice.

GENERAL CONSIDERATIONS

The one area greatly affecting the ultimate EMC performance is the PCB layout. Present-day multilayer boards combined with surface mount (SMT) components afford excellent EMC performance. This is achieved

by keeping traces (antennas) short and shielded from radiated fields by good ground planes. PCB layout, especially for high speed, is a discipline in its own right, and for sophisticated, demanding designs one is well advised to subcontract such layouts to a professional. There are several fundamental rules worthy a reminder:

- provide ample ground and power planes on separate layers. Follow one-point grounding rules.
- keep traces short. Avoid current loops, including ground loops.
- don't allow analog and digital grounds and power planes to overlap to prevent capacitive coupling between them.
- use decoupling capacitors as close to the ICs as possible. Capacitors that fit underneath the chip are ideal.

Increasingly, the major contributors to PCB emissions are VLSI digital circuits, such as microprocessors and FPGAs (field programmable gate arrays). Significant emission reduction can be achieved by judicious assignment of ground and low-speed FPGA connections on ball grid and super ball grid array (BGA and SBGA) packages to the perimeter and the high-speed signals to the inner balls.

Using the IC packages with internal heat sink shows a significant reduction in emissions when grounded in multiple points. It is interesting to note that grounding the heat sink in one point only shows an 8-dB penalty.

The EMC performance of the PCB can be significantly improved with on-board shielding. Recent advances in technology result in the availability of extremely light, efficient metal shields suitable for SMT pick and place machines at a low cost. Many standard shields exist and can be purchased in small quantities for development or low-volume production.

Use of the spread spectrum clock to reduce the circuit emissions is gaining popularity. The principle, used in communications for some time, is quite simple. Instead of having all the energy concentrated in a narrow bandwidth around one discrete frequency, by sweeping the fre-

quency, you widen the bandwidth and reduce the amplitude of the emissions proportionately.

This approach, illustrated in Figure 6, was introduced by Lexmark. It sweeps the clock frequency $\pm 1.7\%$. This results in the emission reduction by about -8 dB—sufficient in many cases to meet the FCC requirements without any additional shielding.

A clock generator chip with the sweep function can be purchased from several manufacturers, such as Cypress. Obviously, the larger the sweep, the greater the emission reduction will be, but the penalty is timing accuracy. The frequency deviation is a compromise providing sufficient reduction to satisfy certification authorities, while maintaining expected timing accuracy.

The objections to the swept clock timing inaccuracy can be alleviated by a similar method, which relies on several discrete clocks offset by a small frequency.

Let's say you have several functional blocks, each operating from and emitting the same clock frequency. The overall emission will be the sum of the module emissions, increasing by 6 dB for each additional emitter. Using a separate clock for each module, offset by, say, 10 kHz (such as 20 MHz, 20, 01 MHz, etc.) will result in a four-module design having the emissions -18 dB below the same unit with the same clock frequency for all modules.

In addition, you could also sweep each of these frequencies, except the one that is timing critical, and achieve even greater reduction. An added benefit of this method is that the diagnostics can be designed to pinpoint a system problem by identifying it through its specific clock frequency.

IN SHORT...

How do you design electronics for electromagnetic compatibility? Exact calculations of radiated fields, whether generated by or their effects on complex circuits and PCBs, are next to impossible. Consider the worst-case scenarios, then rely on sound engineering practice and experience.

As you progress in the design, options left to you to correct potential EMC problems decrease dramatically,

while the cost of their implementation increases exponentially. Therefore, it is in your best interest to invest relatively little time at the beginning and make sure you understand the design requirements related to the EMC and know how to implement the necessary steps to control them.

Prices of EMC test equipment are becoming accessible to smaller companies, so confidence testing can be done early in the design to mitigate the risk. But, even without access to the test equipment during the design process, you can take steps to minimize the risk of a future failure:

- if possible, use a metal enclosure. It will make life easier. Commercial equipment can often satisfy certification requirements (UL, FCC, European) without exotic packaging, if sound engineering design is used throughout.
- for military or aerospace applications, especially when the environmental requirements are taken into consideration, life is too short and quantities too low to even consider designing without a good, metal cabinet.
- use multilayer boards with good ground planes. Make sure there are no current loops. Connect all the grounds at a single point. On each PCB put a reverse parallel diode combination between the digital and analog grounds.
- use SMT components as much as possible, keep the tracks short, and do not skimp on decoupling capacitors.
- use spread spectrum and/or multiple discrete clocks.
- avoid clock frequencies below the critical frequency of the interface filters.
- ground heat sinks at multiple points and make a provision to use shielding over microprocessors, FPGAs, clock generators, and sensitive, primary analog circuits. Depending on the test results, the shields may be omitted from the final assembly, which is easier than adding them after the fact.
- even with the luxury of dual cavity packaging, maintain the shortest

possible connections between the connector pins and the I/O filters on the PCB. That means the filters should be right where the wire enters the board. Ideally, the connector itself (which may have filter pins) should be soldered directly into the board. If wires have to be used they should be shielded if longer than approximately 1".

- use filter connectors, dual cavity design, and low-pass filters to limit the frequency spectrum reaching the electronics.
- proper grounding is absolutely crucial, and its importance cannot be overstated.

THE WAY FORWARD

I've discussed the impact of aspects of the EMC on design of electronic systems in general. In Part 2, I'll look specifically at how to design grounding, power supply and distribution, inputs and outputs, and cabinets (among other things). ☒

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

REFERENCES

- [1] G. Novacek, "Testing 1,2," Circuit Cellar Online.
- [2] DO-160D, RTCA, Inc. Washington, D.C.
- [3] O. Hartal, *Electromagnetic Compatibility by Design*, R&B Enterprises, West Conshohocken, PA.
- [4] Cypress Semiconductor Corp., W181, *Peak Reducing EMI Solutions*.
- [5] *EMI Control, Methodology & Procedures*, D.R.J. White and M. Mardiguian. Interference Control Technologies, Don White Consultants Inc., Gainesville, VA.
- [6] F.A. Fisher and J. A. Plumer, *Lightning Protection of Aircraft*, Lightning Technologies Inc., Pittsfield, MA.

NOUVEAU PC

Edited by Harv Weiner

REDUNDANT POWER SUPPLY

Ziatech Corporation has introduced two reliable, redundant, and modular switching power supplies for CompactPCI computers. Available in both AC and DC input models, the two new power supplies provide 150 W of power in a rugged 3U format that plugs directly into a CompactPCI backplane. Both power supplies feature hot swap and load sharing capability, enabling their use in mission-critical applications that require total availability.

The ZT 6301 power supply features a universal input voltage range from 90 to 254VAC at 47 to 63 Hz with power factor correction. The ZT 6311 power supply accepts an input voltage range from 36 to 72VDC. Both units provide four outputs that are capable of providing a total of 150 W for 3.3 VDC, 5 VDC and ± 12 VDC with independent output regulation. Overvoltage, short circuit, and overtemperature protection is provided for both units along with built-in EMI filtering, status LEDs, and main output remote sense.

Both are compatible with the CompactPCI specification of the PCI Industrial Computer Manufacturers' Group (PICMG), and meet UL, CSA, IEC, TUV, and CE certification standards.

The ZT 6301 Power Supply sells for \$342, and the ZT 6311 for \$360.

Ziatech Corporation
(805) 541-0488
Fax: (805) 541-5088
www.ziatech.com



DIGITAL I/O INTERFACE

The DIO-16.PCI is a digital I/O interface card that provides eight optically isolated inputs and eight reed relay outputs. The isolated inputs provide protection to connected equipment, and the reed relay outputs allow PC control of lights, buzzers, switches, motors, and other low-current devices supporting on/off control. Applications include PC based control and automation of equipment including sensors, switches, satellite antenna control systems, video and audio studio automation, and security control systems.

The Seal/O suite of Windows 95/98/NT drivers is included with the card. Seal/O provides a consistent and straightforward API (Application Programmer Interface) to facilitate application development. Supported

development environments include Visual C++, Visual Basic, and Delphi. Seal/O includes a utility for configuring the driver parameters under Windows 95/98 and Windows NT, to further simplify installation.

Seal/O TST, a Windows 95/98/NT Console application (32 bit), is included to allow the user to exercise

the inputs and relays. Source code for Seal/O TST is also included to facilitate software development. Seal/O VB, a 32-b Visual Basic sample with GUI (Graphical User Interface), allows control of individual and groups of relays and timed relay activation.

All of the card's outputs are through a DB-37 connector. The KT-101 Terminal Block Kit is available to provide a simple means of connecting field wiring to the card. The Kit consists of a basic 6' M/F cable and positive tension screw terminal block.

The DIO-16.PCI sells for \$259. The KT-101 sells for \$49.

Sealevel Systems, Inc
(864) 843-4343
Fax: (864) 843-3067
www.sealevel.com



NOUVEAU PC

REAL-TIME NETWORKING SYSTEM

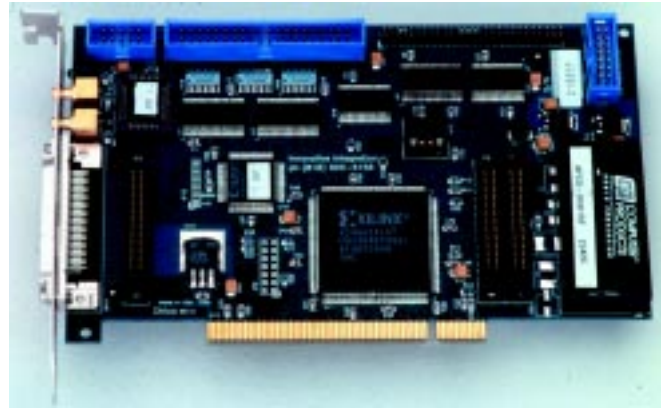
The SCRAMNet+ SC78 series is a deterministic real-time alternative to Ethernet networks. SCRAMNet SC78 boards provide 3.4-MBps throughput, comparable to standard IP-based networks, but with lower application-to-application latencies and greater network determinism. SC78 networks incur node latencies, measured in nanoseconds rather than milliseconds, with far more predictability. The result is a network that communicates data within a few microseconds rather than many milliseconds, with no data collisions.

The SCRAMNet Network is based upon a replicated-shared memory concept, where each network interface card (NIC) stores its own copy of the network's shared memory set. When any NIC makes changes to its local copy of shared memory, the network updates all other nodes on the network automatically. No real-time drivers are needed, and there are no time-consuming software routines needed to pack, queue, transmit, dequeue, and unpack messages because network communication is controlled at the hardware level.

SC78 series boards are supplied with intuitive API software, making the network simple to install and easy to program. Once all network nodes are properly configured, no additional software is required to make the network operate as a real-time link. There are no bulky IP drivers to write, greatly reducing development time and cost.

SC78 Series NICs are based on proven SCRAMNet+ technology, but they are designed with a new hardware architecture that supports a single fiber optic connection over distances up to 300 m. Currently interfaces are available in the PCI short card form factor, featuring 1 MB of onboard shared memory and an embedded fiber optic media interface.

Systran Corporation
(937) 252-5601
Fax: (937) 258-2729
www.systran.com



DATA ACQUISITION BOARDS

Chico is a Bus Mastering PCI card that is designed for high-performance data acquisition. It features 32 b of digital I/O, two 24-b timers, and a proprietary Synlink for multicard synchronization. Chico has a real-time event-driven data streaming engine, which automatically streams data between the PCI bus and add-on I/O modules. The card handles bi-directional data streams through the PCI bus while controlling A/Ds, D/As, timers, and digital I/O completely independent from the Host CPU and can sustain gap-free data acquisition.

Chico uses the OMNIBUS I/O module family of products to achieve a variety of analog and digital I/O combinations. These I/O modules connect to the Chico card and provide up to 16 channels, sample rates up to 40 MHz/channel, resolutions up to 18-b, various converter configurations, and S/N ratios as high as 100 dB.

A proprietary SyncLink timing hitch provides accurate inter-board triggering for up to 16 Chico boards in large data acquisition systems. Chico may also be linked directly to Innovative's compatible DSP cards through an 80-MBps FIFO port.

The card does not require complex programming to achieve real-time performance. It comes fully equipped with Innovative's rapid application development (RAD) software, Armada. Armada is an entirely new way to develop sophisticated, hardware-accelerated, real-time data acquisition and control application programs under Win 9x/NT using high-level C++ software components. Data acquisition packages such as LabView can also be used.

Chico sells for \$395. The add-on OMNIBUS I/O modules range from \$845 to \$1645.

Innovative Integration
(818) 865-6150
Fax: (818) 879-1770
www.innovative-dsp.com

NOUVEAU PC

MULTIPROCESSOR COMPACTPCI BOARD

The **Atlas-C** is a multiprocessor CompactPCI board based on the Intel Coppermine-256 Processor and features clock speeds up to 733 MHz. It provides built-in support for symmetric and real-time asymmetric multiprocessing, enabling its two Coppermine processors to work together on the same program.

Each processor is equipped with 256 KB of no-wait-state on-die L2 Cache. The two processors also share up to 1 GB of 100-MHz synchronous DRAM main memory. I/O and networking options include dual 10/100-Mbps Ethernet interfaces (twisted pair); a 40-Mbps UltraWide SCSI interface; and a 64-b AGP graphics engine with four MB of video RAM optimized for 3D rendering. Also available are two Ultra-DMA 33 IDE interfaces, a pair of USB ports, dual serial I/O with optional RS-422 drivers, and a parallel port.

I/O is available via a 80-mm rear panel connector, which can accommodate a

2.5" 9GB IDE drive. For applications that must be deployed without a rotating hard disk, the board provides up to 340-MB SanDisk 1.5" Flash IDE on the rear panel.

For applications requiring high performance at a reduced cost, Atlas-C is also available with a single 466-MHz Celeron PPG370 processor, scalable to 500 MHz. The Celeron processor includes 128 KB of on-die cache with a one-to-one clocking.

Atlas-C runs a variety of popular desktop and real-time operating systems, including Windows NT, Solaris x86, QNX, and VxWorks. Atlas also comes equipped with AMI's BIOS and on-board diagnostics software and status LED's.

Pricing for Atlas-C starts at \$2299 less processor and memory.



General Micro Systems, Inc
(909) 980-4863
Fax: (909) 987-4863
www.gms4vme.com

Ingo Cyliax

Build a Power-On Self Test Card for PC/104

Setting out to weed through an ever-growing pile of “spare” boards and resurrect his backup laptop motherboard, Ingo discovered that a POST card would make the task easier, especially if the card was PC/104 compatible....



If you work with PCs as much as I do, you have probably run across one or two of them that actually does work right. Well, OK, if you're like me, you have a pile of old boards that you are not sure whether they work or not. One of my New Year's resolutions was to go through my pile of boards and save what works and salvage what doesn't.

What makes matters worse, I have PC/104 CPU modules, regular PC motherboards, and some notebook main logic boards (motherboards). One of the laptop boards I use as a backup for my everyday laptop. Although, the board broke several months ago.

When a PC or embedded PC works right, it will boot up into BIOS, initialize the keyboard, look for extension BIOS, initialize any VGA or graphic adapters it might find, and

then proceed to boot from one of the boot devices that are available.

But, what do you do if the problem prevents the computer from making it that far, and it just dies before the screen even comes up? My spare laptop board really fits into this category. In many embedded systems, there may not be a standard screen because the computer doesn't even have a display adapter. What do you do then?

GET POWERED UP

Common PC BIOSs have a power-on self test (POST). This is a series of tests and confidence tests the BIOS performs while exploring the system's hardware and looking for devices. During this test, it will check things like the BIOS ROM check sum, the CMOS battery-backed SRAM, the system memory, various DMA and timer functions, the display adapter, keyboard controller, and whatever boot devices it may find.

When one of the POST routines discovers an error, it typically displays an error message (if the display has been initialized) and waits for the user to clear the error condition and reboot the machine. A common error, for example, is not being able to find the keyboard device or the floppy. These types of errors are usually easy to clear. Some BIOSs allow you to bypass the keyboard test, and disabling the floppy will prevent the POST from halting when it doesn't find one.

The BIOS on embedded computers also allows you to control the behavior on test failure. Sometimes, you can drop into a hardware debugger or tell it to automatically reboot, as well as ignore errors. The last option is pretty risky, because it lets the system boot with potentially bad hardware.

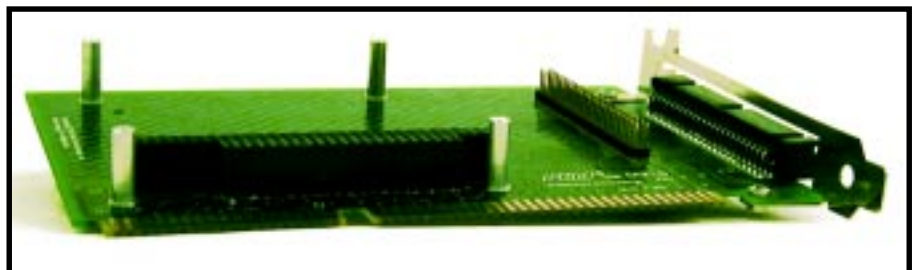


Photo 1—The ISAbus adapter board allows me to plug my PC/104-based POST board into an ISAbus-based system.

But, what do you do if the system boots and halts before initializing the display adapter? Or, perhaps you don't have one, or the one you have is not compatible with the BIOS and relies on high-level routines in the application to initialize it? Well, most BIOSs have a method of dealing with this.

They write special POST codes representing the different phases in the POST to a hardware register in the I/O space. The idea is that you can capture these codes as the POST progresses and when the system stops on an error. The last POST code captured will indicate the routine that encountered an error, or the last POST routine that tried to run before the system hung up. In many cases, this will help you quickly find the error.

There are ISA bus-based POST cards that will capture these codes and display them on an LED display. You then look up the code in a list to decode them. Some products even provide a diagnostic floppy that does a more thorough test of the system than the POST. However, you'll need to have enough functionality in your system to be able to boot from a floppy disk.

PROJECT

Because I have this pile of motherboards and I really want to get my spare laptop motherboard working again, I decided to build a simple POST card. However, not wanting to waste my time with a one-off solution, I decided to build the POST card on a PC/104-compatible card.

A PC/104 card uses stackable, dual-row headers/sockets. This makes sense if you are testing PC/104 systems. PC/104 uses the standard ISA bus protocol, and PC/104-to-ISA bus adapters are available, which allow you to plug a PC/104 module into an ISA bus-based motherboard.

