#181 August 2005

www.circuitcellar.com

CIRCUIT CELLAR®

THE MAGAZINE FOR COMPUTER APPLICATIONS

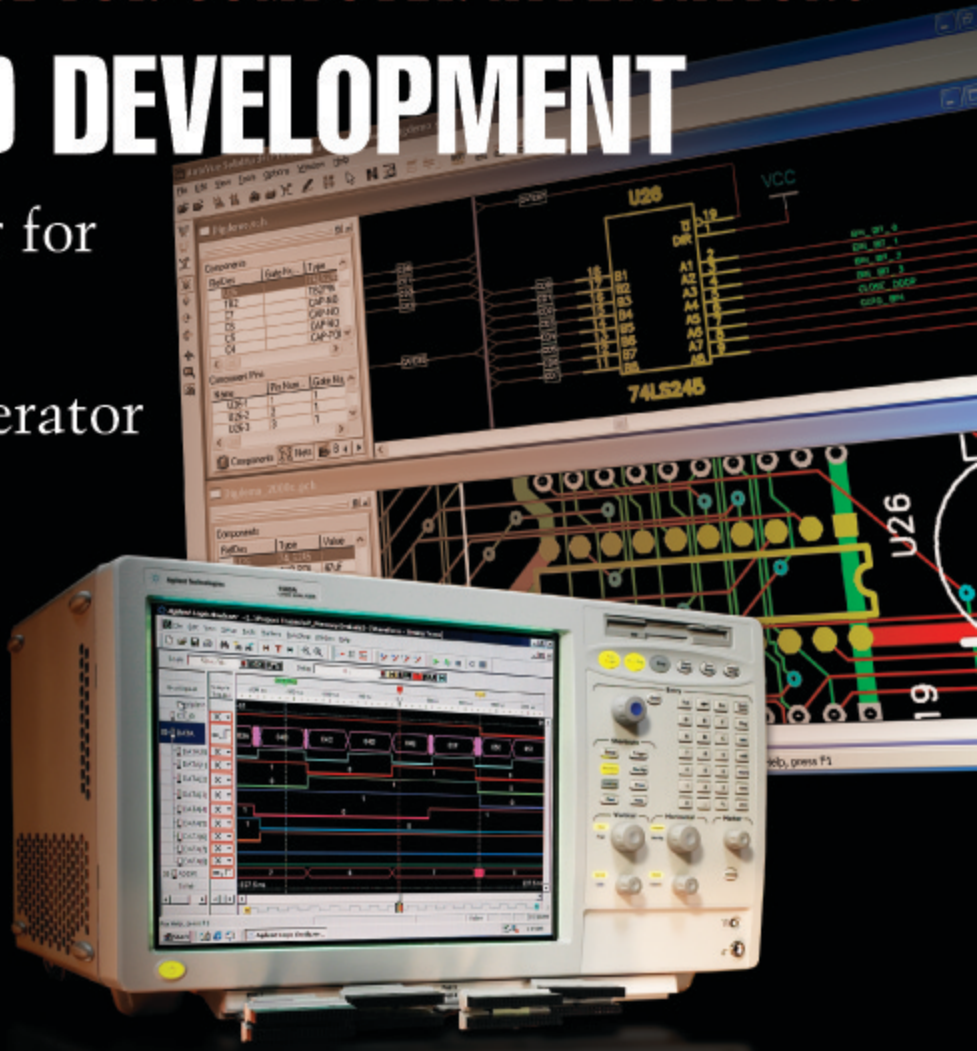
EMBEDDED DEVELOPMENT

Flight Computer for Rocketry

Hardware Accelerator for Robotics

VHDL Tutorial

Revamped Pulse Generators



08>
7 25274 75349 9
$4.95 U.S. ($5.95 Canada)

NetBurner Mod 5270

Do the Math!



95 MHz

+95 MIPs

+32 bits

$59 *Priced per 1000*

NetBurner

Networking in 1 day!

1-800-695-6828 • www.netburner.com

Your *Black Box* approach to visual embedded design.



Go from concept to embedded ASIC in one hour or less with PSoC Express.™

Start with our Programmable-System-on-Chip™ (PSoC™) mixed-signal array: Powerful programmable analog and digital blocks with MCU and flash for less than $2, replacing thousands of fixed-function ICs. Pair it with free PSoC Express software, our revolutionary, code-free visual design tool. The result is hassle-free system-level embedded design with unprecedented time to market. The PSoC Express tool suite provides:

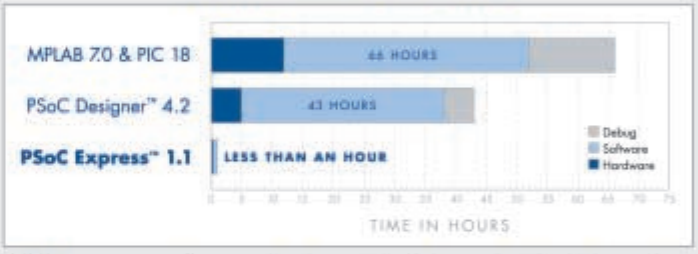
HOT SPEC *Development time reduction of 90% + vs. competitive toolsets*

- *Freedom from C and assembly coding, automatic code generation with no registers to define, no code optimization, and no debug required*
- *Drag-and-drop access to our library of temperature sensors, switches, potentiometers, relays, FETs and fans*
- *Automatic generation of a datasheet, schematic and bill of materials*
- *Portability to our full-featured PSoC Designer™ tool*

I WANT MY BLACK BOX DESIGN KIT

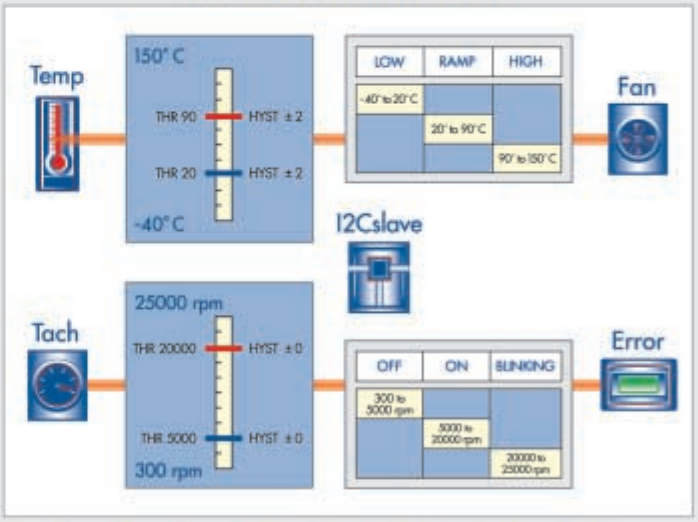
For a PSoC Express Development Kit, free PSoC Express Software, or to attend a free NetSeminar; visit www.cypress.com/ad/psocexpress

DEVELOPMENT TIME COMPARISON



PSoC Express can reduce your development time from weeks to less than an hour.