My laptop doesn't have an ISA bus or a PC/104 bus. Because the PC/104 bus connector is essentially a dual-row header, it's fairly easy to build bus adapters for PC/104 cards. I can build a special test cable setup that plugs into a PC/104 module and brings the signals out on wires, which I can solder or clip to the motherboard.

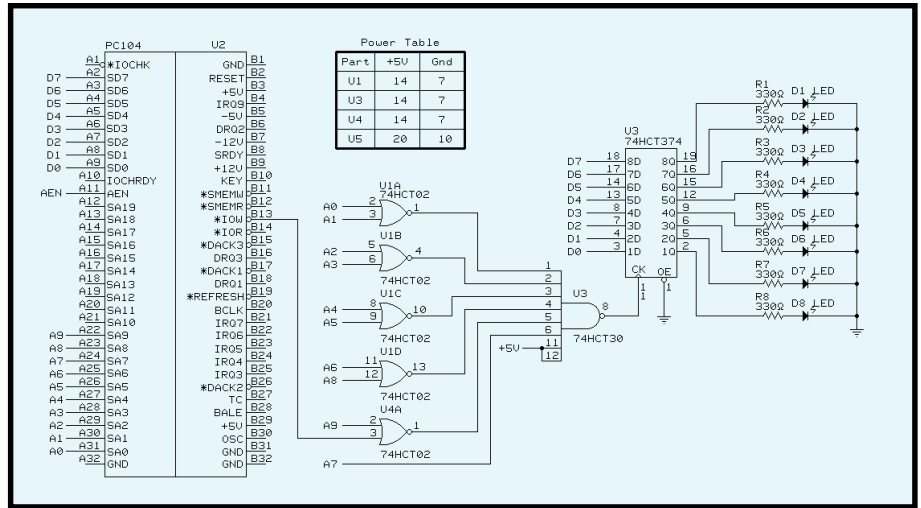


Figure 1—Here is what makes the POST board tick. It's an 8-bit register that latches the data from the data bus whenever the host writes to the I/O address 0x80. You can use standard 74xx technology parts to implement this, because the speed and power consumption are not critical for this application.

A POST board is a write-only register. The content of the register is displayed with LEDs. When the POST writes to the register, the code is displayed and can be looked up in a list of POST error codes.

The register is at address 0x0080 of the PC I/O space. Typically, only the first 10 address lines (A0–A9) are necessary to decode the POST register, although, a cleaner implementation would decode 16 bits of addresses. It's a write-only register and should not interfere with other registers in the address range above 0x3ff. All that is necessary to implement the decoder are the signals A0–9, SD0–7, and IOW.

Figure 1 shows the POST board that I put together. The register is a 74HCT374, although, any 8-bit register would work. The address decode is implemented using several two-input NOR gates (74HCT02) and an eight-input NAND gate (74HCT30). When a valid address and a write strobe are on the bus, the output of the NAND gate acts as the clock to store the state of the lower 8-bit data bus.

You can use whatever gates you have to implement the decoder function. Also, a couple of 8-bit comparators would work well. Even a PAL (22v10) will work well. The combination of gates I chose in this project should be available at most electronic part outlets. You can also use 'LS technology, but they are getting

harder to find these days. 'HCT is more common and performs just as well in this application.

The basic ISA bus protocol, even when used on a PC/104 card, is simplicity in itself. The processor places the addresses on the bus. When doing a write, it also places the data to write on the data bus. To write to memory or I/O, the memory write strobe (MEMW) or I/O write strobe (IOW) is asserted. The peripheral latches the data. During a read, the memory read strobe (MEMR) and I/O read strobe (IOR) are asserted, and the addressed peripheral places the data to read on the data bus.

Of course, things get a bit more complicated when doing 16-bit transfers and DMA, but POST only uses 8-bit transfers, so you don't need to worry about it at this point.

The output of the POST register is displayed on a 10-segment LED array using the outer two banks of four LEDs. There also is a 330-Ω current limit resistor on each LED. If you want, you can use some black marker and mark out the two middle LEDs. This makes it easy to decode the upper and lower nibble of the hex code.

For a fancy implementation, you might use a couple of hex-to-seven-segment decoders and drive two seven-segment registers. But hey, we're embedded systems guys, we can handle doing the conversion in our heads. I put the LED array on a right-

angle socket on the edge of the board, so I can see it from the side.

That's the basic POST board.

There's not much to it. I was able to build it using parts from my junk box. The only hard-to-find part is the PC/104 stack-through connector. In a pinch, if the card is used as the last card in a PC/104 stack, you can use a dual-row, wire-wrap header for the PC/104 connector. It just won't have the socket on top. I included a source for PC/104 connectors in the Sources section, and Photo 2 shows the board.

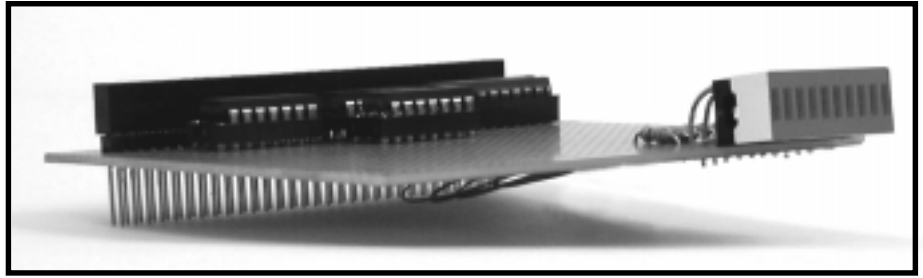


Photo 2—The PC/104-based POST board based on a hole-per-pad prototype board.

With four chips and a LED array, there is still a lot of real estate left on that PC/104 module. I could add a com-

parator circuit that would give an over/under indication of the power supply voltages (5, 12, -12 V) on the bus.

If you really want to get fancy, you might install headers that mate with your favorite logic analyzer. They can be wired to all of the bus signals. This is a handy feature to have, which you only realize after you wire up the little wires from your logic analyzer to a bus once or twice.

Reading the POST error codes should be easy now, but that's only half the battle. What do the codes mean? For this, you need to find the POST error codes that match the particular BIOS that are being used on the CPU board. Luckily, there are only a few. AMI Award BIOS is popular.

If you're using a CPU module that is specifically sold to the embedded PC market, you should be able to get the POST error codes for the module from the vendor. Or, if you know how to ask search engines the right questions, you'd be surprised what you can find.

I wasn't able to debug my laptop motherboard by the time this article was due. I still have to hunt down some of the ISA bus signals on the board and add some solder on the test points to the board. The strategy is to find a peripheral chip like a multi I/O or floppy controller with a datasheet. Find A0-9, IOW, and SD0-7, and then you'll be in business.

I did try the POST board on several CPUs and motherboards. I tried to boot it both with and without VGA cards and removed the system memory to check the codes. Sure enough, most of the time the codes made sense. But, not always.

It is possibly because I have the wrong error code list for that particular BIOS, or the error the POST detects might not be obvious. For

example, pulling the system memory might get one of the early tests to fail, and not the memory test like you would expect. This kind of failure would be different than something like a stuck-bit error in the memory.

This month, I showed you that it is easy to build your own PC/104 modules. I like building PC/104 modules because you don't need special prototype boards like you do for an ISA bus. I just find any old hole/pad prototype board, add a header, and I'm set. Also, because the protocol is an ISA bus, it's a breeze to decode and use. Unless, of course, you want to do fancy things like master DMA modes.

Now, if building a prototype PC/104-*plus* board were only that easy. PC/104-*plus* is the standard for using PCI bus protocol in PC/104 systems. The PCI signals are carried on special 128-pin, 2-mm centered stack-through connectors. The connectors are available from the same sources that PC/104 stack-through connectors are.

However, because the centers are 2 mm instead of 0.1", you can't just

use a pad/hole prototyping board. The other issue is that, to implement PCI, even on PC/104-*plus* cards, you're pretty much going to need PCI-compatible chips. PCI on PC/104-*plus* runs at up to 33 MHz and is much more complex than ISA bus.

If you'd like to see more PC/104 and PC/104-*plus* board-level type projects, send me or the editor e-mail, and let us know. There are many possible projects, I just have to be prodded into writing about them.

WHAT ELSE

One nice thing about having a POST register board in my system is that I can drive it from my application to display system status while the system is running. For example, I can display the interrupt level my software is servicing on the display. By allocating bits for a specific interrupt level, I can turn on the corresponding bit when I enter the routine and turn it off when I exit the routine. If you plug a POST board into your system, you will instantly be

able to see exactly what's going on in your system. ■

Ingo Cyliax has written for Circuit Cellar on topics such as embedded systems, FPGA design, and robotics. He is a research engineer at Derivation Systems Inc., a San Diego-based formal synthesis company, where he works on formal-method design tools for high-assurance systems and develops embedded-system products. You may reach him at cyliax@derivation.com.

RESOURCE

AMI Award BIOS POST error codes, www.u-net.com/epr/electron/issue2/feat0607.htm

SOURCE

PC/104 and PC/104+ stack through connectors

Astron
(408) 232-1100
Fax: (408) 232-1108
www.astron-us.com

Fred Eady

Under the Covers

Part 1: Get Embed(ded) with Windows NT 4.0

What's so great about Windows NT Embedded 4.0? According to Fred, it all depends on how you look at it. This month, Fred's looking at it via Ampro's Little Board/P5x SBC (which came with some other goodies, too).



I love my job. I get to play with the latest embedded PC gadgets and savor the flavor of exotic and not-so-exotic operating systems. In my business, I never know what the UPS guy will bring for me to smoke test today.

For those of you who dare to follow my adventures every month, you know that everything from tiny Internet appliances to embedded PC backplanes have mysteriously appeared on the Florida-room stoop. You've been privy to my mishaps with QNX, Phar Lap's TNT, Datalight's ROM-DOS, and most all of Bill's products. You've seen me cuss when I do something stupid and hold my hat in my hand after almost smoking an embedded SBC. You know that I like blinky LED projects, Ethernet, and Bill G.

But, I'm a little miffed at Bill this month. I've been begging for Windows NT 4.0 Embedded information for quite a while with absolutely no luck. My mom says, "Pity the rat that has only one hole to run to in times of danger." So, reading between Mom's lines, I called upon a reputable and well-known embedded PC hardware company for help.

A few days later, my trusty UPS guy delivered a large box with Ampro tape all over it to the Florida room. I

was eager to cut it open to see what goodies abounded within.

My eyes just about popped out and fell onto the Florida-room's terrazzo floor. Your eyes would have been in the dirt too if you could have been here with me to behold what came out of that box. It's an Ampro Little Board/P5x SBC that you see in Photo 1. It's even loaded with an IBM 340MB Microdrive! Hold on folks, this is gonna' be fun.

LOOK OUT, DICK TRACY

Dick always had the latest embedded technology. His two-way wrist radio is still a hit today. Looks like we have the latest in hard-disk technology from the IBM folks right here in the Florida room. Before we get into the Ampro host embedded PC, let's go back and see how the IBM Microdrive really got here.

Disk storage was coming into its time in the 1950s. On September 13, 1956, a small team of IBM engineers in San Jose, California, introduced the 305 RAMAC (random access method of accounting and control), the first computer disk storage system. This first-of-its-kind, mass-storage device could store a whopping 5 MB of data on 50 24" diameter disks. This random access technology found its way into the airline industry and the space race.

By the 1960s, IBM introduced the first storage unit with removable disks, the IBM 1311. 1966 brought in the IBM 2314, the first disk drive with a wound-coil ferrite recording head. The IBM 3735 was a prelude to the IBM 3340, which was the first disk drive to use low-mass heads, lubricated disks, and sealed assembly. Today, we all know yesterday's IBM 3340 drive as the Winchester drive. The 3340 Winchester drive featured two spindles with a storage capacity of 30 million characters each, hence the terms 30-30 or Winchester. The '70s were kind of boring, so while they weren't doing anything else, engineers at IBM invented the floppy.

The 1980s brought in the 3380 storage array, which could store 6000 times as much data as the original RAMAC arrays. Data recording densities reached 1 GB per square inch

using magnetoresistive heads. (I've walked around a few 3380s.)

1991 introduced the world to IBM's giant magnetoresistive heads and the first one-gigabit, 3.5" hard disk drive. My first hard disk was a full-sized 5.25" Seagate that held a mind-bending 5 MB unformatted. I can still remember getting my first 1-GB drive. I had no idea how I could ever fill it.

A new, world record in data density was set with 3 GB per square inch in 1995. In September 1998, IBM developed the technology to build a 1" hard disk drive platter, which would be introduced in 1999 as the Microdrive, the world's smallest hard disk drive—and I have one! As the little girl on the Shake and Bake commercials used to say.

Let's get out of the truck for a minute. For those of you not familiar with the Shake and Bake girl, she was a little girl who appeared in TV commercials for Shake and Bake years ago. The commercial centered on this little girl helping Mom cook chicken in the kitchen. When the little girl spouted her line, "And I helped!" you definitely knew she was from the South. OK, back in the truck.

As you know, that same basic RAMAC approach to storage, spinning magnetic disks, and flying read/write heads is still in use today. In fact, the RAMAC name was used again recently by IBM to identify a new line of large system storage, which packed larger disk storage capacity into a much smaller footprint. (It's a whole lot smaller than the old 3380 boxes, but it still takes a few steps to walk around it in the computer room.)

Take a look inside your desktop or, better yet, inside your laptop. Disk drives have shrunk significantly in size and increased in capacity by more than 5,000 times since the introduction of RAMAC. Think about this. The technology in the IBM Microdrive, shown in Photo 2, puts your already tiny laptop



Photo 1—You can't see it here, but there's a 3-GB hard disk in this little black box, too. This setup is a hardware heaven. There are connectors on the back of this box for everything you would ever use in an embedded situation.

drive in the technological dirt! Taking a peek at IBM's web site and, noting that the company's incorporating the Microdrive into its products, it's safe to say that the Microdrive has arrived.

To give you an idea of what kind of affect the IBM Microdrive will have, here's some of what it can do right now. The IBM Microdrive is believed to be the smallest hard disk drive shipping today.

A 40-MB Klik Disk dwarfs the Microdrive. Get three quarters together, and I'll give you some dimensions. The Microdrive is exactly three-quarters deep and approximately 1.5 x 1.7 quarters in girth. It weighs less than an AA battery, and can hold 1,000 compressed digital photographs, the equivalent of 200 floppy diskettes, or six hours of near CD-quality audio.



Photo 2—You can bet this little puppy is going to get smaller in size and bigger in density. An iOMEGA Klik Disk holds 40 MB.

AN ALTERNATE PATH

Other than getting to hold a 340-MB Microdrive in my hot little hands, there were other good reasons to call on Ampro to help me get some NT Embedded OS code for you. As you have already seen, the Ampro Little Board/P5x SBC has a few tricks up its sleeve.

The Ampro Little Board/P5x SBC uses either a 166- or 266-MHz Intel Mobile Pentium processor with MMX, known as Tillamook. Its onboard peripheral content is equal to the logic contained on six external expansion boards. It's fully PC/AT compatible, and there's more level two cache on this tiny board than on most desktops, 512 KB. Memory tops out at 256 MB and can be either a DIMM of EDO DRAM or SDRAM.

All of the required PC-compatible goodies are onboard—15 standard

interrupt channels, seven DMA channels, and three programmable counter/timers. That's equivalent to the standard Intel ICs (8259, 8237, and 8254, respectively) that you saw on PCs of yesteryear.

There are four 16550-equivalent serial ports, with two acting as RS-232 only. The remaining two serial ports can be mixed among RS-232, RS-485, TTL, or IRDA. The parallel port is standard IEEE-1284 with bi-directional data lines. I was surprised but

pleased to see SCSI capability in the onboard Adaptec ACI-7860 adapter. The standard complement of IDE drives is also supported. Of course, floppy drives are supported, and there's even a 10/100 BaseT Ethernet port just for me! I could go on for another couple of pages, but chances are, your favorite peripheral is on the list.

The Florida-room version you see in Photo 1 runs at 266 MHz and is equipped with 64 MB of RAM. All this horsepower is contained on an EBX 1.1 standards-based CPU module.

To differentiate the Ampro Little Board/P5x SBC from its desktop predecessors, Ampro has made some improvements to the architecture and firmware. Things like a watchdog timer and serial console support are essential items for true embedded operation. The watchdog timer monitors the boot process and is accessible via calls to BIOS. Serial console sup-

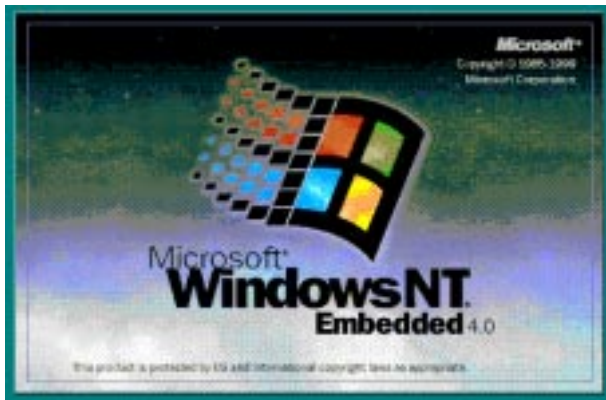


Photo 3—How about that! This is the last operating system I would have expected to see on the same page with the word “embedded.”

port is exactly that. Just hook up an ASCII dumb terminal and take charge. Ampro also has included some really neat boot options like serial boot loader, batteryless boot, fail-safe boot, and accelerated boot.

The serial boot loader allows boot code to be loaded from another external serial source. Batteryless boot is another way of saying that the system setup information is stored in non-volatile EEPROM instead of battery-backed RAM. This is used in the

event of onboard RTC/setup RAM battery failure. Fail-safe boot does it until it gets it right. Under BIOS configuration control, the fail-safe boot process retries boot devices until a successful boot is achieved. Accelerated boot is simply whizzing past the POST routines. If those boot options aren't what your app requires, you can use Ampro's BIOS extensions to customize your boot scenarios.

I fired the Ampro Little Board/P5x SBC up while I was describing the BIOS enhancements to you. Look at this, the guys at Ampro loaded Windows NT Embedded 4.0 on this little fish.

WINDOWS NT WHAT?

Many of you are probably wondering why all the hoopla over another embedded operating system entry—especially an OS that's been around for a while. What's so great about Windows NT Embedded 4.0?

I want all of you who are Microsoft Certified (MCP, MCSE, etc.) to raise your hands—OK, good. Now, all of you who write Windows desktop applications raise your hands—OK. Did you know that all of you who raised your hands for either question are potentially capable of writing embedded Windows NT Embedded 4.0 applications? Well, congratulations on your newfound talent.

Windows NT Embedded 4.0 is the latest Windows NT 4.0 technology utilizing Service Pack 5. Don't confuse this with the recently announced Windows 2000. Windows NT Embedded 4.0 is designed to allow the embedded developer to leverage his or her existing Windows programming knowledge and produce better products faster. Windows NT Embedded 4.0 is designed to allow developers to use the vast off-the-shelf Windows resources to affect embedded designs.

An authoring tool set consisting of Target Designer and Component Designer allows embedded developers to configure and generate a specific Win-

dows NT Embedded 4.0 operating system. Target Designer is used to configure a device-specific Windows NT Embedded 4.0 operating system using Windows NT binaries embedded-enabling technologies and an application.

As an option, Component Designer can be used to create reusable components that were not included in the Windows NT Embedded 4.0 product. These components usually will be device drivers or special applications. New components then can be imported into Target Designer where they are incorporated into the device operating system. Once an operating system has been configured using Target Designer, file and application dependencies are checked, and a bootable image is produced that can then be loaded onto the embedded device and executed.

WHY NT4 EMBEDDED OVER CE?

If I developed embedded Windows programs for a living, the very first thing I would ask myself is "Why do I

have to use Windows NT Embedded 4.0 for anything embedded?" If I want to embed in Bill's world, I already have other embedded operating systems, including a Windows-based Windows CE. One answer is the size of the embedded device you are writing to. What I mean by size here is size in the physical or in the hardware, such as memory size or data storage size.

Another answer is security. Will your design be used by the military or by kids on the block? Yet another answer may lie in functionality. What does your device have to communicate with? Does it participate in a LAN, or does it just dial up the 'Net and browse the web? Without bashing CE or Windows NT Embedded 4.0, let's explore the possibilities.

CE supports quite a few more processors than Windows NT Embedded 4.0. They share 'x86 compatibility, but that's about all. CE can do MIPS, ARM, and PowerPC to name a few. Windows NT Embedded 4.0 tends to lean toward Pentium class processors

like the AMD K5 and K6. Windows NT Embedded 4.0 also supports Cyrix 5x86 and 6x86 CPUs.

As far as CPU speed, CE runs the gamut. The recommended speed for Windows NT Embedded 4.0 is 200 MHz and above. You can see this in my Ampro Little Board/P5x SBC, which is clocking in at 266 MHz. I have no doubt that the Ampro Little Board/P5x SBC running Windows NT Embedded 4.0 with a 166-MHz CPU would perform well also.

CE was designed for small single-processor mobile and palm-top type embedded applications. Because Windows NT Embedded 4.0 is a binary equivalent, configurable subset of Windows NT 4.0, just like it's desktop brother, Windows NT Embedded 4.0 can handle a number of CPUs in a multiprocessor configuration. Thus, the overhead of supporting multiple CPUs makes Windows NT Embedded 4.0 bigger as far as code goes.

The same size comparison can describe the multitasking capabilities of each product. Both CE and Windows NT Embedded 4.0 do preemptive multitasking, and both support threads. However, CE is limited to 32 concurrent applications. Windows NT Embedded 4.0 is limited only by the amount of storage it can grab.

Memory requirements are another factor that makes Windows NT Embedded 4.0 big when compared to CE. Windows NT Embedded 4.0 requires a minimum of 12 MB to play in and 8 MB of available storage, if you don't factor in networking. With networking installed, those numbers increase to 16 MB of play space and additional 16 MB of available storage. CE does its thing in 1 MB of play space and doesn't require any additional storage.

There's really not much difference in usability between CE and Windows NT Embedded 4.0. Both operating systems have a gaggle of built-in utilities like Internet Explorer, Windows Explorer, and Command Shell, and both operating systems can be configured to work without the seemingly ever-present GUI.

On CE and Windows NT Embedded 4.0, this GUI-less mode is called headless support. Headless simply

means that you can generate your Windows NT Embedded 4.0 or CE system without a display. In this mode you talk to your embedded sweetheart via a telnet server or a web-based management interface.

Although the Windows NT Embedded 4.0 implementation could be run in headless mode, the Ampro system has its hat on, so I can show you things via screenshots. Speaking of screenshots, CE being optimized for smaller applications is also optimized for smaller displays. You can stop your resolution count at 800 × 600 for CE. Windows NT Embedded 4.0 starts at 640 × 480 and goes up from there.

Communications protocols for CE include TCP/IP, PPP, SLIP, PAP, CHAP, HTTP, and IrDA. Windows NT Embedded 4.0 protocols are all of those you had to know about to pass the Workstation Exam for your MCP or MCSE. The notable protocol exceptions contained in Windows NT Embedded 4.0 and not in CE are AppleTalk, Netbeui, and IPX/SPX. This says to me that if you want to network off-the-shelf with the legacy big boys, don't design your device with CE, or be prepared to write some interface code.

For those of you who design in sensitive environments, the security features offered by NTFS make Windows NT Embedded 4.0 a good candidate for your projects. NTFS allows the administrator to control access down to the file level. CE has no inherent security features.

In addition to NTFS, Windows NT Embedded 4.0 also supports FAT and compressed data formats. CE does FAT and the "Kryptonite" FAT32. (Windows NT Embedded 4.0 can't do or coexist with FAT32.) These data formats can be stored in ATA flash memory, M-Systems DiskOnChip, IDE hard disks, and bootable CDROM using Windows NT Embedded 4.0. Our Ampro Little Board/P5x SBC steps up to include CompactFlash and the IBM Microdrive mini-spinner. CE can do all of that, too.

Power management is a must on CE-equipped mobile computing platforms, and CE has features to allow for extensive battery management. On

Windows NT Embedded 4.0 platforms, the power management capabilities are generally left to the hardware. For example, the Ampro Little Board/P5x SBC offers "green PC" power-saving modes using APM (advanced power management) BIOS functions.

NT4 IN YOUR FUTURE

I've nearly run out of paper here. I did some more exploring while you were reading. Seems like the Ampro engineers felt sorry for me. Not only did they load Windows NT Embedded 4.0, I see QNX and Linux as boot options on the Windows NT Embedded 4.0 loader screen, too. Looks like the IBM Microdrive is also loaded with a version of Bill's MS-DOS. We're going to have a good time next month!

Next time, I'll continue my investigation of Windows NT Embedded 4.0 and try to run an application or two. If I can figure out how to make those other two operating systems tick, I might do something there, too. Until then, I'll leave you with Photo 3 which proves that even Bill G. thinks it doesn't have to be complicated (or CE) to be embedded. ☛

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

Little Board/P5x

Ampro
(408) 360-0200
Fax: (408) 360-0222
www.ampro.com

Windows NT Embedded 4.0

Microsoft
(206) 882-8080
Fax: (206) 936-7329

Microdrive

IBM
(407) 443-2000
Fax: (407) 443-4533

A Quick Meter Made

FEATURE ARTICLE

Markus Desgronte

Fast Frequency Meter for Low Frequencies

When it comes to measuring frequencies, as the frequency you are measuring drops, so does your accuracy. In this article, Markus shows us how to make a frequency meter that's quick and has no problem handling lower frequencies.



Since the early days of digital technology, frequency measurement has been done by counting signal cycles during a specified, fixed-gate time, which varies from 1 ms to 10 s. The number of signal cycles counted during 1 s would directly give the signal-frequency in Hz, and a gate time of 1 ms would give kilohertz, as you can see in Figure 1.

This measuring method is satisfactory for many applications and can provide instruments for measuring frequencies in a range of 10 kHz up to megahertz and gigahertz ranges. When these instruments show measuring results on a digital display, the useful measuring rate would be limited to two or five measurements per second, as a result of human ability to follow such a readout.

In terms of measuring accuracy, this method is fine when you are in high frequency ranges, such as megahertz or gigahertz, and have long enough gate times. In fact, the accuracy increases proportionally with frequency and gate time.

For example, let's assume a frequency of 10 MHz is to be measured during a 0.1-s gate time. This would deliver 1 million cycles to be counted during the 0.1-s gate time. So, the

methodical counting error would be a ± 1 count, which equals 1-ppm measuring error. When our instrument uses a time base with a 10-ppm accuracy, the time-base error would already be 1 decade above the methodical error—so, no problems with the measuring method. Although, coming down to lower frequencies directly increases the methodical error:

1 MHz	-> 10 ppm
100 kHz	-> 100 ppm
10 kHz	-> 1000 ppm
1 kHz	-> 1%
100 Hz	-> 10%
20 Hz	-> 50%

Figure 2a and b shows the principal accuracy limitation of classical frequency measurement.

So, low frequency measuring needs longer gate times (an often-used approach) or another measuring method. With an extended gate time of 10 s, the classical method would deliver an improved accuracy by a factor of 100:

1 kHz	-> 0,01%
100 Hz	-> 0,1%
20 Hz	-> 0,5%

In the real world, slow measuring is not an advantage. Slow measuring just delays feedback loops and reduces understandability in laboratory situations. A faster and better measuring method would help.

FAST FREQUENCY MEASURING

The measuring method used here improves this situation in both areas—higher accuracy and faster measurements of low frequencies.

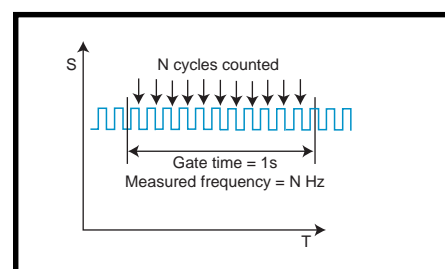


Figure 1—With the normal frequency measuring method, this is the fixed gate time (1 s) during which the number of trigger edges (N) is counted.

Actually, the measuring accuracy is constant over the full measuring range, which is another advantage of fast frequency measuring (FFM).

The improvements of FFM come from using more existing information from the signal to be measured. Besides counting the number of periods during a certain time window, FFM

also adjusts the time window to get a more accurate time measurement for the counted number of signal cycles.

The number of cycles and window time values are converted into a frequency by the quotient:

$$f(\text{Hz}) = \frac{\text{number of cycles}}{\text{window time (s)}}$$

This relation easily shows that we have the potential for high accuracy, which is independent from the actual signal frequency and can be increased as needed with increased accuracy of the time window measuring. An example of the measurement of a 3.438-Hz signal with FFM is shown in Figure 2b.

Listing 1—In a digital-readout-only version, the frequency meter consists of just a few BASIC code lines.

```
INSTALL_DEVICE #1, "LCD1.TDD"      ' text LCD 4 x 20
INSTALL_DEVICE #2, "TIMER.A.TDD",1,200 ' TIMER-A: 12.500 kHz
INSTALL_DEVICE #3, "FREQ1_84.TDD"  ' Measurement on Pin L84
...
...
PUT #3,#0,#UFCO_DEV_OPT1, 5000    ' Set gate time
PUT #3,0                            ' Start ENDLESS Measuring

FOR K=0 TO 1 STEP 0                <- top-of-endless-loop
  GET #3,#0,#UFC_IBU_FILL,0,FILLING ' get Input-Buffer: Filling
  IF FILLING > 0 THEN              ' measurement available?
    GET #3,#0,4,FREQUENCY          ' get Frequency (mHz)
    PRINT USING #1, FREQUENCY ;"Hz" ' show frequency in Hz
  ENDIF
NEXT                                ' --> loop-end
END
```

Listing 2—Defining bitmap graphics (as the "3" shown here) may be done directly in the source code. The binary format here allows the use of digits "0", "1", "*", or "." characters.

```
' --> 76543210 <-- Bit-position in bytes
DATA BYTE ". . * * * * ."B ' 0 Character = 3
DATA BYTE ". * * * * * ."B ' 1
DATA BYTE "* * * * . * * * "B ' 2
DATA BYTE "* * * * . * * * "B ' 3
DATA BYTE "* * * * . * * * "B ' 4
DATA BYTE ". . . * * * * "B ' 5
DATA BYTE ". . * * * * ."B ' 6
DATA BYTE ". . * * * * ."B ' 7 8 x 17 (w x h)
DATA BYTE ". . * * * * ."B ' 8
DATA BYTE ". . . * * * * "B ' 9
DATA BYTE "* * * * . * * * "B ' 10
DATA BYTE "* * * * . * * * "B ' 11
DATA BYTE "* * * * . * * * "B ' 12
DATA BYTE "* * * * . * * * "B ' 13
DATA BYTE "* * * * . * * * "B ' 14
DATA BYTE ". * * * * * ."B ' 15
DATA BYTE ". . * * * * ."B ' 16
```

Listing 3—Here are the two codelines responsible for the logarithmic-scaled analog bargraph.

```
LOGFREQ=22*LOG(FREQUENCY/1000) ' scale for logarithmic display
...
...
' DESTIN W1, H1, X, Y, W2, H2, MODE
GRAPHIC_FILL_MASK (SCREEN2$,240,128,31,97-LOGFREQ,11,LOGFREQ,0)
```

THE REALIZED INSTRUMENT

In this article, I describe how to build a frequency-measuring instrument using FFM with just two components needed—a LCD and a BASIC Tiger computer module. An optional amplifier for non-TTL signals may be added (see Figure 3a).

For best readability, I am using a high-resolution graphics display, which shows the frequency values in both an analog and a digital form. The logarithmic analog display reflects the constant relative accuracy over the full measuring range as well as the digital display, which always displays four relevant digits (1.000 Hz to 4999 Hz measuring range). The BASIC Tiger computer is used for the measuring process and the user interface as well.

The measuring part of the frequency meter is covered by device drivers, which are supplied in the BASIC Tiger development environment. Your job is to install the device drivers, do the range settings, and get the measured frequency values from the device driver.

The elementary program needed to measure frequencies continuously and display results on a text display is shown in Listing 1.

The device driver responsible for frequency measurement is installed with install_device instruction:

```
INSTALL_DEVICE #3,
  "FREQ1_84.TDD"
```

which defines Port 84 as the input pin for measuring. Device driver TIMER.A is installed and started as the corresponding timebase. The following two PUT instructions are setting driver specifications as measuring rate and count. The GET instructions in the endless FOR-loop are check-

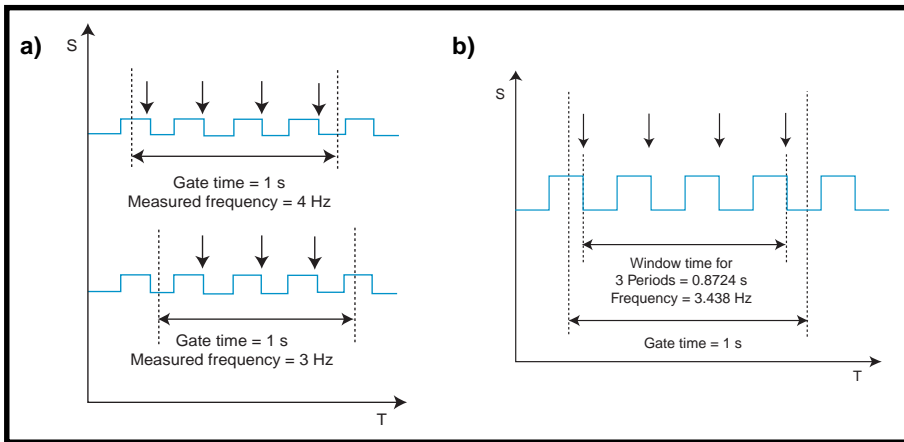


Figure 2a—The shortcoming of the normal frequency measuring method is that the same signal might be measured as a 3- or 4-Hz frequency. **2b**—FFM measures the real window time within the gate time, resulting in much more accurate frequency measurements.

ing the input buffer for availability of more measurements and getting the next measurement.

In the real project, you would replace the simple PRINT USING output to a text display with a better looking graphics output (see Figure 3a).

PLANNING THE LAYOUT

BASIC Tiger allows the generation of moved, high-resolution graphics to be output on a device such as a LCD.

This feature gives this instrument a well readable and informative appearance. I have selected a T6963 LCD with 240 × 128 pixel, a 6" × 2.6" viewing area, and CFL illumination for best black-and-white contrast.

The instrument layout here is already designed for four channels to be displayed simultaneously. Depending on the application, I might install the frequency device driver several times for up to four measuring chan-

nels, or I might use free space on the display for other information if I think of expanding this application.

Graphics in a BASIC Tiger program can either be bitmap graphics or vector graphics in Cartesian or polar coordinates. I'm using both techniques to make up the construction of the instrument to be shown on the LCD.

This instrument is made up of several components that are overlaid and displayed together on the LCD. Included is a black and white bitmap graphic, which is 240 × 128 for the instrument background. This can be designed with any graphics package on a PC and exported as a black and white bitmap format (.BMP).

Also there is an analog display (in the form of a bar graph), which is drawn with the FILL_MASK function and the digital reading, which is always a 4-digit result with differently set decimal points, depending on value. (The Y-size of the bar is proportional to the logarithmic value of the measurement.)

To get big, readable digits here, you do not use the built-in character set of the display. Instead, you use pre-defined bitmap graphics for digits 0 through 9 and the decimal dot. These pixel arrays for each digit are moved into the screen as needed.

Figure 3b shows you the instrument layout and the three components it's made of.

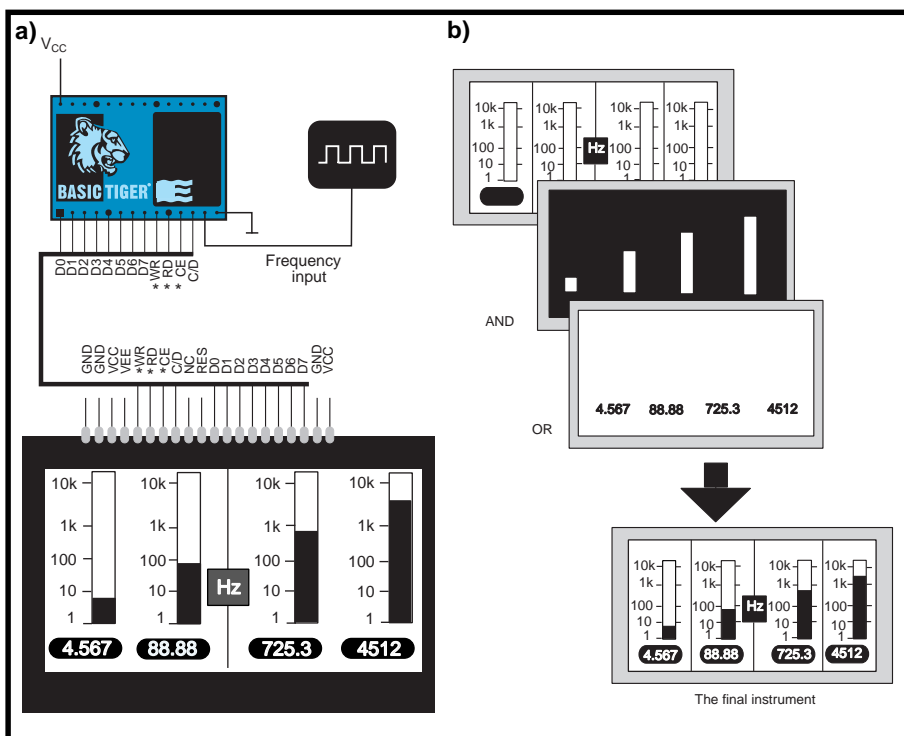


Figure 3a—The instrument consists of just two components: a BASIC Tiger multitasking computer and a graphic LCD. The TTL signal is directly connected to a Tiger I/O pin. **3b**—As seen in this example of a 4-channel analog and digital meter layout with logarithmic scaling, several graphical layers can be overlaid through AND, OR, and XOR functions.