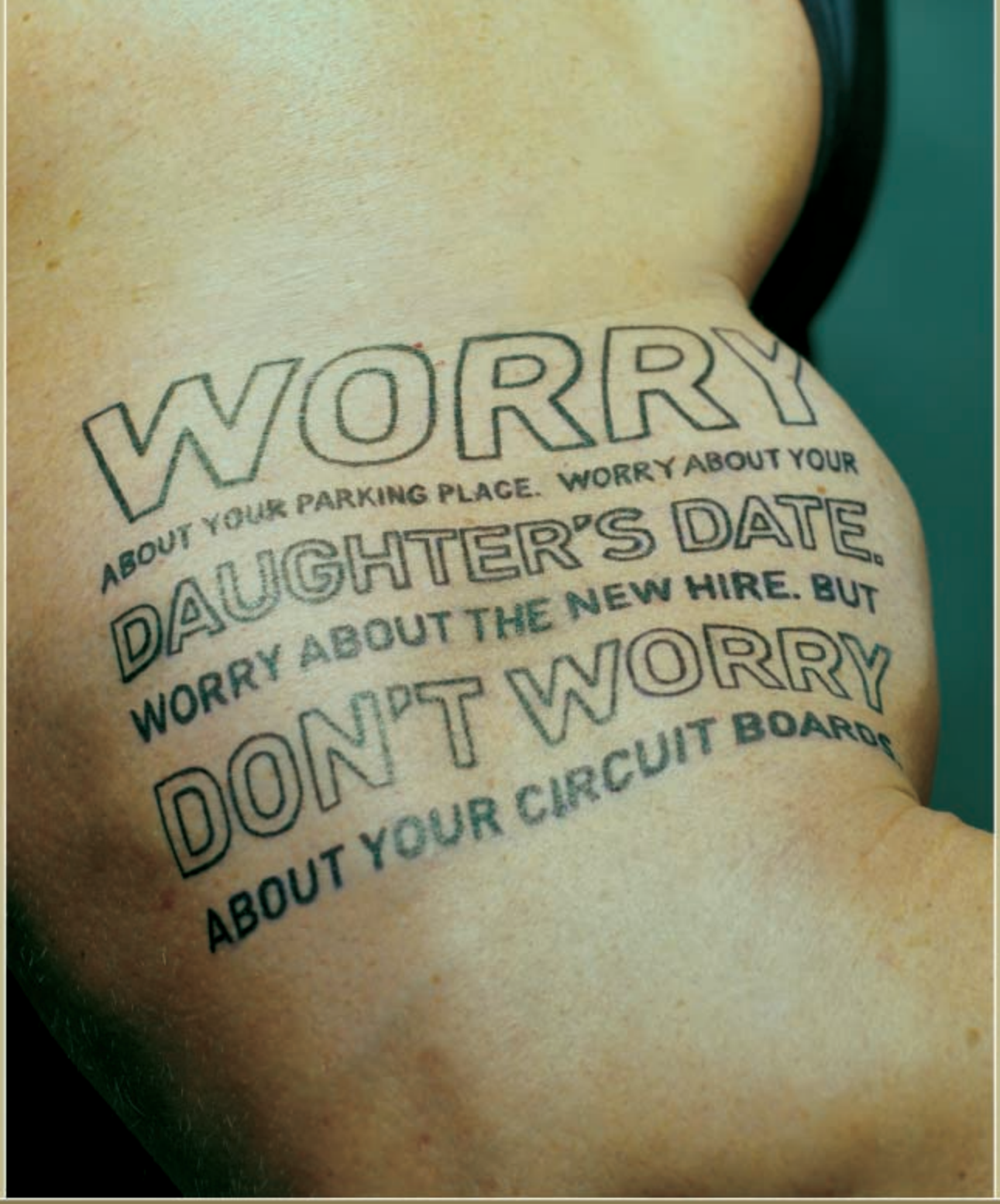
FAST CODE-FREE VISUAL DESIGN



Visually input external sensors, logic functions, output devices, and I2C communications while PSoC Express codes, simulates, and automatically programs your application into the PSoC IC.

CYPRESS PERFORM

PSoC, Programmable System-on-Chip, PSoC Designer and PSoC Express are trademarks of Cypress Semiconductor Corporation. ©Cypress Semiconductor Corporation. All rights reserved.



Visit our website for incredible discounts and monthly promotions.

www.4pcb.com/tattoo or 1-800-979-4PCB.

Sure, there are plenty of things to be anxious about. But with Advanced Circuits, your circuit boards don't have to be one of them. Our FreeDFM service makes sure your files are good to go before you send them in. And once your files get here, our 24 hour tech support means you'll always have someone to talk to—any time, day or night. So go ahead and worry. We know it's what makes a great engineer. You'll just have to worry about something else. Advanced Circuits. One less thing to worry about.

ADVANCED CIRCUITS

Seetron Serial LCDs

...almost too easy!

LCDs have a well-deserved reputation for being a pain in the neck to interface. Their non-standard parallel busses and weird timing are enough to turn your hair gray—if you don't pull it out first. Our LCDs have an easy-to-program serial interface (up to 9600 bps) that makes them a snap to use, particularly with friendly micros like BASIC Stamps®.

• *Easy hookup and mounting.* We don't stop at simplifying display interfacing and programming. For easy, quick connections, we offer a variety of ready-made wiring harnesses. Need mounting hardware? We've got that, too.