BITMAP AND VECTOR GRAPHICS

Bitmap graphics in a BASIC Tiger program can either be imported from *.BMP files, defined in DATA instructions, or generated from a BASIC Tiger program. To import the instrument background, a BMP-file, the DATA instruction is used with the filter GRAPHFLT,0 as follows:

```
DATA FILTER
"MESS_HZ.BMP", "GRAPHFLT", 0
' Meter graphics
```

If you look at Listing 2, you'll see that to define the set of numerical characters, you need to use DATA instructions in a pixel format, where "*" represents 1 bit (black pixels) and "." represents 0 bits (white pixels).

The example shows the digit “3” in an 8 × 17 grid, which is chosen for its best fit into the available space. To make up a 4-digit number, you are subsequently moving pixel data from the character definition area in flash memory into the screen area, SCREEN\$ (a string), in RAM. The commented program code in the listings shows the details.

For displaying variable bar graphs, use vector graphics. Vector graphics are open to any kind of calculations, such as scaling, rotation, shifts, and transformations. In this case, I am showing bar graphs proportional to the logarithmic value of the measurement, using the `FILL_MASK` instruction shown in Listing 3. The mask generated by this `FILL_MASK` function is a white bar on a background area in `SCREEN2$`.

As you’ve seen, this project demonstrates the construction of a smart instrument with minimal hardware and programming overhead. With only a tiny multitasking computer (\$49) and a graphic display (\$20–\$80), you can have a sophisticated instrument at your fingertips. ☒

Markus Desgronte is an electronics engineer who, when not fighting bugs and glitches, enjoys traveling and spending time with his family. You can reach Markus by phone at ++49 (241) 918 9031, by fax at ++49 (241) 918 9068, or via e-mail at desgronte@wilke.de.

SOFTWARE

The full commented program is available for download at www.wilke-technology.com.

SOURCES

BASIC Tiger ENN-4x

Wilke Technology
+49 (241) 918 900
Fax: ++49 (241) 918 9044
www.wilke-technology.com

LMG7420 (LCD)

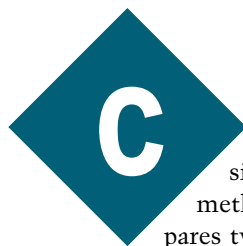
Hitachi
(800) 448 2244
(415) 589-4207
Fax: (415) 583-4207

All About Correlation

FEATURE ARTICLE

Ron Tipton

If you need to perform signal analysis, Ron takes a graphical approach to explaining the details of correlation. By the time you finish this article, you'll be more than ready to download the software and start working with these data-analysis tools.



Correlation is a signal analysis method that compares two time sequences to find out how alike they are. If the two signals are identical, they are said to have a correlation coefficient (R) of unity. If they are completely different, they don't "correlate" at all, and R is equal to 0. This coefficient (R) is a dimensionless number, whose magnitude varies between zero and one.

So, how do you find this coefficient, and what practical use does it have? The method is probably best understood by looking at it graphically.

Suppose you have two sine waves with the same amplitude and frequency drawn on separate pieces of transparent film, placed one on top of the other. If you hold the bottom film stationary and slide the top film to the right or left, you have a simple correlator (see Figure 1).

When the two signal traces coincide, $R = 1$. When they are 90° out of phase, $R = 0$. (Mathematically, you have a coefficient of -1 when they are 180° out of phase, but for practical reasons, you'll only look at the 0 to 1 interval.)

Although it's not obvious yet, this is a powerful analysis method. You

can construct numeric band-pass filters with narrow passbands for recovering periodic signals buried in noise. Applications include audio reproduction, acoustics, sonar, seismic event detection, and many others.

The examples in this article can be run on a personal computer, and you will be able to use your PC to model virtually any real application by scaling the "real world" signal frequency. Then you can program a digital signal processor if you are building some special-purpose hardware or need the processing speed.

Yes, correlation is a mathematical procedure, so you will have to take a look at one equation to get started:

$$R(m) = \frac{1}{N} \sum_{n=0}^{N-1} x(n)y(n+m)$$

Fortunately, you can relate this equation to the previous graphical example. $x(n)$ is the sequence of amplitude values in the signal on the bottom film, $y(n)$ is on the top film, and m is the variable that describes moving the top film in steps of the signal sampling interval, n . Each correlation number (R) is just the normalized sum of the x signal multiplied by the y signal.

AUTO- AND CROSS-CORRELATION

If you use the same signal for both x and y , you have auto-correlation (i.e., a signal correlated with itself).

Cross-correlation is using different x and y inputs, and this can provide up to a 20-dB improvement over auto-correlation in pulling a signal out of the noise. This improvement isn't free, so you need to know approximately what the signal looks like without the noise. Let's look at a few examples of

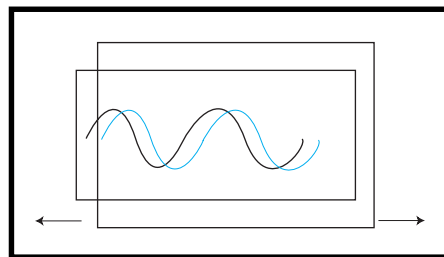


Figure 1—You can make a graphical correlator by plotting wave forms on two sheets of transparent film. Sliding the top film right or left is equivalent to solving the correlation equation.

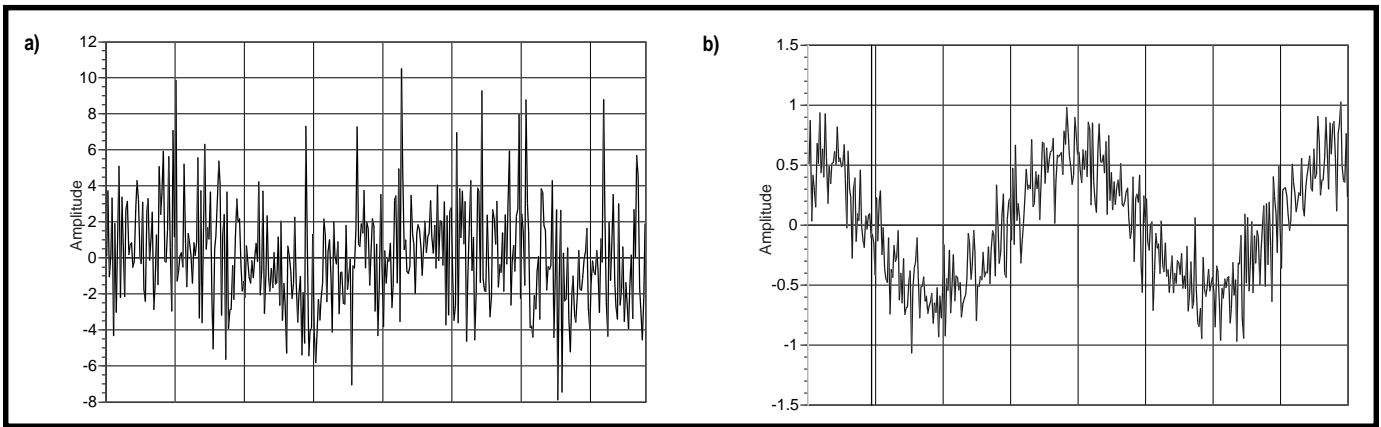


Figure 2a—Here are two cycles of a sine wave with 99% RMS added noise. Auto-correlation (**b**) does not require any knowledge of the signal, but recovery is poorer than is possible with cross-correlation.

auto- and cross-correlation.

Figure 2a shows two cycles of a sine wave with added random noise. The RMS value (see the Root Mean Square sidebar) of the noise is equal to the RMS value of the sine wave, so visually it looks like noise. If you don't know the wave shape and frequency, the best you can do (at least initially) is auto-correlate. Figure 2b shows the results of auto-correlation. You start to see the signal, but it's still noisy. However, if you know the frequency, you can cross-correlate the noisy signal with a noise-free sine wave and get the improvement shown in Figure 3a.

These correlations were done with program `correlat.exe`, a DOS program written in C. The well-commented source code, along with the executable, can be found in `correlat.zip` on the *Circuit Cellar* web site. A companion program,

`gensine.exe` with the C source code, is included to generate input files with and without added noise. The programs are written in MIX Software's PowerC, but they should be easily portable to other compilers if you want to make changes.

So, what does `correlat.exe` actually do? After asking for input from you, it finds the maximum correlation coefficient and displays it on the screen, along with the corresponding delay between the two inputs. Then, using this delay time, it correlates the input signals and writes an ASCII output file with each value terminated by a new line character. The output file is named either `AUTO.DAT` or `CROSS.DAT` depending on which operation was chosen.

`Gensine.exe` also writes the same kind of output file, which makes these files easy to import into a math program (such as Mathcad) for graph-

ing the data in reports and other presentations. The output file is named `SINE.DAT`, and it will be written again every time you run the program. So, if you create files containing varying amounts of noise, rename them something descriptive between runs. I wrote all of these programs as development tools, so they lack some of the refinements you expect in commercial software.

To show how powerful cross-correlation can be, I generated an input file setting the RMS value of the noise to be 10 times the RMS value of the sine wave, which is 1000% noise. Auto-correlation shows not a hint of the buried signal, but cross-correlation pulls it out neatly, as you can see in Figure 3b.

Cross-correlation also can be looked at as a type of digital or numeric band-pass filter, and this accounts for its remarkable performance

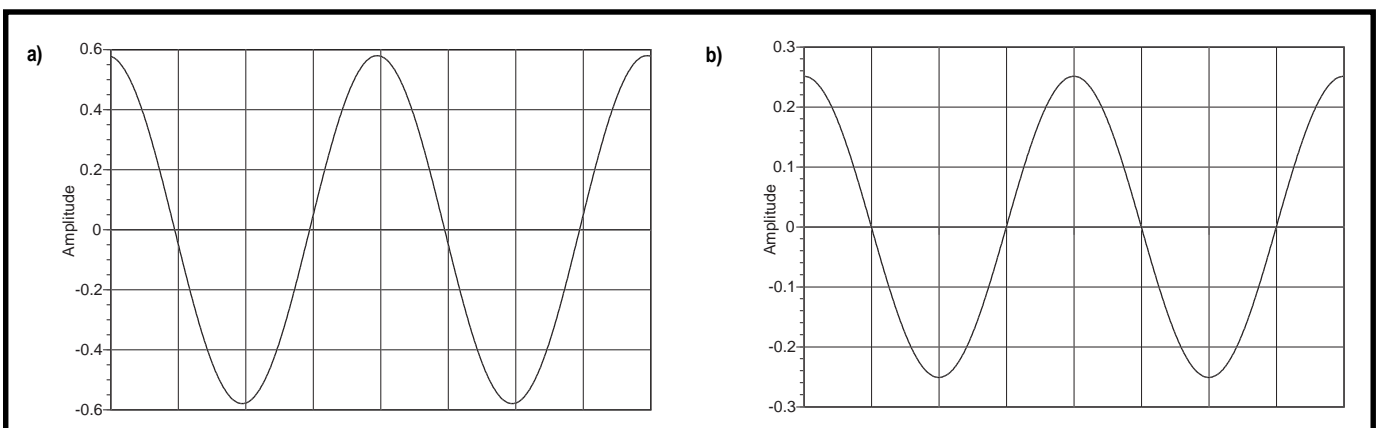


Figure 3—This is the output of cross-correlating a noisy sine wave with a “clean” reference. In **a**), the input had 99% added noise, and in **b**), there was 1000% noise added. Cross-correlation is a powerful tool for pulling signals out of the noise if you can construct a reference signal.

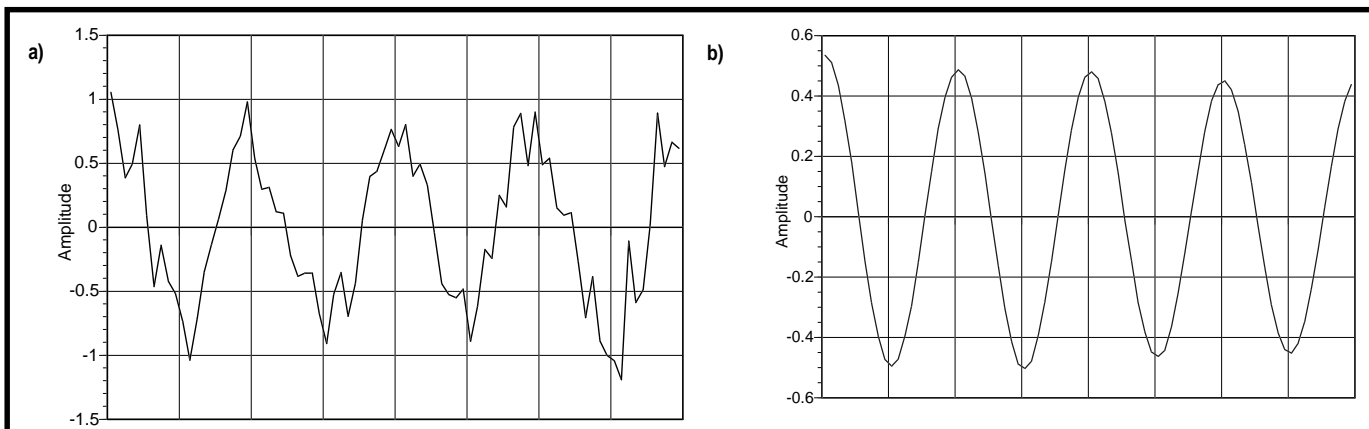


Figure 4—Notice the effect of using a slow sampling rate. This (a) is what auto-correlation of a signal with only 50% added noise sampled at 20 points per cycle looks like. The cross-correlation output (b) still shows some distortion from the low sampling rate.

in pulling a signal out of noise. This article isn't meant to be about digital filters, but I included a brief explanation in a second sidebar for those of you who are interested (see the Digital Filters and Correlation sidebar).

In the previous example, I used a sample rate of 200 points per sine wave cycle because it shows dramatic results! A good rule of thumb: Use the highest sample rate you can, within practical limits. As you'll see in a later example, with today's analog to digital converters (ADCs) and DSPs, the highest practical rate can be pretty high.

So, what happens if you have to use a slower sample rate? Figure 4a shows an auto-correlation output of a sine wave with 50% RMS added noise sampled at only 20 samples per cycle. Still noisy, but you can start seeing the periodic signal. By looking at zero crossings, you can estimate the signal's frequency, which let's you use cross-correlation to clean it up as best you can (see Figure 4b). When the signal frequency is unknown or guessed at, simply vary the noise-free signal frequency to minimize the signal distortion out of the correlator.

PHASE ANGLE RECOVERY

Sometimes the only information needed is the phase angle between two signals at the same frequency, where one signal is corrupted by noise. Cross-correlation excels here. Because the phase measurement resolution depends on the sample rate,

you may need some fast ADCs.

For example, I had to update a data-collection system, which required me to design a cross-correlation phasemeter. The original phasemeter was a counter—one input signal started a high frequency counter, and the second input signal stopped it. The accumulated count was a measurement of the phase difference between the two input signals.

This works well when both signals are "clean," but typically, one of the inputs varies between noisy and extremely noisy. Because it's a constant frequency system, performance is optimized by passing the noisy signal through a narrow band-pass filter and then averaging a number of successive phasemeter counts.

Typically, 8 to 48 measurements are averaged and this improves the phase difference estimate by the square root of the number of averaged counts (N). Meaning, the standard

deviation of the jitter is reduced by the square root of N , but the dynamics of this system prevent increasing N to more than 48.

Counter phasemeters also have a problem that occurs when the phase difference is near 0° or 360° . When the input signal is noisy, successive counts may jump erratically between zero and full scale. Solving this requires additional logic in the design. However, the correlation meter does not have this problem.

To get an idea of how much improvement I could expect from the correlation phasemeter, I wrote another C language simulation program, `simphase.c`. Along with other parameters, the program asks for the signal amplitude and the amount of noise to add (in RMS percent), which lets us set the input signal to noise ratio (SNR). The second (clean) input is read from a data file generated by the companion program, `genref.c`.

Root Mean Square

The root mean square, or RMS voltage, is a measure of the energy in a signal. Meaning, one RMS ampere flowing through a resistance produces the same amount of heat as does one DC ampere. For a sine wave, the RMS voltage is equal to 0.707 times the peak voltage.

The RMS value is literally the square root of the sum of the squares of the time-sampled values. Calculating it is an easy way to find the RMS value, if you are looking at the output of an analog to digital converter (ADC). Just square each value and accumulate a running sum for at least as long as the period of the lowest frequency of interest, and then take the square root.

True RMS voltmeters usually use a nonlinear circuit to approximate the relationship between average and RMS. And a few (such as the Hewlett-Packard model 3403C) use a thermal AC to DC converter.

(These programs are included in correlat.zip, which is available for download via the Circuit Cellar web site.)

After playing with the simulator for a while, I was convinced that the correlation phasemeter would yield a significant improvement in performance, so I designed and built a prototype. (The clock rate is too high for a successful breadboard, so I laid out a printed circuit board for the initial model.)

The incoming noisy signal is digitized at an 8.5-MHz rate, and the resulting 12-bit words are stored in RAM. (This is 1700 samples per input cycle for a phase resolution of 0.212° .) RAM-1 and RAM-2 are used in a Ping-Pong technique, with RAM-2 being filled while RAM-1 is processed (see Figure 5).

The read and write enable signals for both RAM banks are generated by a timing circuit, which relieves the DSP from having to keep track of

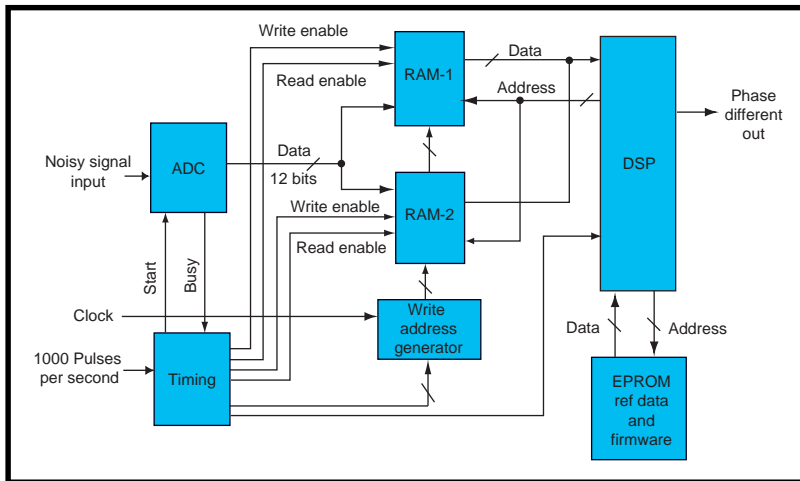


Figure 5—The clean input (reference signal) is read from the EPROM. This meter generates a phase difference output every millisecond using a Motorola fixed-point DSP with a 50-MHz clock.

which RAM bank it is processing. The second (reference) signal values are read from EPROM. Program execution is started by an interrupt signal every millisecond from the timing logic.

The DSP firmware is a bit complex and is divided into two parts. The first part reads the data from RAM-1 or RAM-2 and estimates the count “z” to the next negative-to-positive zero crossing. This count is used as a starting point for the second part of the program, the cross-correlation.

A correlation is run, z is incremented, and another correlation is run. This process is continued until a peak in the correlation coefficient is found. The phase difference or answer is just the delay time corresponding to this maximum. The amount of time saved by starting with an estimated z is crucial to having this phasemeter

run fast enough.

When the input signal is badly corrupted by noise, the zero-crossing estimate may be in error by several counts. Therefore, the second correlation estimate can be smaller than the first one, which means you’ve missed finding the peak. When this happens, the original z is decremented and used as a new starting point until a peak is found. A

peak is always found in no more than seven correlations, even at minimum SNR. Figure 6 shows the output SNR for both a counter and cross-correlation phasemeter.

NON-PERIODIC SIGNALS

Correlation works well in recovering sine waves from noise, because the energy is concentrated at a single frequency, and the noise is broadband (at least when compared to the bandwidth of the signal). A noisy square wave is not recovered well because its signal bandwidth is too broad. The same is true for noisy pulse trains.

Remember that correlation can be thought of as a narrow band-pass filter, so the signal must have most of its energy within this passband. However, a matched filter will do the job.

Digital Filters and Correlation

If the time domain waveform shape is known, an optimal data recovery filter is the convolution between the noisy data and the data’s “non-noisy” shape. This means, the filter’s coefficients are the noise-free, time-sampled data values in time reverse order.

Time reversal is the only difference between cross-correlation and convolution. If the signal is a sine or cosine wave, then there isn’t any practical difference between the forward and reversed time values, so correlation and convolution are the same.

Convolution is sometimes called a “matched filter.” A Finite Impulse Response (FIR) digital filter is one way to perform the convolution.

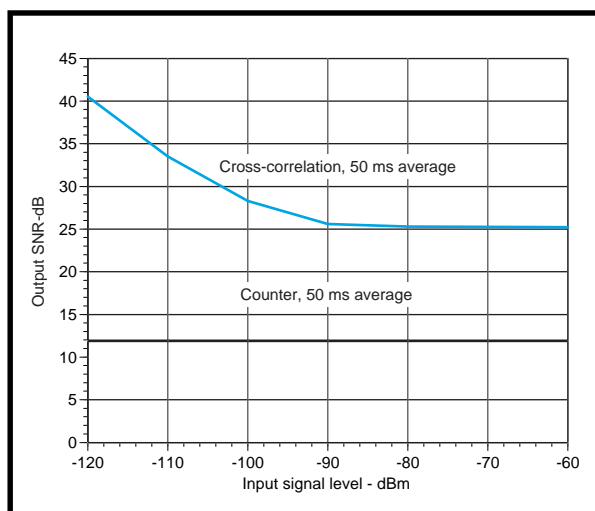


Figure 6—The cross-correlation phasemeter provides nearly 30-dB SNR improvement for low input signals and is always better than a counter phasemeter. The higher cost of the cross-correlator is justified when performance is a critical requirement.

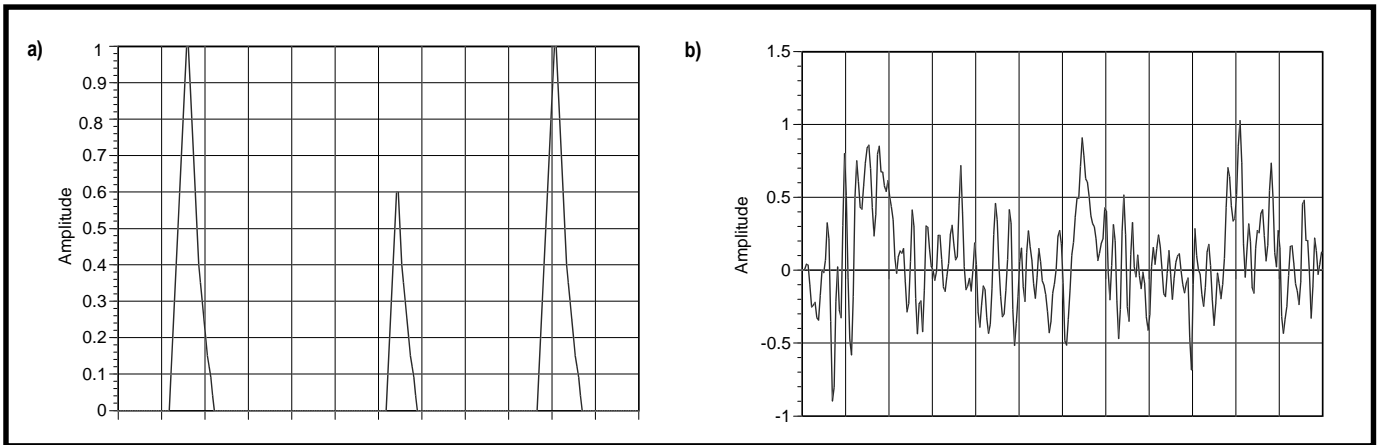


Figure 7—Here you can see an example pulse train (a) and these pulses buried in noise (b). The 30% peak noise was low-pass filtered at 20 kHz by program `genpulse.exe` before the addition.

Figure 7 shows a test group of pulses that I buried in the noise using program `genpulse.exe`. This program assumes a sample rate of 100 kHz, so the noise is low-pass filtered in the program through a 2-pole Butterworth with a 20-kHz cutoff frequency. These filter coefficients are coded into the program, but they can be changed for a different filter if needed. (This filter was designed with the Momentum Data Systems software listed in Resources.)

The matched filter coefficients for recovering the pulses are in file `filter.coe`. They are the sampled amplitude values of one of the pulses in time reversed order and normalized so they sum to unity. Meaning, each coefficient has been divided by its original sum. This is also a plain ASCII file that can be created or changed with any ASCII file editor.

The noisy data file and coefficient file is read by `fir_bp.exe`, which performs the filtering. The result is shown in Figure 8, and the recovery is

good. You can try adding different amounts of noise and then look at the results. If you look at the source code, you will see that the “process” routines look the same in `correlat.c` and `fir_bp.c`. Mathematically, correlation and convolution are similar, even though they have different uses.

I suggest that you play with the programs. It’s a good way to get a practical appreciation and feel for the data-analysis tools. As I mentioned earlier, they lack commercial polish, but work well and run fast on a Pentium II-class computer at clock speeds of 200 MHz or more. You can also take routines from them to use in writing your own solutions. ☒

Ron Tipton is an engineer with more than 40 years experience in analog, digital, and software design. He is the president of TDL Technology, Inc., a company he started that specializes in consulting and prototype development. Reprints of his other magazine

articles are on the TDL web site. You may reach him at rtipton@zianet.com.

SOFTWARE

All the software mentioned is in `correlat.zip`, which is available on the *Circuit Cellar* web site and also on the TDL site at www.zianet.com/tdl.

RESOURCES

- R.H. Higgins, *Digital Signal Processing in VLSI*, Prentice Hall, Englewood Cliffs, NJ, 1990.
- K.G. Beauchamp, *Signal Processing Using Analog and Digital Techniques*, John Wiley and Sons, 1973.
- C.B. Rorabaugh, *Digital Filter Designer’s Handbook: Featuring C Routines*, McGraw-Hill, NY, 1993.

SOURCES

PowerC Compiler

Mix Software
 (800) 333-0330
 Fax: (972) 783-1404
www.mixsoftware.com

Mathcad

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

Digital filter software

Momentum Data Systems
 (714) 378-5805
 Fax: (714) 378-5985
www.mds.com

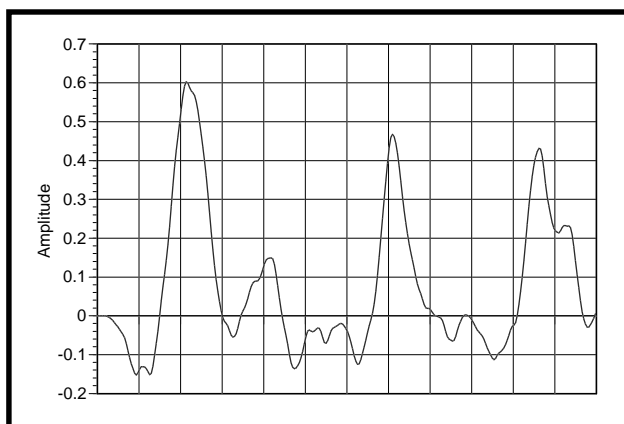


Figure 8—This is what the output of a convolution filter with the noisy pulse train of Figure 7b as the input looks like. Note the pulse time delay as compared to Figure 7a. This is real or “causal” filter even though it’s done in software rather than a physical circuit.

MICRO SERIES

Joe DiBartolomeo

Op-Amps Specifications

Served Italian Style

Part
1
of
4

If you're a digital or software designer who only delves into the realm of analog and op-amps when absolutely necessary, grab a cappuccino and a pastry, and then get ready for an earful on op-amps. *Andiamo!*



My neighborhood has finally become civilized—good cappuccino is now within walking distance.

I'm not talking about one of these bookstore coffee shops where people hush you because they are reading. I mean a real old-fashioned Italian bakery with gelato and pastries.

A place where kids are screaming and an old lady in the back makes food that will let you forget anything. A place where the Italian culture, with all of its beauty and warts, is out in the open and on display.

As I know first-hand, an Italian family bestows on its members a number of things: unconditional love, brutal honesty, and high-decibel conversations. Growing up, when you did something wrong, you could be assured of being yelled at until your ears rang, then suddenly getting hugged, kissed, and a double helping of your favorite pasta.

Why am I telling you all this? Well, normally I e-mail ideas and drafts of articles to friends for their input. The comments I get back from the group are normally short and nondescript.

For this series, however, I just happened to be sitting with friends at my favorite Italian bakery. When I

mentioned my idea for a series on op-amps, I got an earful. I guess the bakery brought out the Italian in them.

The response from one of my friends was unexpected. To paraphrase: "You, you stupid or something? What's with the op-amp? It's eight pins, two inputs, one output, and a couple of power pins. Three of the pins aren't even used. There's too many op-amps. What's the big deal? Foor-get about it."

Another of my colleagues basically admitted that he had a few op-amps that he was familiar with, and depending on the application, he'd just select one.

Now, I'd never get this kind of honest and loud response in the office. Could you imagine sitting in a meeting with two engineers saying such things? I guess coffee doesn't bring out the honesty like a good cappuccino (not to mention a veal sandwich and a few pastries).

I was, to say the least, surprised by their reactions. I know they use op-amps—did they really believe what they were telling me? Did they really design this way?

Then it dawned on me—my friends are basically digital/software designers who, because of a lack of resources, are forced to delve into the black-magic world of analog. Furthermore, op-amps have come a long way in recent years. Those three pins that do nothing on today's op-amps used to have a purpose.

There has certainly been a shift in engineering jobs towards the digital. Given this, it is not uncommon to find digital/software engineers taking on the analog role. An engineer who doesn't use op-amps on a regular basis could be left wondering why there are so many op-amps.

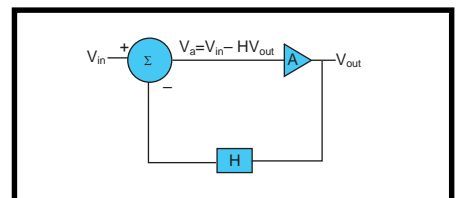


Figure 1—The amplifier gain (A) is large, $\gg 100K$. The feedback is applied in such a way so as to reduce the output. This will provide for constancy of gain while reducing nonlinearity and distortion.

A digital engineer may only want an op-amp to follow a current-output DAC. That person may find the selection overwhelming. But even if you have worked with op-amps, a great deal of the selection process is non-trivial. In fact, it may be said that the more you know about op-amps, the more time consuming your selection process.

When I first envisioned a series on op-amp specifications, I was thinking about heavy-duty, in-depth analog stuff meant for analog designers. However, I changed the target audience after talking to my friends and looking at *Circuit Cellar's* subtitle: *The Computer Applications Journal*.

It became clear to me that there are a lot of digital designers out there who are applying computers to the analog world. It is this group of engineers that I want to address.

This four-part series is about op-amp specifications—what they are, what they mean, and what to watch for. Op-amp basics are presented also. The specifications of op-amps are meaningless without a basic understanding of the op-amp itself.

The series is intended as a basic review for people who use op-amps, but don't have time to go back to their textbooks and read Op-Amp 101. If you've been working heavily with op-amps for many years, there probably isn't much in this series for you. For that I make no apologies.

Part 1 gets us started by discussing some general topics and basic op-amp concepts. In the next three articles, I'll look at op-amp specifications such as bias currents, offset voltages, and input and output impedance.

THE OP-AMP

The name "operation amplifier" comes from the original use of these modules in analog computers. The only analog computer I've ever seen was brought into class by one of my professors as a novelty.

The analog computer consisted of several operational blocks—adders, multipliers, integrators. Basically, an input analog signal was routed through the operational blocks to solve a differential equation.

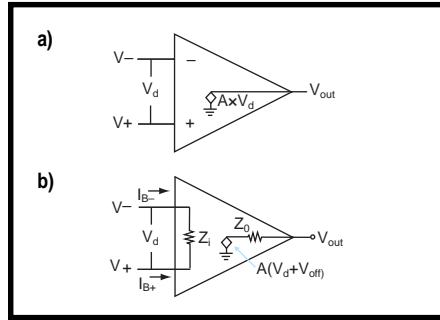


Figure 2—The ideal op-amp (a) has infinite gain, no output impedance, and infinite input impedance. Non-ideal op-amps (b) have both input and output impedance and non-infinite gain.

These operational blocks contained amplifiers, hence the name op-amps. The op-amps first were made with tubes, then with discrete transistors, and finally in the monolithic form seen today.

The op-amp is the fundamental building block of analog electronics. It is used in a range of applications from simple buffering of signals to complex nonlinear applications. Because of the wide variety of applications, there are a huge number of op-amps manufactured by several companies.

DATASHEETS AND SPECS

What helps the engineer to select a particular op-amp? The main source of information is the manufacturer's datasheet.

The datasheet is the source of a great deal of debate. Engineers have been known to accuse manufacturers of supplying misinformation on their datasheets, while manufacturers claim that the information is extremely difficult to present.

One of the problems with datasheets is that manufacturers and engineers use them differently. To the engineer, the datasheet is an engineering tool. To the manufacturer, it is a marketing tool as well as an engineering tool.

Engineers need to understand this dual purpose. The datasheet must market the product and at the same time provide real,

supportable engineering information. After all, someone might measure the specifications of the device.

As I mentioned, when it comes to op-amps, the information is difficult to present. All op-amp specifications vary depending on temperature, operating frequency, and load. The best manufacturers can do is provide performance charts, tables, and graphs.

With so many op-amps and specifications out there, the first place to start narrowing down your choices is the application. The broadest demarcation is whether you have an AC or DC application.

In DC applications, precision and accuracy are important. Specifications such as bias currents and offset voltages are of concern, but the speed of the op-amp is not as important.

In AC applications, the speed of the op-amp is important. You need to pay attention to slew rates and bandwidths, but the DC performance is not as critical.

The engineer must collect the information from the datasheet and apply it to the application in question, and it is here that the engineer must be careful. The data is there, but it must be interpreted properly. To do this, the engineer has to understand what the specifications are as well as the test conditions under which the specifications were measured.

NEGATIVE FEEDBACK

One concept I want to briefly look at is the concept of negative feedback. Today, we have no problem with the idea, but when it was invented, it was novel.

In 1928, Harold Black applied for a patent for negative feedback. Amplifiers of the day suffered from incon-

	Ideal	Real-world
A	Infinite (infinite gain)	80–170 dB
BW	Infinite (no frequency effects)	1 kHz–1 GHz
$V_{in}(V_d)$	0 (virtual short)	
Z_{in}	Infinite	$10^6 \Omega$, 2–20 pF
Z_{out}	0	1–1000 Ω
I_{b-}, I_{b+}	0 (no current into op-amp input pins [input bias currents = 0])	30 fA–250 nA

Table 1—This is a comparison of the most common op-amp specifications for both ideal and real-world op-amps.

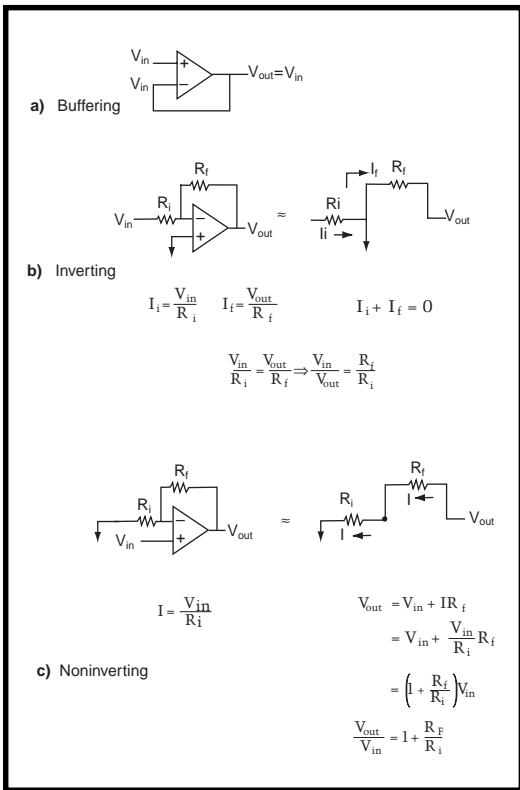


Figure 3—There are three basic op-amp circuits—(a) Buffering, (b) Inverting, and (c) Noninverting. Note how the ideal op-amp model can be used to remove the op-amp from the circuit.

stancy of gain, nonlinearity, and distortion. How could feeding back some of the output to the input, in such a way as to reduce the output, help?