• *In stock right now.* Hate to wait? Our products are in stock and ready to ship the day you place your order.

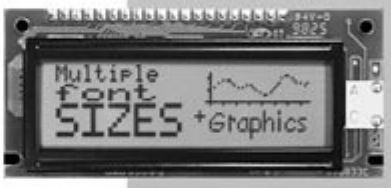
• *Online satisfaction.* Our customers love our web site—it's fast, uncluttered, and jammed with tech data on our products. The online store runs weekly sales with savings up to 40%. Checkout is a breeze with a simple one-page form.



BPI-216N
2x16 serial LCD
ultra-low power
45.00



ILM-216L
2x16 serial LCD
w/ switch inputs,
config EEPROM
49.00



SGX-120L
120x32-pixel
serial graphics LCD
99.00



TRM-425L
4x20 serial LCD
w/ keypad input,
95-screen memory
119.00

www.seetron.com

Many more models available at seetron.com

Scott Edwards Electronics, Inc.
1939 S. Frontage Rd. #F, Sierra Vista, AZ 85635
phone 520-459-4802 • fax 520-459-0623
web www.seetron.com • email sales@seetron.com

TASK MANAGER

Modern Solutions Improve Outdated Projects

Last week I looked at some of the preliminary responses from the 2005 readership survey. One comment in particular jumped out at me. Someone noted he/she would like to see more old projects revamped with up-to-date hardware. For that reader, and anyone else who shares the same interest, this month we have a pair of articles about effective updates to two different pulse generator projects.

First up we have Tom Napier's article, "Real-Time Pulse Generator" (page 18). Tom's original TTL-level pulse generator was designed around a Microchip 20-MHz PIC16C55 microcontroller. It generated pulses from 0.1 µs to 0.99 s long. In the 1990s, the 20-MHz microcontroller was sufficient for Tom's needs. But in 2005, 20 MHz pales in comparison to what's available with new devices. The introduction of 40-MHz PIC16C5*x* chips opened the door to new possibilities for this old project. By redesigning his project around the latest PIC16C55, Tom doubled the clock speed and vastly improved the system's performance. The minimum pulse period improved from 5 to 3 µs. Along the way, Tom also took the opportunity to simplify his original code. Although Tom's original design served its purpose, the end result of the redesign is a more efficient and more useful tool.

Brian Millier also recently took advantage of newly released hardware to improve a pulse generator he designed in the late '90s (page 59). Built around an Am9513 from Advanced Micro Devices, the original project lacked the kind of power Brian was looking for. He found that power almost a decade later by upgrading to a Cypress PSoC. Restructuring the project around a CY8C27443 microcontroller enabled Brian to achieve his goal of generating pulses below 100 ns. As Brian explains, the versatility of the CY8C27443 microcontroller's internal and external clocking options was a critical factor in the success of his redesign.

There's a lesson to be learned here: Don't discard your old projects. Whether you reluctantly designed a mediocre project at a time when better hardware simply wasn't accessible or you realize your formerly up-to-date project is unimpressive by today's standards, you probably have the start of a solid, useful project. New hardware that offers more processing power, faster speed, and enhanced flexibility can help you modernize your original projects. Tom and Brian's refined pulse generators showcase what can be achieved when you combine cutting-edge technology with classic ideas.

jennifer.huber@circuitcellar.com



CIRCUIT CELLAR®
THE MAGAZINE FOR COMPUTER APPLICATIONS

EDITORIAL DIRECTOR/FOUNDER
Steve Ciarcia

MANAGING EDITOR
Jennifer Huber

TECHNICAL EDITOR
C.J. Abate

WEST COAST EDITOR
Tom Cantrell

CONTRIBUTING EDITORS
Jeff Bachiochi
Ingo Cyliax
Fred Eady
George Martin
Ed Nisley

NEW PRODUCTS EDITOR
John Gorsky

PROJECT EDITORS
Steve Bedford
Ken Davidson
David Tweed

CHIEF FINANCIAL OFFICER
Jeannette Ciarcia

CUSTOMER SERVICE
Elaine Johnston

CONTROLLER
Jeff Yanco

ART DIRECTOR
KC Prescott

GRAPHIC DESIGNER
Mary Turek

STAFF ENGINEER
John Gorsky

QUIZ COORDINATOR
David Tweed

ADVERTISING

PUBLISHER
Dan Rodrigues E-mail: dan@circuitcellar.com

ASSOCIATE PUBLISHER/DIRECTOR OF SALES
Sean Donnelly Fax: (860) 871-0411
(860) 872-3064 E-mail: sean@circuitcellar.com
Cell phone: (860) 930-4326

ADVERTISING REPRESENTATIVE
Shannon Barraclough Fax: (860) 871-0411
(860) 872-3064 E-mail: shannon@circuitcellar.com

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

ADVERTISING ASSISTANT
Deborah Lavoie Fax: (860) 871-0411
(860) 875-2199 E-mail: debbie.lavoie@circuitcellar.com

Cover photograph Chris Rakoczy—Rakoczy Photography
www.rakoczyphoto.com

PRINTED IN THE UNITED STATES

CONTACTING CIRCUIT CELLAR

SUBSCRIPTIONS:
INFORMATION: www.circuitcellar.com or subscribe@circuitcellar.com
To Subscribe: (800) 269-6301, www.circuitcellar.com/subscribe.htm, or subscribe@circuitcellar.com
PROBLEMS: subscribe@circuitcellar.com

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
NEW PRODUCTS: New Products, Circuit Cellar, 4 Park St., Vernon, CT 06066
newproducts@circuitcellar.com

AUTHOR CONTACT:
E-MAIL: Author addresses (when available) are 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 $39.95, 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 5650, Hanover, NH 03755-5650 or call (800) 269-6301.

Postmaster: Send address changes to Circuit Cellar, Circulation Dept., P.O. Box 5650, Hanover, NH 03755-5650.

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®.