In fact, the whole idea appeared somewhat silly. According to Black, when he tried to patent the concept, “Our patent application was one for a perpetual motion machine.”[1] Well, regardless of what the patent-dispensing experts thought, let’s take a closer look at negative feedback anyway.

A portion of the output signal is fed back and subtracted from the input. This process reduces your gain, but in return you get constancy of gain and freedom from nonlinearity and distortion.

As the negative feedback is increased, the system characteristics depend less on the amplifier’s characteristics

(open-loop gain, no feedback) and more on the feedback network itself. The fact that you give up some gain is easily compensated for by starting with much more gain than is needed. Op-amps typically have open-loop gains greater than 100,000.

What does negative feedback look like mathematically? Taking a look at Figure 1, notice the amplifier with a large open-loop gain (A). There is a feedback network with a transfer function (H) and a summer that provides the negative feedback by subtracting the output of block H from V_{in} . Equation 1, representing Figure 1, is our starting point.

$$V_{out} = (V_{in} - HV_{out})A \quad [1]$$

$$V_{out} = V_{in}A - V_{out}HA$$

$$V_{out}(1 + HA) = V_{in}A \quad [2]$$

$$\frac{V_{out}}{V_{in}} = \frac{1}{\frac{1}{A} + H} \quad [3]$$

$$\frac{V_{out}}{V_{in}} = \frac{1}{H} \quad [4]$$

Expanding equation 1, we get equation 2. Gain is defined as output over input. Manipulating equation 2, we get the expression for gain shown in equation 3.

Recall that the open-loop gain is made purposely larger than needed. In fact, in an ideal op-amp, A is infinite. Taking the limit as A tends to infinity, we get equation 4. As long as $A \gg 1$, the gain of the amplifier depends only on the feedback network.

For simplicity, I did not consider the frequency dependence of H and A . To account for frequency, rewrite A and H using the Laplace notation $A(s)$ and $H(s)$.

When you stop and take a look at negative feedback today, it looks straightforward. However, I suspect Black's colleagues must have given him a hard time. He applied for the patent in 1928 but did not publish openly until 1934.[2]

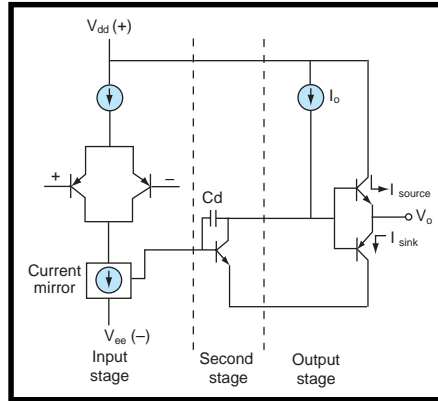


Figure 4—The op-amp is made up of three stages. The input stage converts the differential input voltage signal into a single-ended current signal while rejecting any common mode signal. The second stage converts this current into a voltage and provides frequency compensation. The output stage provides drive and low output impedance.

IDEAL OP-AMP

Figure 2a shows the equivalent circuit of an ideal op-amp. Three basic assumptions are made with the ideal op-amp:

- gain A is infinite
- input impedance Z_{in} is infinite
- output impedance Z_{out} is zero

The op-amp is a differential to single-ended amplifier. The output is simply $A \times V_d$. Any common-mode voltage—that is, voltage that appears on both inputs—is completely rejected.

From the above statement, many things are derived. No current flows into the op-amp's input terminals, so Z_{in} is infinite. Therefore, V_d is equal to zero. These two results, when combined with negative feedback, lead to the concept of a virtual short.

Through the feedback path, the op-amp reacts to the input changes in such a way as to keep its inputs at the same potential.

Notice I wrote “virtual short” and not “virtual ground.” We’re so used to seeing one of the op-amp input terminals tied to ground, and therefore setting both inputs at ground, that these terms get confused. The left-hand column in Table 1 summarizes the ideal op-amp and what it leads to.

Figure 2b is a real-world op-amp, and some typical values are given in the right-hand column of Table 1. (There are many other op-amp specifi-

cations not shown here. The information is intended to contrast with that of the ideal op-amp.)

It would appear that the ideal op-amp model is not very useful, but in fact it is. The ideal model is often used for first-order calculations. Now, let's take a look at this ideal model with a simple example, using feedback of course.

The op-amp has three basic configurations—inverting, noninverting, and buffer. Figure 3 shows that because no current flows into the op-amp input terminals and the op-amp's gain is infinite, the circuit response is determined solely by the feedback network.

Thus, we can essentially remove the op-amp from the circuit calculations. The ideal op-amp model works well as a first approximation.

INSIDE THE OP-AMP

The op-amp can be divided into three stages—input, second, and output (see Figure 4).

The input stage is a long-tailed structure consisting of a high-gain DC-coupled differential input amplifier, a current source, and a current mirror. The purpose of the input stage is to amplify the difference signal present at the op-amp input terminals, while rejecting the common signal.

This stage turns the differential-voltage signal into a single-ended current signal that drives the second stage. Symmetry of the input stage is critical to its performance, as I'll illustrate later in this series when I discuss offset current, bias currents, and offset voltages.

The second stage is a trans-resistance amplifier converting current into voltage. The second stage also provides frequency compensation. Frequency compensation is required so that the open-loop phase shift is less than 180° at all frequencies where the gain is greater than 1. If this is not the case, you get positive feedback and oscillation.

The most common method of compensation is dominant-pole compensation, achieved in Figure 4 by adding C_d . This moves the first pole of the op-amp (i.e., the 3-dB point) down in frequency toward the 20-Hz range and

holds the phase shift at 90° over most of the passband. I'll examine this arrangement more carefully when I discuss phase margins and oscillation in later articles.

The final stage is a typically AB push-pull amplifier. The output stage provides drive current and low-output impedance for the voltage coming from the second stage.

WHERE WE'RE HEADED

So, to start out the series this month, I covered both ideal and real-world op-amps, and you saw that an ideal op-amp model is a useful tool.

I also looked inside an op-amp and saw that it is made up of three stages—input, transresistance, and output. And, I mentioned some of the op-amp specifications to be examined more closely in following articles in this microseries.

Next month, I will look at the op-amp input stage and discuss specifications like input offset currents and voltages as well as input bias current. These are termed DC specifications because they are important in low-frequency and precision applications but aren't of significant concern in high-speed or RF applications.

Finally, there's one more thing I think you should keep in mind. The next time you have a meeting or want some real feedback on something, go to an Italian bakery. There, as I discovered, you're more likely to have an honest discussion with your colleagues. And even if you don't, the food is sure to beat your regular coffee and donuts. ☒

Joe DiBartolomeo, P.Eng, has more than 15 years of engineering experience. He is currently employed by Texas Instruments as an analog field engineer. You may reach him at j-dibartolomeo@ti.com.

REFERENCES

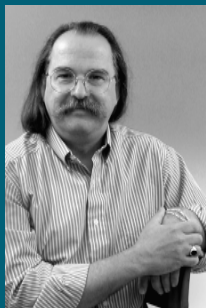
- [1] T. Horowitz and W. Hill, *The Art of Electronics*, Cambridge University Press, Cambridge, 1989.
- [2] J. Karki, *Understanding Operational Amplifier Specifications*, Texas Instruments white paper sloa011, 1998.

FROM THE BENCH

Drop the Incredible Bulk

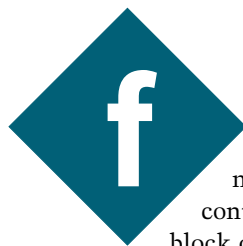
Using Capacitors as Isolation Components

Jeff Bachiochi



If your application needs a modem,

you may want to consider the Si2400. Before you scoff at 2400 bps, Jeff has some pretty good support for the less-is-more argument.



For years, all modems have contained a great big block of iron. A 600- Ω transformer was used to match POTS (plain old telephone service) line impedance and to form the basis of isolation protection.

It is certainly worthwhile for us to protect those linemen who keep our telephone system from deteriorating. And, when a power line falls onto the telephone cables, we (or our equipment) are protected from the hot juices. I'm not one for more government, but safety is an important issue.

The data access arrangement (DAA) is a circuit that performs all the necessary functions to provide a

safe interface between your equipment and the POTS. Recently, Silicon Labs designed a silicon DAA with a flexible design, allowing it to cover the individual needs of the global phone community.

Although public telephone and telegraph (PTT) compliance is universal, manufacturers of global equipment must comply with the different standards of each country. Until global standards become universal, SL's programmable line interface makes compliance with the FCC, CTR21, JATE, and other PTT compliance organizations a whole lot easier to deal with.

As a next logical step, Silicon Labs designed a complimentary embedded modem to take full advantage of its silicon DAA. The Si2400 ISO modem is the first all-silicon modem solution. Right now, you might be thinking to yourself, "A 2400-bps modem, who'd use that?" And if all you need is a faster way to move huge amounts of data, I'd agree. But as you will see, for many applications that just isn't so.

SILICON TO THE RESCUE

Silicon Labs' modem touts a 75% reduction in space, a 30% cost saving, and a 50% reduction in power. Their's big brag'ns! But, is there any truth to it? I do concur that over the standard PC card modem or its external universal RS-232 brethren there is a substantial reduction in parts count, which obviously reduces cost and power usage.

The Si2400 ISO modem integrates a microprocessor, DSP, AFE, DAA, and voice codec into this unique two-

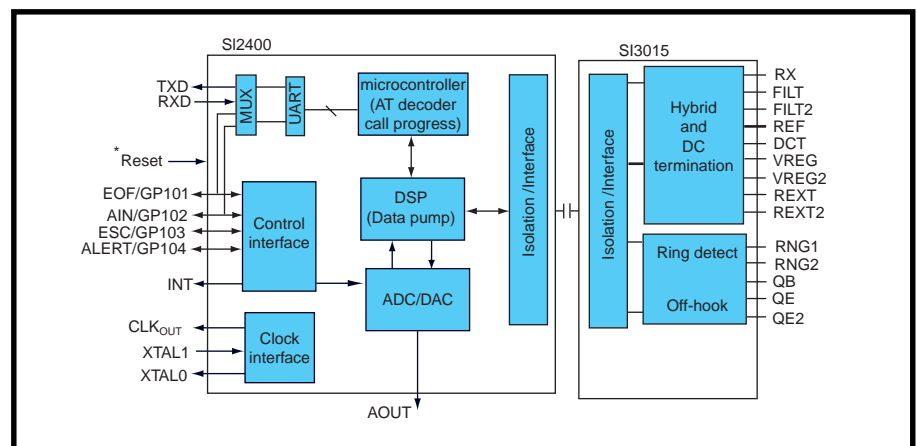


Figure 1—The Si2400 modem chipset handles communications across the 3-kV isolation barrier between the modem and the interface chips.

chip set. Surely, the introduction of a smaller modem allows products that were already waiting in the wings for new technologies to burst forth. Not all products rely on such a level of miniaturization, however, and costs may preclude product feasibility.

What kinds of applications are the targets for such a device? Well, any product that needs to communicate data to an external site, in this case over the POTS. Besides the obvious advantages of lower parts costs over the higher transfer-rate modems, it's the actual connection time that makes these lower transfer-rate modems any competition at all to their higher transfer-rate counterparts.

If you've ever listened to your PC make a connection to your ISP, you have undoubtedly heard the awful sequence of screeches necessary to make a handshake connection between the two modems. The sequence of the search for matching rates begins with the slowest of the communication rates and tries all possible standards until it matches the one of the calling modem.

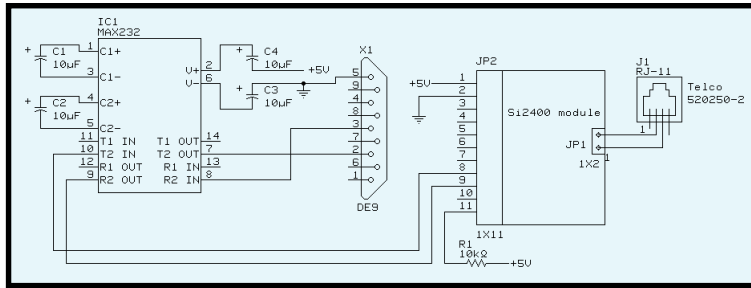


Figure 2—Although the Si2400 has additional I/O, adding an RS-232 converter and modular phone jack is all you need to connect the module to a PC.

With the newer and faster modems at the end of the sequence, a lot of time (actually, only seconds) is wasted before a connection is established. When the data that must be passed is low, the actual time for data transmission can be less than the connection time of faster modems. This means a slower modem can establish connection and complete the data transmission before a faster modem has even established a connection. A shorter phone call not only means a smaller phone charge per call but also the ability to handle more connections per hour.

The products that can take advantage of these benefits are numerous. They include point-of-sales terminals like ATMs, gas pumps, and so on, which require credit card verification, and industrial and medical monitoring

like set-top boxes, utility metering, and security systems.

I've designed products in the past that have almost become obsolete, because the low-speed modem used isn't in production anymore. The particular company that made this modem didn't under-

stand the continuing need for this kind of product. They, like many other manufacturers, left for the higher priced PC market. It looks like someone at Silicon Labs saw the need.

SIZE DOWN, FEATURES UP

You'd think that with this simplification would come a less elaborate feature set. That's not the case here. Silicon Labs has increased its potential market by designing the DAA to be internationally compliant. The DAA incorporates parallel handset intrusion detection by monitoring the POTS line voltage (when on-hook) and line current (when off-hook).

DAA parameters can be tweaked, affecting the DC and AC termination, ring detection, ringer impedance, off-hook intrusion, and current limiting to comply with individual country

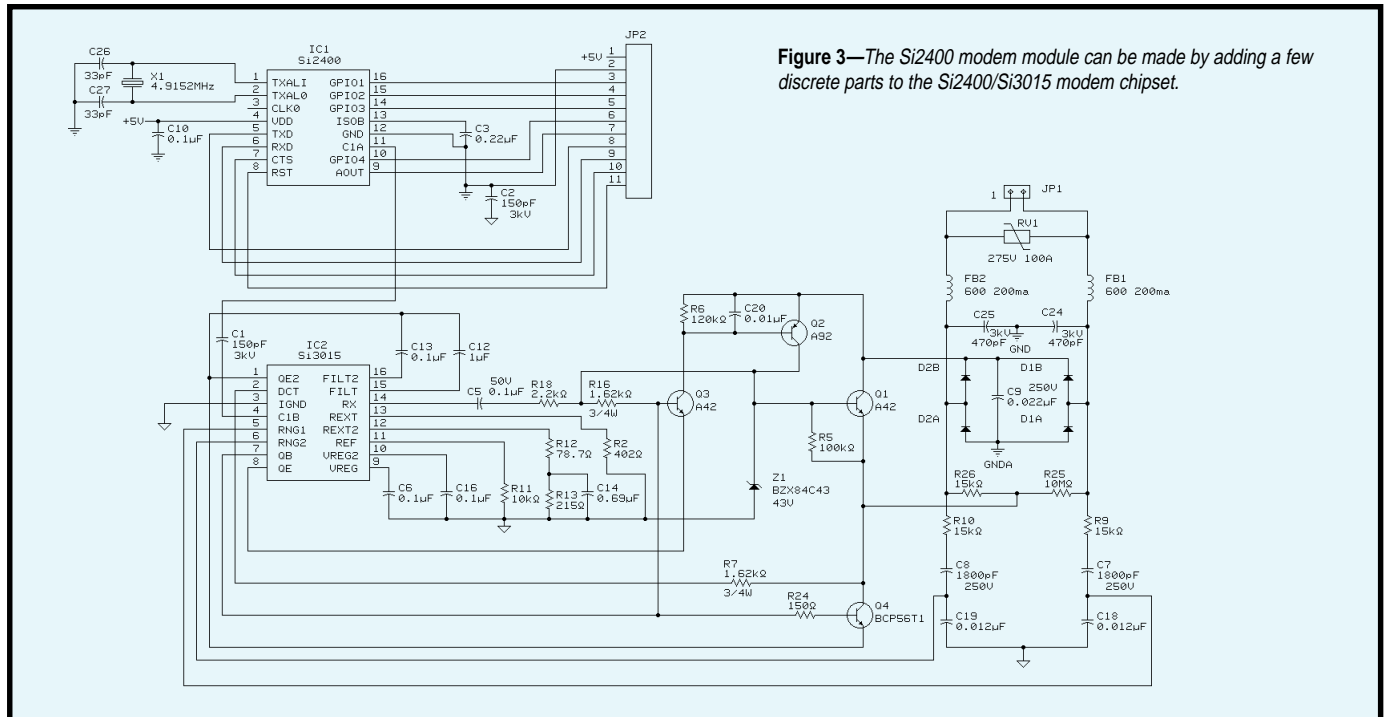


Figure 3—The Si2400 modem module can be made by adding a few discrete parts to the Si2400/Si3015 modem chipset.

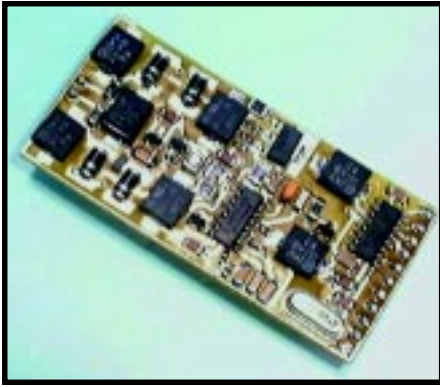


Photo 1—This module was easily mounted on a prototype board to create the PC interface circuit shown in Figure 2.

specifications. Transmit, receive, and control data is passed to the DAA from the data modem chip through a capacitor isolation barrier. The data modem automatically sets up the parameters for the U.S. However, they can be reached through the data modem's S-registers.

The data modem can use many data formats, including V.21/Bell 103, 300 bps; V.22/Bell 212A, 1200 bps;

V.22bis, 2400 bps; V.23/Bell, 1200 bps; V.23 reversing; and V.25-based fast connect. Fast-connect mode assumes both originate and answer modems are preconfigured, such that no negotiation is necessary.

In addition, Security Industry Association (SIA) pulse, generic digital, and other security protocols are supported. (Hmm, I think I smell another project here somewhere.) Both DTMF generation and detection are supported. The full 16 DTMF characters are supported (unlike a phone's dial pad). Also, Caller ID detection and decoding are supported for many countries. Call progress tones are decoded through the use of BI-quad filters.

Silicon Labs provides a good breakdown of the ring and busy cadences used by many countries of the world. Take one look at Table 1 and you'll see just how non-standard the world's phone system is. All numbers in Table 1 are 10-ms time units. Each on/off time listed has a delta time associated with it. The cadence can be off plus or minus this delta.

DRESS CODE

Because one of the main objectives of this chipset is to reduce the overall size of a modem, the Si2400 chipset comes in a one-size-fits-all packaging (two 16-pin SOIC packages). The chipset supports both 3.3- and 5-V operation with power consumption ~50 mW at 3.3 V.

Although up to eight signals can be interfaced, TX and RX are the only requirement if connection transfer rate is the same as the interface transfer rate. The power-up default rate is 2400 bps. The serial interface rate can be changed via an S-register, but the user must use the CTS handshaking line if the interface rate is anything other than the modem rate.

A reset input is available to totally restart the modem set. Four GPIOs (general purpose) can be programmed for digital I/O or alternate functions. Alternate functions include audio input, alternate pins for TX and RX (possibly for adding a second processor), an escape input, and an alert output. As digital outputs, these pins

Country	Ring (on)	Ring (off)	Ring (delta)	Busy (on)	Busy (off)	Busy (delta)
Australia	7	3	1	37	37	2
Austria	18	93	4	30	30	2
Belgium	18	56	4	50	50	2
Brazil	18	75	4	25	25	2
China	18	75	4	35	35	2
Denmark	14	140	4	25	25	2
Finland	14	93	4	25	25	2
France	28	65	4	50	50	2
Germany	18	75	4	50	50	2
Great Britain	6	4	1	37	37	2
Greece	18	75	4	30	30	2
Hong Kong
New Zealand	7	4	1	50	50	2
India	7	3	2	75	75	2
Ireland	7	4	1	50	50	2
Italy	18	75	4	50	50	2
Norway
Israel
Netherlands
Thailand
Switzerland
Japan, Korea	18	37	4	50	50	2
Malaysia	8	4	1	35	65	1
Mexico	18	75	4	25	25	2
Portugal	18	93	4	50	50	2
Singapore	7	4	2	75	75	2
Spain	28	56	4	20	20	2
Sweden	18	93	4	25	25	2
Taiwan	18	37	4	50	50	2
US, Canada (default)	75	37	4	50	50	2

Table 1—This list of ring and busy cadences for various countries shows just how far we are from a world standard.

can sink 40 mA (20 mA at 3.3 V), plenty for status LEDs.

ISO BARRIER

The Si2400 modem is a chipset because of its unique isolation barrier. A capacitive barrier is used instead of the usual transformer isolation. A

chipset is necessary to handle both sides of a 1-bit data bus (see Figure 1).

On the host side, the Si2400 modem chip has a DSP datapump, which handles the high-speed, bidirectional communication via an ISO cap interface. The same ISO cap interface is built into the line side Si3015 DAA

Command	Function
A	Answer the Line Immediately
DT#	Tone Dial (#=number to dial)
DP#	Pulse Dial (3=number to dial)
E	Local Echo (E0=on, E1=off)
H	Hangup (go off line)
I	Return Product Code + Chip Revision
M	Speaker Control Options (M0=off, M1=on until carrier, M2= on, M3=on after last digit dialed until carrier)
O	Pickup (go on line)
RO	V.23 Reverse
S	Read/Write 'S-register' (S##?, S##=##, where ## is 2 hexadecimal digits)
w##	Write 'S-register' in Binary (wbbbbbbbbbbbbbb, where bbbbbbb is 8 binary digits)
r#	Read 'S-register' in Binary (rbbbbbb, where bbbbbbb is 8 binary digits)
m#	Monitor 'S-register' in Binary (mbbbbbbb, where bbbbbbb is 8 binary digits)
Z	Software Reset
z	Wakeup on Ring (sleep until ring)

Table 2—These are a subset of AT commands supported by the Si2400 modem module.

Resultant code	Function
B	Bury Tone Detected
r	Ringback Detected
R	Incoming Ring Signal Detected
N	No Carrier Detected
O	Modem OK Response
K	SAI Contact ID Kissoff Tone Detected
c	Connect
d	Connect 1200bps (V.22bis modem)
S	Resending SIA Contact ID Data
H	Modem Automatically Hanging Up
f	Hookswitch Flash or Battery Reversal Detected
x	On-hook Intrusion Detection Detected (phone off-hook)
l	No Phone Line Detected
L	Phone Line Detected
x	Over Current Detected After an Off-hook Event
l	On-hook Intrusion Completed (phone on-hook)
m	Caller ID Mark Signal Detected
a	British Telecom Caller ID Signal Detected
v	Connect 75bps (V.23)
w	Si2400 Woken Up
^	Kissoff Tone Detection Required
,	Dialing Complete
t	Dial Tone

Table 3—These resultant codes are returned by the Si2400 modem module in response to AT commands or line status. Be careful, upper and lower case letters contain unique codes.

chip, which handles DC/AC termination, ring detection, and codec.

Transmit, receive, and control data all pass through the ISO cap interface. Because the chipset is highly integrated to provide specific functionality, the transparent interface can be viewed only as a means to isolation. The individual chips are essentially useless on their own. Although, let it be noted that a simplified DAA chipset is available for those who need to roll their own.

COMMAND SET

Similar to other standalone modems, the Si2400 supports a subset of the AT command set intended for use with a dedicated microprocessor. Commands are surrounded by an AT packet, with "AT" preceding the command and ending with a <CR> (carriage return). See Table 2 for the command set summary.

The modem also has a set of single-character resultant codes to communicate its status to the host processor. See Table 3 for a list of these codes.

SIMPLE PC INTERFACE

Although the Si2400 chipset is designed for embedded applications, you can make a quick RS-232 inter-

face for direct connection to a PC by adding a level shifter (MAX232) and a RJ-11 (see Figure 2). With this configuration, you can experiment with the modem using your favorite communication program. In Photo 1, you can see the small surface mount module I designed (see Figure 3).

I am quite excited about this new approach to low-cost, low-speed modems. Silicon Labs has indeed been thorough in its design for global compliance. Allowing the designer or user to alter interface parameters through registers via the AT command really makes global compliance an easy beast to conquer. This should set them up nicely as fierce competition for the growing embedded modem market. ☒

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

SOURCE

Si2400 modem chipset
Silicon Laboratories, Inc.
(512) 416-8500
www.silabs.com

SILICON UPDATE

Tom Cantrell

A Winter Timer Tale



The weather outside may not exactly

be frightful, but that won't stop Tom from settling down with his *TTL Cookbook* and reflecting on what made the '555 so delightful. It's not as old-fashioned as you think.



h, the winter season, curling up in front of the fire with an exciting datasheet and the new scope Santa left under the tree. Except here in Silicon Valley, where the weather's so nice I feel like I should be out enjoying it.

Now, before all you snowbirds get envious and pack up to head this way, keep in mind that the nice weather won't be much solace in a few months when folks watch their grass die because they used their meager water ration to wash the car. Anyway, these days "getting out" has devolved into sitting in traffic jams with all the other misguided people.

I'll stick with the datasheet and scope. I don't need the fire, though. Proving my wife wrong, I turn to the

dusty bookcases filled with yellowing silicon dreams and aspirations, many subsequently fulfilled but just as many dashed. Admittedly, some of these books haven't been cracked in years (make that decades), but I can't bring myself to chuck them.

I certainly can't take the risk of throwing out a baby like the *TTL Cookbook* by Don Lancaster, with the bath water.[1] Yes dear, you can get away with tossing some obscure items while I'm not looking (Come to think of it, where is that Z8000 manual?), but don't touch my *TTL Cookbook*!

A classic of its time (1974), the *TTL Cookbook* helped tutor an entire generation of neophyte bitheads, providing hours of hands-on entertainment in an era when other diversions were the Brady Bunch and disco.

GOOD VIBRATIONS

Though quaint by today's System-on-Chip standards, the venerable '555, discussed in the *TTL Cookbook*, was a workhorse that found its way into all manner of early digital designs. Remarkably, the same '555 from 25+ years ago still sells today, testimony to the timeless practicality and elegance of the design.

Checking under the hood (see Figure 1), you find a few dozen transistors. The comparator on the right drives the OUTPUT low when the THRESHOLD input reaches two-thirds the supply voltage.

The OUTPUT can also be driven low directly with the RESET input. The low OUTPUT turns on the DISCHARGE transistor. When the TRIG-

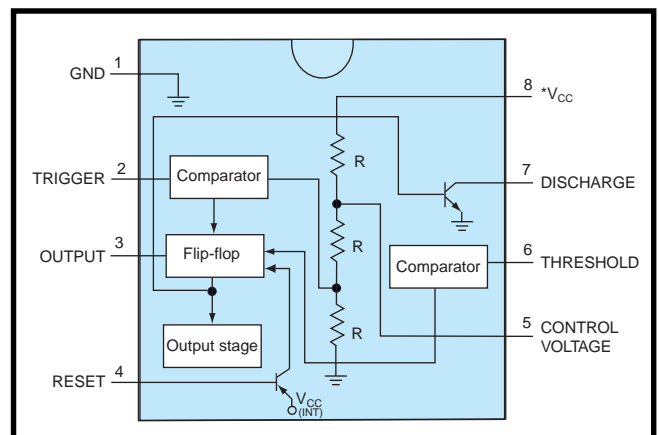


Figure 1—Timing is everything they say and, though old as the hills, continuing popularity proves that the '555 takes a licking and keeps on ticking.

GER input goes below one-third the supply voltage, the OUTPUT is driven high, turning off the DISCHARGE transistor. So simple, yet with the addition of a few external resistors and capacitors, so versatile.

Probably the most common application is simply an oscillator, also known in the nomenclature of yore, as an astable multivibrator. By selecting the proper values for two external resistors and a capacitor, you can get anything from close to 1 MHz down to fractions of 1 Hz, with the duty cycle of your choice.

Of course, a '555 oscillator isn't nearly as accurate as a crystal (few percent versus fraction of a percent error) because of tolerances and temperature characteristics of the resistors and capacitors, not to mention variations resulting from a particular PCB layout.

To counter these effects, it's quite common to see a trimpot used in place of one of the resistors to allow the timing to be fine-tuned. In its favor, once dialed in, a '555 is remarkably immune to supply voltage drift, because timing is determined by a voltage ratio rather than by absolute reference.

Wire up the '555 slightly differently and you get a monostable multivibrator (also known as a one shot) that, as the name implies, delivers one cycle of the timed output in response to the trigger. Other circuit tricks and tweaks can turn a '555 into a frequency divider, pulse-width modulator, pulse-position modulator, linear ramp generator, and so on.

Not to say the '555 is perfect. Datasheet specifications and practical limitations (i.e., sticking with standard catalog values) for the external resistor and capacitor cramp the '555's style a little. Also, the range of adjustment (roughly a few microseconds to a few seconds) is relatively wide, but no one would complain if it was wider.

Finally, the ubiquitous trimpot makes for cost, board space, and packaging headaches, not to mention the hassle of manual adjustment.

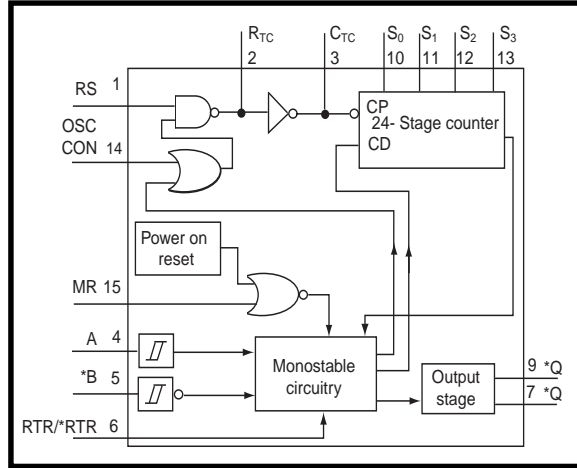


Figure 2—One of the first of a new breed of digital '555s, the 74HCT5555 freshens the design while remaining true to the spirit of the original '555.

WELCOME IN THE NEW

Over the years, a number of better '555s have been introduced. For example, wading through the bookcase again, I find the appropriately numbered 74HCT5555 in a 1993 Philips databook (see Figure 2). Old-timers may recall that Signetics, acquired by Philips some years back, was a TTL powerhouse, and Philips has carried that legacy forward.

The '5555 is similar to the '555 in principle, but in practice, it has major improvements. Like other would-be '555 successors, timing is digital, relying on a counter rather than analog circuits.

In the case of the '5555, it's a 24-bit ripple counter, although only sixteen stages (the first and last eight) can be selected. Much like a modern MCU, the oscillator (pins RS, RTC, and CTC) can be driven by an RC, a crystal, or an external input. The oscillator can handle from 1 Hz to 4 MHz using an RC, and from 32 kHz all the way to 20 MHz with a crystal.

Thanks to eight extra pins, the '5555 offers the luxury of both active high and low outputs (Q, *Q) and trigger inputs (A, B), in addition to the reset input (MR). The RTR/*RTR configures the device as re-triggerable or not and OSC_CON determines whether the oscillator runs at all times (no startup delay), or only when triggered (low power).

There are some catches. Because accuracy varies depending on the oscillator frequency and divide ratio, it's safe to say that the digital '5555 bests the analog original

in terms of both accuracy and range.

I stumbled across another novel pseudo-'555 when I was surfing the Seiko web site to check out the latest on their 7600A embedded 'Net chip (Silicon Update, *Circuit Cellar* 111). I made a quick cutback when I noticed the S-8081B CR Timer (see Figure 3).

The '81B is a simpler variation than the '5555. Although it is digital, the 20-stage divide ratio is fixed, so timing is completely determined by the external resistor and capacitor.

While the '5555 is faster than a '555, the S-8081B is known for being slower. Make that much slower—the datasheet's specified limits for the resistor and capacitor accommodate a timing range from 10 s to 10 h. Of further merit, power consumption is low at only 200 μ A, making the chip a good match for battery-driven apps.

DAY-TRIPPER

Enough of this paper chase and the nice weather, I'm itching to wire something up. Fortunately, Santa

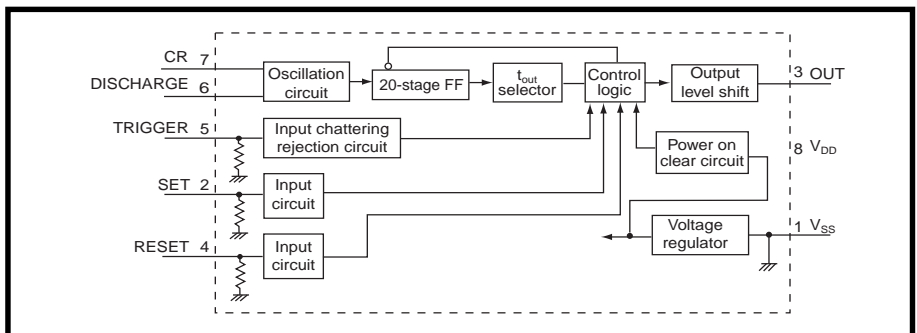


Figure 3—Long duration timeout (10 s to 10 h) and low-power (200 μ A) are the Seiko S8081-B's claim to fame.

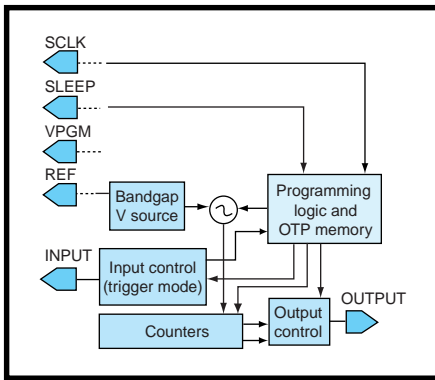


Figure 4—The latest in a long line of '555 wannabes, the Zilog ZSBI050 incorporates several advances: wider range, clock serial interface, and OTP configuration.

came through with delivery of a shiny new ZSBI050 from Zilog (see Figure 4). Like the other chips, the basic timing for the OUTPUT is established by an external resistor on the REF pin, but the similarity stops there.

The '050 is unique in three major ways. First, as shown in Figure 5, the complement of dividers is increased to five. Three fixed (two divide-by-1024 and a divide-by-32) dividers can be individually enabled or disabled. The output of the fixed dividers feeds a pair of 10-bit fully programmable divide-by-n counters, the ratio between the two, thus setting the duty cycle.

Second, rather than dedicating pins or compromising the number of selections (as in the '5555), the '050 utilizes a clocked serial interface (SCLK and SDATA) for setting the divide ratios and other operating parameters. Finally, the '050 is kind of an OTP '555, in that it incorporates fuses that allow a configuration to be permanently burned into the chip.

Thirty-five bits of memory define the operating modes and timing. Until programmed, the chip powers up in a default mode (all 0s; i.e., 50/50 oscillator at maximum frequency) using RAM for the configuration. It's an easy matter to pump in 35 bits to configure and calibrate the chip for subsequent OTP programming.

The latter process involves individually shifting in and programming each 1 bit to limit current flow during programming. The programming voltage is 7 V, but because a dedicated VPGM pin can be left open during normal operation, it's easy to accommodate production-line programming.

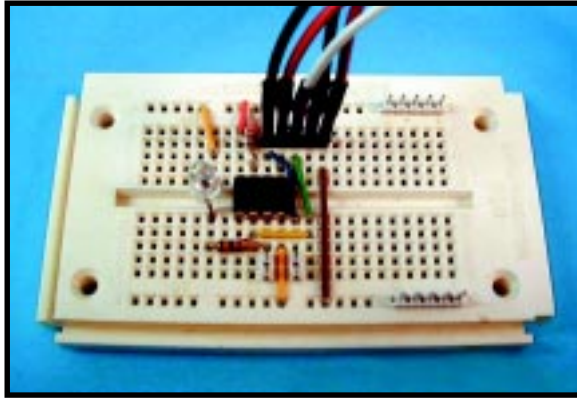


Photo 1—Eight pins and a simple clock serial interface make the ZSBI050 an easy design-in.

In one-shot mode, SDATA acts like the trigger input, configurable for rising, falling, or both edges. In oscillator modes, SDATA is a gate input that enables or disables OUTPUT.