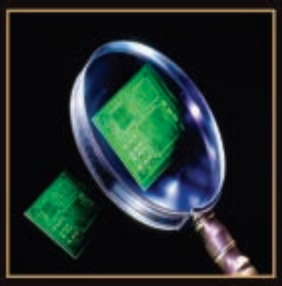
The information provided by Circuit Cellar® is for educational purposes. Circuit Cellar® makes no claims or warrants that readers have a right to build things based upon these ideas under patent or other relevant intellectual property law in their jurisdiction, or that readers have a right to construct 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 © 2004 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.

When you take a **risk** here



You **pay** for it here in field failures






And that's no way to run a **business**

Sierra Proto Express High-Reliability Lead-Free PCBs

Replace "risk" with "reliability" during the assembly process to give you peace of mind when it comes to your business. Our **High Reliability Lead-Free PCBs** are RoHS, WEEE compliant, and made using Pb-free laminate with a Td of 364° C, with fully automated systems & most stringent process controls to guarantee 100% functionality & reworkability.

For orders & information, call **1.800.763.7503 x500**
or visit our website: **www.protoexpress.com**

Because to be the best *you need the best!*

Sierra Proto Express, 1108 West Evelyn Avenue, Sunnyvale, California 94086, U.S.A.

Testing Services courtesy IRTS, manufacturer of the HATS™ Tester. Information about the Highly Accelerated Thermal Shock HATS™ Tester can be found at www.HATS-Tester.com.

August 2005: Embedded Development

FEATURES

10 **Flight Computer for High-Power Rocketry**
Steven Pope

18 **Real-Time Pulse Generator**
Tom Napier

24 **Build a Hardware Accelerator**
Amir Darvishi & Wael Badawy

30 **Strong Encryption for PICs (Part 1)**
Triple-DES Algorithm
Romeo Costanzo

38 **Hardware Synthesis Using VHDL (Part 1)**
VHDL Implementation
Michael Griebling

59 **PSoC Pulse Generator**
Brian Millier


Z8F08-Based Flight Computer (p.


A Platform for a Hardware Accelerator (p. 24)


Build a PSoC Pulse Generator (p. 59)

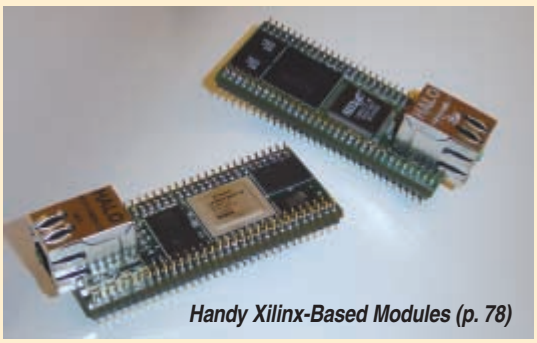
COLUMNS

46 **FROM THE BENCH**
Simulate First
Simulation Techniques for Testing New MCUs
Jeff Bachiochi

68 **APPLIED PCs**
Reach for Interface Control
Add a GUI to a ZF8042A-Based Design
Fred Eady

75 **ABOVE THE GROUND PLANE**
Analog Tips and Tricks
Ed Nisley

78 **SILICON UPDATE**
I Left My Chip in San Francisco
Tom Cantrell


Handy Xilinx-Based Modules (p. 78)

DEPARTMENTS

4 **TASK MANAGER**
Modern Solutions Improve Outdated Projects
Jennifer Huber

8 **NEW PRODUCT NEWS**
edited by *John Gorsky*

93 **TEST YOUR EQ**
edited by *David Tweed*

94 **INDEX OF ADVERTISERS**
September Preview

96 **PRIORITY INTERRUPT**
Risk and Opportunity
Steve Ciarcia



With Atmel's AVR microcontrollers, your designs will always give you more than you expected.

If you're looking for ways to get better designs to market faster (and who isn't these days), start designing with Atmel's AVR® family of MCUs. No other 8-bit micro can give you its exceptional combination of performance, flexibility and power management.

AVR is the most efficient and powerful 8-bit MCU on the planet. It has an enhanced RISC core running single-cycle instructions and a well-defined I/O structure that limits the need for external components. So it's not only easier to use, it delivers blazing execution of up to 20 MIPS.

And when you consider AVR's outstanding Flash and EEPROM capability, plus its huge range of packaging, peripherals, memory sizes, power management modes and development tools, there's almost no limit to what you can design. From complex battery operated applications to industrial controls, and a whole lot more in between.

That's why AVR has become the most popular microcontroller of its kind, with over a billion installed around the world.

So if you're looking for ways to get more out of your designs, make sure you use the microcontroller that delivers more.

Check it out today at www.atmel.com/ad/avr and register to win a free development kit. It just could be the start of a very fruitful relationship.

Find out more about Atmel's AVR solutions today at www.atmel.com/ad/avr

Performance	Up to 20 MIPS
Power consumption	100nA (power down)
Memory size	1 to 256 Kbytes Flash
Packaging	8 to 100 pins
Peripherals	UART, SPI, 2-wire interface, timers, PWM, 10-bit AD, CAN, LCD
Migration path	All devices code compatible for seamless upgrade

ATMEL®

Everywhere You Are™

© Atmel Corporation 2005. All rights reserved. AVR, Atmel, logo and combinations thereof, are registered trademarks, and Everywhere You Are is a trademark of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others.

NEW PRODUCT NEWS

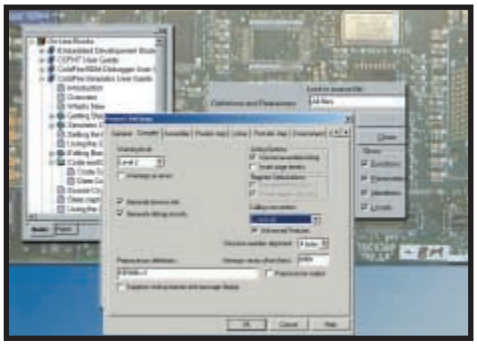
Edited by John Gorsky

UPDATED COMPILER FOR COLDFIRE CHIPS

The third version of **Crossware's C compiler** is for Freescale's ColdFire microprocessors and microcontrollers. The new compiler simplifies development and increases the speed of compiled programs.

The compiler can carry out the flow analysis of a program and then use this analysis to perform optimizations that aren't otherwise possible. New additional calling conventions can be used to further increase your compiled program's speed. In addition, you can pass function parameters in registers and use function inlining to eliminate the function call.

A range of facilities was developed to test the compiler and the libraries that come with it. For instance, a special version of the embedded development studio can automatically generate, compile, and execute complex expressions. The code produced is self-checking. Execution halts if a result is incorrect. Similarly, self-checking code can be automatically generated to test library functions. Randomly generated input data ensures that the library functions operate correctly with a wide range of inputs.



The new compiler, which is available as part of the ColdFire Development Suite, includes a C compiler assembler, linker, simulator, and debugger all integrated into an easy-to-use development environment. A USB BDM interface is also available to complete the set of development tools.

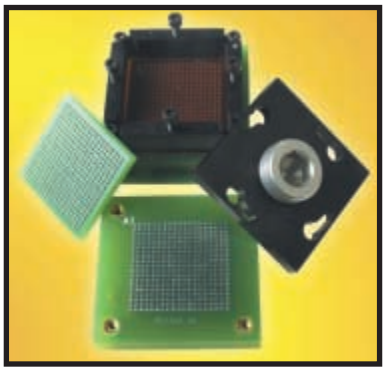
The suite costs approximately **$2,100**.

Crossware
www.crossware.com

HIGH-PERFORMANCE SOCKET

The new high-performance **SG-BGA-6156** socket allows 0.8-mm pitch, 17 mm × 17 mm ICs to be used in a socket and operate without compromising performance in high-speed computing and communication applications. The sockets easily support dense BGA devices using a high-performance conductive elastomer contactor. The contact resistance is normally 10 mΩ.

The sockets have a precision design that guides the IC to the exact position for the connection of each ball and uses an aluminum heatsink screw to provide compressive force. The socket has a thermal pad to conduct heat from the silicon to the heatsink screw. The socket accommodates IC packages such as the Motorola 1473-01 package, 0.8-mm pitch BGA, and 19 × 19 array of balls. The socket requires no mounting holes and it needs only 3.75 mm of perimeter space for the SMT adaptor to be soldered on the target PCB. The socket also can be mechanically mounted on the SMT adaptor.



The SMT adaptor (SF-BGA332A-B-05) consists of a standard high-temperature FR4 with plugged via technology, eutectic SN63PB37 solder balls, and threaded inserts for mounting the socket. These patented ZIF sockets are simply mounted on the SMT adaptor, which is soldered to the target PCB.

The socket SG-BGA-6156 costs **$417**. The SF-BGA332A-B-05 costs **$112**.

Ironwood Electronics
www.ironwoodelectronics.com

DC BRUSH SERVO DRIVER

The **EZSV10 DC Brush Servo Driver** is the smallest fully programmable DC servo driver and controller on the market. Measuring just 0.95″ × 1.4″ × 0.6″, the EZSV10 is smaller than a standard quadrature encoder and requires little or no tuning when used with most motors measuring less than 1.5″ in diameter. A single four-wire bus (two power and two communications) can daisy chain up to 16 DC motors simultaneously.

The fully intelligent driver/controller accepts high-level commands from a serial port to control motors at 1.5 A continuously from 12 to 40 V. Programming the EZSV10 is intuitive. A first-time user can make a servo motor move intelligently in less than 30 min.



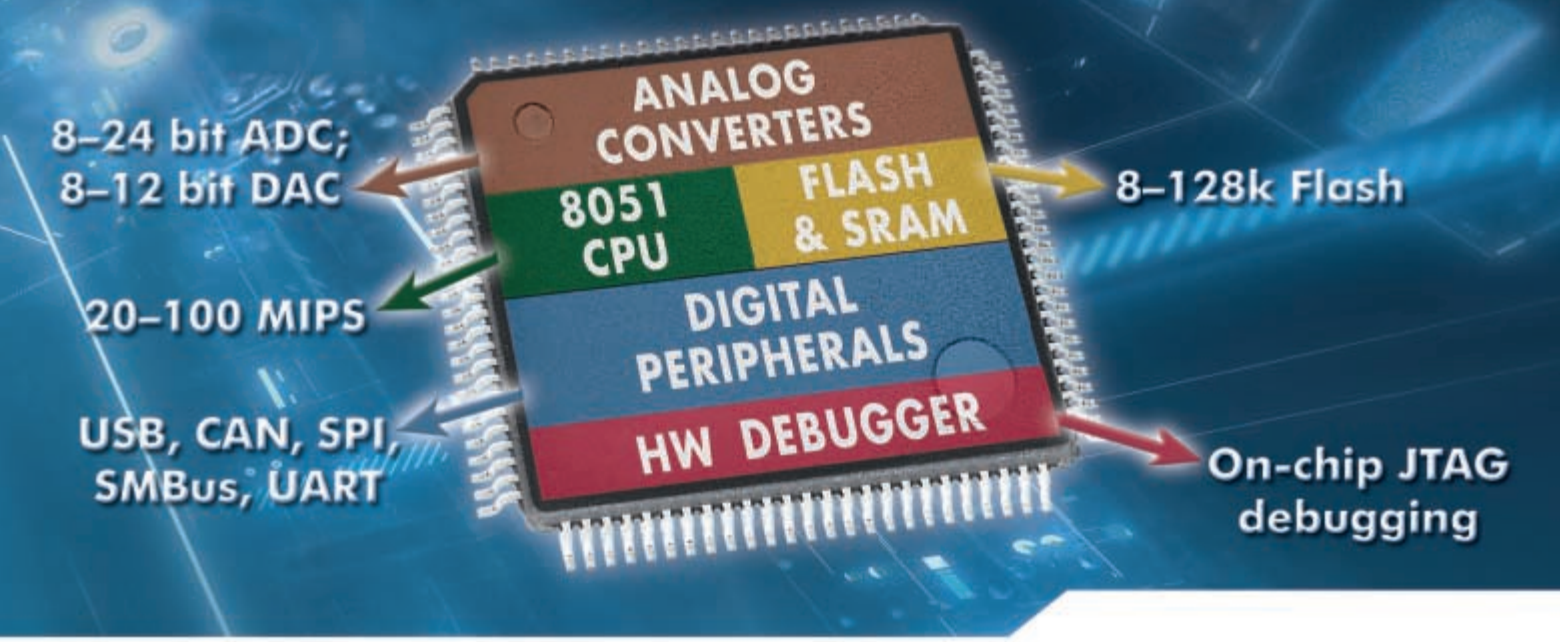
The EZSV10 is capable of stand-alone operation. By using an on-board EEPROM for user program storage, it can be set to execute a string of commands at power-up. Commands also can be issued from any serial terminal program or from the EZServo/Stepper Windows application.