Depending on the value of the REF resistor, the oscillator frequency ranges from about 10 kHz to 4 MHz. The OUTPUT frequency is determined by the formula:

$$F_{\text{out}} = \frac{F_{\text{osc}}}{(2 \times \text{divide-by-}n) + 4} \times K$$

where K is the product of enabled fixed dividers.

Plug in the numbers and you'll find the possible timing range covers a huge amount of territory, anywhere from a microsecond to more than a month!

TIME-TRACKER

With a rather sparse preliminary datasheet in hand, I had to kind of feel my way with the chip. On the surface, the

'050 seemed simple enough with only 8 pins and 35 bits to deal with. However, I've found there's nothing like actually wiring up a chip and getting it working to raise (and answer) the questions the datasheet overlooks.

Verifying the chip is alive and kicking is easy enough. Just hook up 5 V and GND and hang a resistor between REF and GND. On one page of the document, it says the '050 "requires no external components other than one fixed resistor." Yet, in the AC/DC specs, it alludes to the need

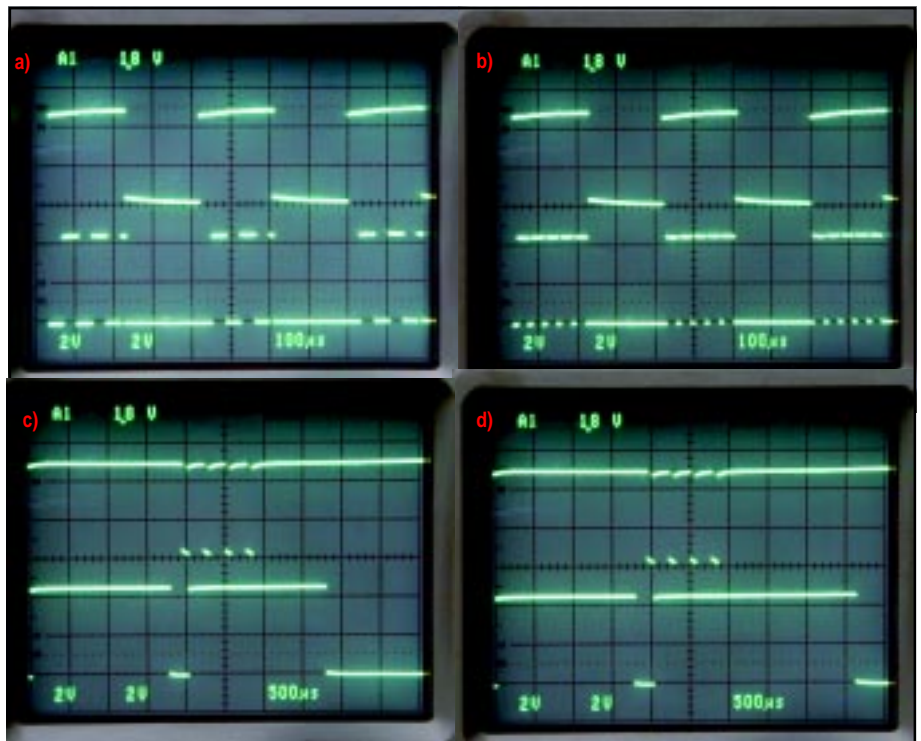


Photo 2—The ZSBI050 in action. The top trace is SDATA, and the bottom trace Output. (a) and (b) show the '050 configured as an oscillator (50/50 and variable duty cycle, respectively) with SDATA used as a gate input. (c) and (d) show non-retriggerable and retriggerable one-shot modes with SDATA as the trigger.

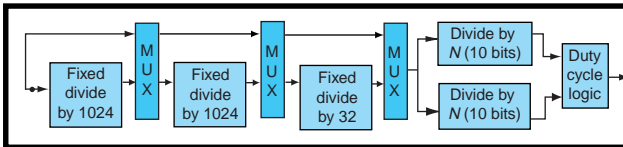


Figure 5—How slow can you go? With five dividers, the ZSBI050 can hold its breath for a long time.

for a reference capacitor as well. I decided to believe the less-is-better version and found the '050 seemed to work fine with just a resistor.

Don't forget to connect SDATA to 5 V to enable the OUTPUT, but you can leave SCLK, SLEEP, and VPGM hanging in the breeze. Powerup and, as mentioned earlier, the chip should come up in the default mode of operation as a 50/50 duty cycle oscillator, running at one-quarter the frequency selected with the REF resistor.

I scrounged around and came up with an 82-Ω resistor, which, according to the datasheet, corresponds with about a 200-kHz internal frequency. Sure enough, I saw a nice clean 50-kHz or so square wave on OUTPUT.

Having established that the chip was alive, I proceeded to wire the SCLK, SDATA, and SLEEP lines to a BASIC single-board computer and threw in an LED for kicks (see Photo 1). Just to confirm that SDATA gates the OUTPUT, and SLEEP shuts down the '050.

To configure the '050, I wrote a short program that simply reads 1s and 0s from Data statements and bit-bangs them over to the chip (see Listing 1). The '050 accepts SDATA on the rising edge of SCLK but don't overlook the 10-μs setup and hold time spec. Also, be careful that SDATA only changes state when SCLK is low, lest you inadvertently put the chip into OTP programming mode (invoked by rising edge of SDATA when SCLK is high).

Listing 1—A simple program is all it takes to configure the ZSBI050. Note the Wait statements that guarantee the required 10-μs setup and hold time.

```
PROGRAM ZSBI050
INTEGER i,j
CONST zclk = ~WAIT 1:OUT $8000,~
CONST zdata = ~WAIT 1:OUT $8001,~
CONST zslp = ~OUT $8002,~
BEGIN
zclk 0
zdata 0
zslp 0
OUT $8003,$80 /* init PIO */

DATA 0,0,0,0,0 /* reserved */
DATA 0,0 /* mode select */
/* 0,0 = 50/50 duty cycle oscillator
   0,1 = variable duty cycle oscillator
   1,0 = retriggerable one-shot
   1,1 = non-retriggerable one-shot */
DATA 0,0 /* reserved */
DATA 0,0 /* trigger select */
/* 0,0 = rising edge
   0,1 = falling edge
   1,x = both edges */
DATA 0 /* divide by 32 */
DATA 0 /* divide by 1024 */
DATA 0 /* divide by 1024 */
DATA 0,0,0,0,0,0,0,1,1,0 /* primary divider */
DATA 0,0,0,0,0,0,0,0,0,0 /* duty cycle */
DATA 0 /* memory protect */

FOR i=1 TO 35
  READ j
  zdata j
  zclk 1
  zclk 0
NEXT i

zdata 1

END
```

I was ready to pull the trigger when I realized I wasn't quite sure who's on first (i.e., the datasheet didn't make the "endian-ness" of the chip real clear). Should I shift the 1st or 35th bit first? Same for multibit fields like the divide ratio—do you send the most or least significant bit first?

"If in doubt, try it out," I say, and it was easy enough to deduce the internal layout by judiciously shifting a single 1 bit. It wasn't long before I was successfully in command of the '050 and able to try out the various features and modes (see Photo 2).

Having explored the '050 operation in RAM mode, I made an admittedly hacker-type attempt to grapple with the programming algorithm and blow the OTP. But, it didn't seem to work. Every time I powered up the part I'd just programmed, it would be the same as one fresh out of the tube.

At the last second, a bit of pondering led me to realize the simple error of my ways. I'd been trying to program a single bit (the divide-by-32 bit). But the question is, how does the

'050 know whether to come up in RAM or OTP mode?

It turns out (confirmed by a call to Zilog), that the last bit (Memory Protect) is the key. Its function is described as "preventing further programming" but probably should also include the fact that the OTP is ignored until this bit is set.

Hack the program to blow that 35th bit, and everything works as expected. The divide-by-32 bit had been successfully programmed all along, but the chip was ignoring it.

TIME'S UP

In these days of zillion-transistor chips, you'd think there'd be little need for simple chips like the '555 and its successors, but sometimes a simple tool is best for a simple job. ☑

Tom Cantrell has been working on chip, board, and systems design and marketing in Silicon Valley for over a decade. You can reach him by e-mail at tom.cantrell@circuitcellar.com or by telephone at (510) 657-0264.

REFERENCE

[1] D. Lancaster, *TTL Cookbook*, Howard W. Sams & Co., Indianapolis, IN, 1974.

SOURCES

LM555

National Semiconductor
(408) 721-5000
Fax: (408) 739-9803
www.national.com

74HC/T5555

Philips Semiconductors
(408) 991-5207
Fax: (408) 991-3773
www.semiconductor.philips.com

S-8081B

Seiko Instruments ICs
(408) 433-3208
Fax: (408) 433-3214
www.seiko-usa-ecd.com

ZSBI05

Zilog Inc.
(408) 370-8000
Fax: (408) 370-8056
www.zilog.com

CIRCUIT CELLAR Test Your EQ

Problem 1—Screw terminal blocks usually come in modular form with two or three terminals per block. The modules can be connected side by side via a tongue and groove on each piece. Often they can be purchased in metric (5.0 mm) or English (5.08 mm) spacings. With such a small difference spacing between pins, either can be used in situations where a low-count terminal module is needed. Why is it important to keep these separated in the parts crib?

Problem 2—The ceramic capacitor size 1206 is the same physical size as the tantalum capacitor size "Y", Johnny is laying out a PCB that uses both. Because both types of capacitors are the same size, can Johnny use the same pad dimensions for both types?

Problem 3—Johnny was perusing the local drugstore's display of batteries. Hanging there on the display were AA, AAA, C, and D size batteries. This got Johnny thinking about the obviously missing A and B size batteries. So, was the pharmacy temporarily out of stock on those items, were these bad designations, or were they discontinued?

Problem 4—On what type of device will you find the notation "REN=x.x", and what does it mean?

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

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

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

ADVERTISER'S INDEX

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

Page		Page		Page		Page	
93	Abacom Technologies	89	Digital Products Co.	88	Matrix Orbital Corp.	68	Scott Edwards Electronics, Inc.
91	Ability Systems Corp.	4,5	DreamTech Computers	94	MCC (Micro Computer Control)	91	Senix Corp.
94	ActiveWire, Inc.	18	ECD (Electronic Controls Design)	79	megatel	84	Signum Systems
16	ADAC	67	Earth Computer Technol	84	MetaLink Corp.	92	Sirius microSystems
73,80	Advanced Transdata Corp.	85	EE Tools (Electronic Engineering Tools)	93	microEngineering Labs, Inc.	86	SMTH Circuits
75	AllElectronics	90	ELNEC	66	Microchip	91	Softaid
92	Allen Systems	56	EMAC Inc.	74,82	Micromint, Inc.	41	Solutions Cubed
92	Amazon Electronics	92	Embedded Micro Software	69	Midwest Micro-Tek	84	Square 1 Electronics
94	Andromeda Research	31	emWare	87	MJS Consulting, Inc.	90	Technicraft, Inc.
90	Antonius Digital	18	Engineering Express	15	Monterey Tools Company	25,56,72	Technologic Systems
90	AP Circuits	85	FDI (Future Designs, Inc.)	90	Mosaic Industries, Inc.	86	Technological Arts
27	Ascendco Scientific, Inc.	7	Future Electronics Inc.	33	NetBurner	34,C3	TechTools
92	Avocet Systems	91	General Device Instruments	84	Nohau Corp.	87	Tern, Inc.
63	Axiom Manufacturing	50	General Software	92	Novasoft	93	Triangle Research
94	Bagotronix, Inc.	85	Hagstrom Electronics	44	On Time	68	Trilogy Design
90	Bay Area Circuits	92	Huntsville Microsystems, Inc.	94	PCB Express, Inc.	87	Two Technologies
91	Becterm Inc.	89	IMAGEcraft	48,C2	Parallax	85	Vesta Technology
86	Beige Bag Software	69	IndustroLogic, Inc.	86	Phoenix International Corp.	86	Vetra Systems Corp.
93	Borge Instruments Ltd.	18	Instant Instrument	84	Phytec	88	Virtual Tools, Inc.
57	CAD-UL	89,90	Intec Automation, Inc.	94	Picofab, Inc.	89	WCSC
25,91	CCS (Custom Computer Services)	17	Interactive Image Technologies, Ltd.	85	Pontech, Inc.	45	Weeder Technologies
91	Ceibo	88	International Electronics Corp.	92	Prairie Digital, Inc.	1	Wilke Technology GmbH
91	ChipTools, Inc.	85	Intronics, Inc.	88	Pulsar, Inc.	90	Wirz Electronics
94	Conitec	49, 89	JK microsystems	55	Rabbit Semiconductor	89	Z-World
45,93	Connecticut microComputer, Inc.	69	JR Kerr Automation & Engineering	92	R.E. Smith	94	Zanthic Technologies Inc.
81	Copeland Electronics, Inc.	86	J-Works, Inc.	72	Remote Processing	23	ZiLOG
85	Creative Control Concepts	87	Lemos International	87	RLC Enterprises, Inc.		
86	Crystalfontz America, Inc.	88	Laipac Technology, Inc.	93	RMV Electronics, Inc.		
C4	Dataman Programmers, Inc.	9	Link Instruments	10,78	Saelig Co.		
94	Decade Engineering	88	Lynxmotion, Inc.	90	Scidyne		

A Reliable Network For Embedded Systems— Getting On Track

Weather Data—Getting More than One Wire's Worth

Computer Control Of A Motorized Telescope

Build A Risc System In An FPGA— Part 3: System-on-a-Chip Design

Buying Power—Low-Cost Power Supply for Embedded Applications

The Shocking Truth About EMC—Part 2: Practical Applications

Embedded Living: HCS=Heating Control System?

MicroSeries: Op-Amp Specifications—Part 2: Getting Some Input

From the Bench: Digital and Analog Output Control—Taking Route X-10

Silicon Update: On the Road Again

EPC Real-Time PC: Real-Time Executive for Microprocessor Systems—Part 1: Introduction to RTEMS

EPC Applied PCs: Under the Covers—Part 2: Applications via NT Embedded 4.0

EMBEDDED APPLICATIONS

PREVIEW
118

PRIORITY INTERRUPT

All Things Considered



I'm getting a little long in the tooth now, but I can remember being a kid and my relatives asking, "What do you want to do when you grow up, Steve?" I'm not sure whether they were genuinely interested or whether they had some skepticism that I'd actually make it to adulthood. Of course, back then we didn't have as many of today's behavioral restrictions or statutory impediments. In truth, if most of us look back at the things we did in our "technically inquisitive" growth stages and applied today's conformity-conscious climate, many of us would be serving five to ten.

I always wanted to be an engineer. In high school, when I hardly knew the difference between professions, I usually answered the what-do-you-want-to-be question with, "a chemical engineer." I was moderately enthused about chemistry, and chemicals were certainly easier to get and cheaper for "experiments" than the conventional sources for ICs. Destiny worked against that career pursuit, however. The first message in the mist (literally) was that *little* rocket fuel experiment I did one afternoon in the high school chemistry lab. To this day I swear someone switched potassium chlorate for the potassium nitrate I was fusing together with sugar to make the propellant. It wouldn't have been so bad if the firemen had just opened all the doors rather than breaking the windows....

The second message came in the form of peer obligation, and yes, Danny Boy, there were people around who were crazier than me. I had a mad-scientist reputation (I'll bet you never would have guessed that) and I was always building electronic gadgets. Always eager to make a buck on my expertise, I took an order from a fellow student to make an electronic timer. It seemed simple enough, all he wanted was a circuit that could be set from 0 to 1 hour with one-second precision and would fire a flashbulb at the end of a wire when it timed out.

Of course, he didn't want to pay until he tested it. I suppose I should have been a little more concerned about the location, but a deal is a deal. Out in the middle of the woods he stopped and pointed to a rock outcropping and said, "Put the flashbulb there and string a wire back a hundred feet or so and put the timer next to it." I dutifully did as instructed and then came back. He pulled out a 5-lb bag of white crystalline powder and a pint of oily looking liquid and mixed them together. The consistency was like moist sand. Finally, he pulled out a little plastic pill bottle with about half an inch of white powder in the bottom. He stuck the AG1 flashbulb in the powder, stuffed some putty in on top of it, and then pushed the pill bottle into the bag of wet powder. "OK, let's go!" he said as he jumped up and started running toward the timer end. I didn't have to be a real engineer to realize the purpose of my timer!

"What was that last stuff in the pill bottle?"

"Lead Azide," he said matter of factly. The term "chemical experiments" instantly took on a whole new meaning. That was the stuff they put in blasting caps.

Before I could protest, he had the wires attached to the timer and had set it for five minutes. In retrospect, I realize that he had never asked if the timer worked. He just attached the wires and yelled "Run!" We stopped about 350 feet from the timer and hid behind some rocks. Taking a breath, I nervously added a little levity, "Well, at least we don't have to wait an hour."

"We wouldn't have to anyway. That mixture isn't stable that long!"

"Holy mackerel," I thought, "what have I got myself into?" Before I could finish analyzing how the present situation was undoubtedly accelerating my reputation from merely being "that science guy" to being on the list of "most wanted", there was a giant explosion. The blast was so powerful my ears were ringing. I looked up and noticed that a boulder thrown by the blast had sheared off a 6" diameter tree trunk three feet above my head. There was a 6' deep crater in front of the outcropping.

"What the @#\$% was that?" I yelled. Acrid smoke surrounded me and dirt was still falling from the sky.

"Nitrobenzene with an oxygen accelerant. Not bad, huh?" All I could think was that everyone in Eastern Massachusetts had probably heard the blast. Hanging around this kid could be bad for my health.

So why am I telling you all this now? It's really a reaction to all the people who write to me or come up to me at shows and thank me for inspiring them to be engineers. While I truly appreciate the complements, I don't let it go to my head. In fact, if anything, those kinds of comments make me feel like I should go out of my way to remind everyone that I made lots of mistakes on my way to being an engineer.

I also tend to criticize our overly obtrusive society norms. In truth, I wonder how I would have turned out if I had to deal with today's evaluation of my behavior as a youth. I'd even like to say that I learned not to play with dangerous things after the experience I just related, but that wasn't the case. It took a third traumatic message to set me on the straight and narrow. But, that's a story for another time.

Somehow after all the experiences, both good and bad, I think I turned out OK—certainly good enough to have affected a few people positively. My personal evolution leads me to wonder if we react too severely today. I wonder if the Steve Ciarcias of today have a chance to clean up their act before society puts them in a rubber room. This isn't a trick question and I don't have an answer. I'm just troubled that it might be reality.

steve.ciarcia@circuitcellar.com