The driver communicates by way of ASCII commands to set operating parameters such as velocity, acceleration, and position. The commands are intuitive and simple. The industry-standard communication protocol is compatible with devices that use the Cavro DT or OEM protocol.

The EZSV10 costs **$199**.

AllMotion Inc.
www.allmotion.com

High-performance analog. Supercharged 8051. Small. Fast. Flexible.



Complete Family of Mixed-Signal MCUs

Silicon Laboratories' MCUs combine high-performance analog with the fastest 8-bit CPU (up to 100 MIPS) packaged as small as 3 x 3 mm. Programmable Flash and on-chip JTAG-based debug in each device provide complete design flexibility and superior system performance.

Precision Mixed-Signal	Precision performance for high resolution analog systems
Small Form Factor	Low power and small footprint with superior performance
General Purpose	Ideal for a broad range of embedded control applications
USB	Integrated USB 2.0 function controller and on-chip transceiver
CAN	Connectivity with integrated CAN 2.0 B controller
Interface Products	Single-chip USB-to-UART bridge for RS-232 to USB

For more information, visit www.silabs.com/MCU



Comprehensive Development Kits

- Software CD
- Serial Adapter
- Target Board
- All Cables
- Power Supply

WIRELESS | WIRELINE | OPTICAL | **MICROCONTROLLERS**

SILICON LABORATORIES

©2004 Silicon Laboratories Inc. Silicon Laboratories and the Silicon Laboratories logo are trademarks of Silicon Laboratories.

www.silabs.com

Flight Computer for High-Power Rocketry

Steven's ZSAT II flight computer can compile data like altitude and acceleration, and then send it to his cell phone. The eZ8F642-based system includes a Trimble GPS module, a long-range RF module, and a controller for a wireless video camera.

If you're interested in rocketry, you've probably seen the movie *October Sky* (1999). Or perhaps you've read Homer Hickam's book *Rocket Boys* (1998), the memoir on which the movie is based. It's a story about both a boy's dream to discover our heavens and the days of early amateur rocketry. Can it get any better than that?

Today, amateurs are taking rocketry to the next level. Groups like the National Association of Rocketry (NAR), Tripoli, the Civilian Space eXploration Team (CSXT), and Scaled Composites are working to expand our knowledge of amateur rocketry. Some of the people involved have claimed the big X PRIZE; others have been after the challenge of speed, altitude, and a safe recovery.

High-tech components are now pushing into the realm of amateur rocketry, which has become more sport than hobby. Rockets include RF/GPS tracking systems, on-board flight cameras (video and still), and flight computers for measuring speed, temperature, and altitude. Dual-deployment computers control the release of parachute systems, which some insiders call "laundry." Timing systems control the staging of a multistage rocket (or "air starts"). Universities are using high-power rocket payload bays to test small satellite components. Refer to the CubeSAT program for more information (http://cubesat.calpoly.edu).

After building a high-power rocket of my own, I wanted to add some high-tech gadgetry to the system. I really wanted a flight computer that could measure the rocket's acceleration, altitude, and temperature, and then record the data on a serial flash memory device. Rather than spend the money on an off-the-shelf flight computer, I designed my own.

In this article, I'll describe how I built my Zilog Z8-based ZSAT flight computer. With this system, I can retrieve useful flight data on my cell phone. No laptop computer required. I'll also explain how you can get involved in high-power rocketry.

LEVEL 1 CERTIFICATION

Most of us involved in high-power rocketry started out working with model rocket kits and motors. But high-power rocketry is different than model rocketry. According to the NAR, a high-power rocket must have either a motor with more than 160 newton-seconds (Ns) of total impulse (an H motor or larger) or multiple motors working together to exceed 320 Ns. The motor must also have an average thrust in excess of 80 N. Lastly, the rocket must weigh more than 1,500 g or include airframe parts of ductile metal.

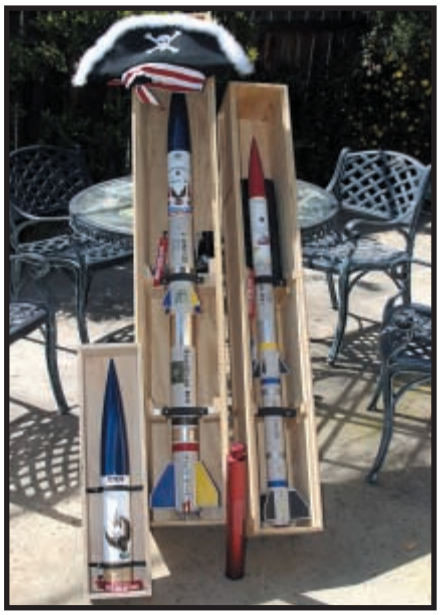


Photo 1—*My Level 1 Mini-Me rocket is on the right. My Level 2 Bootstrap Bill rocket is on the left. To the left of the 54-mm Dr. Rocket motor casing is a second ZSAT II payload bay with a camera system.*

You must be certified to launch high-power rockets. Level 1 certification is the first step. You should first check out the high-power rocket safety code (www.nar.org/NARhpsc.html). Then, join a local rocket club and attend a launch. The process of Level 1 certification is started by purchasing a Level 1 class motor (H and I). You should then buddy up with someone who can help you prepare your rocket for its first flight.

On flight day, you'll obtain your motor reload. Load your motor casing, pack the chute, fill out your flight card, and head out to the range safety officer (RSO) table. One tip: Don't install the igniter in the motor reload until you have cleared the RSO table and your rocket is on a launch pad. For any certification flight, you'll need someone with Tripoli or NAR "prefect" status to monitor your RSO check, launch pad preparation process, and launch. This person will later sign off on your certification paperwork.

Okay, so how did I get certified? First, I made the mistake of buying a Public Missiles, Ltd. (PML) AMRAAM 4 rocket kit with a dual-deployment system, which turned out to be too big and complicated for my first project. Next, I joined the Association of Experimental Rocketry of the Pacific (AERO-PAC) and attended my first launch in 2003 in the Black Rock Desert in Nevada. The event brought



Photo 2—*The Z8F08-based ZSAT I flight computer is the perfect size for a Level 1 rocket.*

me up to speed on all the terms, hardware, and procedures. I then joined Tripoli, NAR, AERO-PAC, and paid for a Low Explosive Use (LEU) permit. The total cost for joining the groups and a three-year LEU permit was approximately $320. During the six-month period that followed, I pored over all the written material I could get my hands on.

I finally ended up purchasing a PML AMRAAM 2 Level 1 rocket kit and a 29-mm Dr. Rocket motor casing (see Photo 1). I also bought RocFlight software for flight simulation. The software simplifies the entire process. By inputting my rocket's size, weight, drag, motor size, and type, I got its speed, altitude, and Gs.

MINI-ME & ZSAT I

My high-power rocket club holds three launches a year at Black Rock: the MudRock launch in June, the Aeronaut launch at the end of July, and the AERO-PAC Extreme Performance Rocket Ships (XPRS) launch at the end of September. I completed my Level 1 certification flight at the July 2004 Aeronaut launch during a mission called "Mini-Me," which is also the name of my AMRAAM 2 rocket.

After building the Mini-Me Level 1 rocket, I decided to add a payload area for a flight computer that measures acceleration, altitude, and temperature. My first rocket flight computer, code named ZSAT I, is shown in Photo 2. I can view the flight data on my cell phone right after a flight. Yes, it's a little geeky!

I built the ZSAT I system around a 28-pin, 3-V, 8-KB, flash memory-based Zilog Encore! F08 microcontroller, which includes IrDA, 1 KB byte of SRAM, a 10-bit ADC, I^2C, and an SPI port.

My Level 1 certification flight with Mini-Me was on a H220W reload. The cost of the motor reload was approximately $35. I didn't do anything special for this flight. I keep things simple. I purchased my reload on launch day from a vendor at the launch site. I then read the reload instructions, installed the loaded 29/180 motor casing in Mini-Me, taped my igniter to the side of the rocket, filled out the flight card, and headed down to the RSO table.

After I hooked up with my Tripoli prefect and cleared the RSO table, I loaded Mini-Me on Pad 13, installed the igniter, hooked the igniter clips to the igniter, and pushed the launch pad test button. All systems were go. I then waited for my launch slot. Man, I had some butterflies.

Five, four, three, two, one…Fire in the hole! Off she went, well, like a rocket.

After my rocket was safely floating in its chute, the butterflies in my stomach disappeared and I set off to retrieve it. My Level 1 certification was almost completed at that point. I just had to pick up the rocket and bring it to my prefect so he could look it over and sign my certification papers.

ZSAT I TEST

I used the Mini-Me mission for more than my certification. I also used it as a test flight for my ZSAT I payload bay. Still buzzing from the flight, I ran down to vendor row and bought another motor reload. For the second flight, I put my ZSAT I flight computer in Mini-Me's payload bay. This time I had an H248R. (I just love all this code stuff. The *R* stands for Redline). I saw a red flame as that baby burned. Cool.

I used the Z8F08200100 development kit for the ZSAT I. One of the microcontroller's coolest features is its IrDA function. I cut out a small window in the payload bay so I can whip out my Palm OS cell phone on the launch pad and arm ZSAT I for flight.

On the launch pad, I loaded the igniter, hooked up the igniter clips, and then pulled the "remove before flight" pin. (I love this added touch.) On ZSAT I, I connected the 9-V battery's positive terminal to the control electronics and took the negative terminal to the other side of a power jack mounted on the side of the rocket. The other side returns ground back to ZSAT I. When the flight pin is installed, the ground point is cut and the 9-V battery is disconnected from the electronics. Pulling the flight pin connects the battery to ZSAT I.

Now came the cool part. I whipped out my Palm OS cell phone, launched my Palm application called Launchpad, and beamed a control message via the IrDA port that told the ZSAT I to prepare for flight. ZSAT I returned with an acknowledgment and flashed the launch LED. The control firmware then started looking at the accelerometer and waiting for a change in its output, thus indicating Launch Detect. When Launch Detect is triggered, the control firmware samples the three sensors and saves the data in an SPI EEPROM device until it's full.

I armed ZSAT I on the launch pad with my Palm OS cell phone, waited for my launch slot, and then let her go (see Photo 3). After takeoff, I tracked Mini-Me on foot. When I reached it, I saw a flashing LED sequence indicting that the EEPROM storage was full. Again, I whipped out my cell phone and asked the rocket for flight status over the IrDA port. The system quickly returned a snapshot of the rocket's altitude, speed, and temperature: 3,400′, 620′ per second, and an average payload temperature of 83°F, respectively. The firmware can display only one value on the screen, but I plan to upgrade it in the near future so it will display a flight graph.

The launch at Black Rock was awesome. Because everything went as planned, I immediately started thinking about a Level 2 mission and a bigger, better IrDA flight computer: ZSAT II.



Photo 3—*Check out Mini-Me taking off for the first time with ZSAT I in its payload bay.*

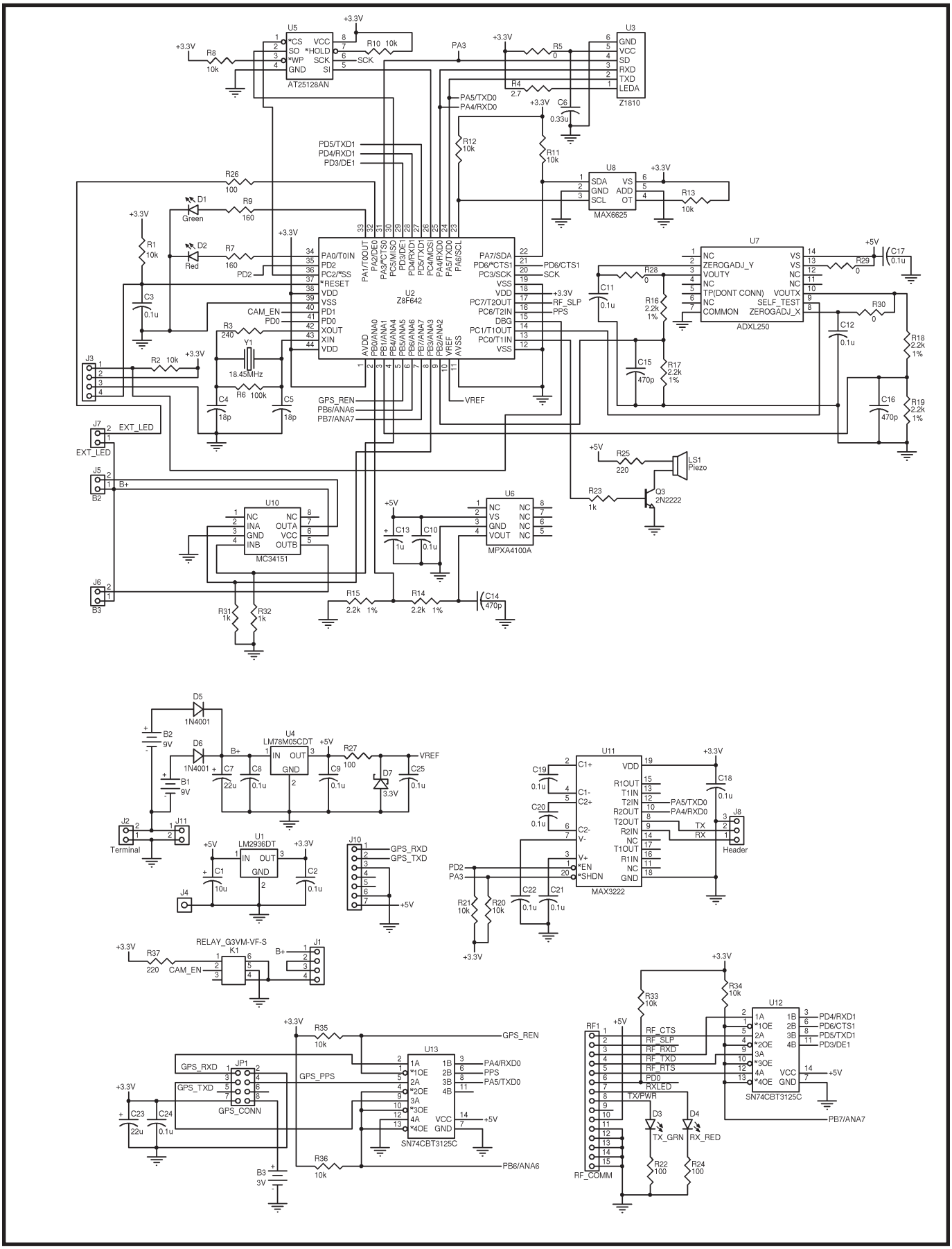


Figure 1—*A Z8F642 microcontroller monitors and controls the ZSAT II flight computer. This flash memory-based microcontroller has 64 KB bytes of program memory. New features, changes, and improvements are easy to download via the debug connector J3.*

LEVEL 2 CERTIFICATION

The Level 2 certification process begins with a 50-question written exam. I took my Level 2 test a few weeks before the XPRS launch in September, so I was sure I could fly. Passing the written test enabled me to purchase motor reloads in the J, K, and L classes. I selected a J415W for Bootstrap Bill, my PML ARRAM3 rocket with a 54-mm motor mount (see Photo 1).

My plan for Bootstrap Bill was to put it up early the next day. I kept the first flight simple. It was a shakedown flight without a payload. I purchased my J415W reload for approximately $80 and prepped Bootstrap Bill first thing in the morning.

We flew a flag at our camp for Bootstrap Bill's maiden flight. With cameras rolling, we carted it over to the RSO table, answered a few questions, and then set off to launch pad 22. After Bootstrap Bill was loaded on the launch rod and the igniter was installed, I gave a big thumbs-up to the launch control officer (so many cool titles). Once again, I got the butterflies as I waited for my rocket to take to the sky.

Five, four, three, two, one, launch.

Seeing Bootstrap Bill clear the tower, the sensation of butterflies in my stomach turned into a feeling of joy. The up part, as they say, was not 100%. About halfway into the flight, Bootstrap Bill fishtailed a little, but the event put Bootstrap Bill back in control. I recovered Bootstrap Bill about two miles from the launch pad. I returned to my Tripoli prefect to get my rocket examined and my paperwork signed.

ZSAT II

ZSAT II is a lot like ZSAT I, but it has a few more features (see Figure 1). I used a bigger microcontroller, the Z8F642, which has 64 KB of on-chip flash memory and 4 KB of on-chip RAM. It also includes a Trimble GPS module, a MaxStream long-range RF module, and a control for a 2.4-GHz wireless video camera (see Photo 4).

ZSAT II can measure acceleration, temperature, and altitude. An Analog Devices ADXL250 ±50-g dual-axis accelerometer measures acceleration. I selected a dual-axis accelerometer to measure both *x* and *y* forces. The accelerometer's outputs, each of which goes into an ADC channel, are sampled to detect a launch and continue to save samples in the external SPI flash memory during a flight. The most important thing here is to mount the accelerometer in the right direction.

After you turn on and enable the flight computer using the Palm IrDA software, the main control loop reads the *x* accelerometer input and waits for a force in the upward direction to start the flight data logging. A backwards accelerometer means no flight data!

Altitude is measured with a a Motorola MPX4100 pressure sensor. The sensor's output is connected to an ADC input on the Z8F642 MCU. When the rocket is armed for flight using the IrDA port, a reading is sampled to determine the launch site's pressure, which is then used to calculate the altitude. During the flight, samples are continually taken and saved in the external SPI flash memory.

A Trimble GPS Module is available

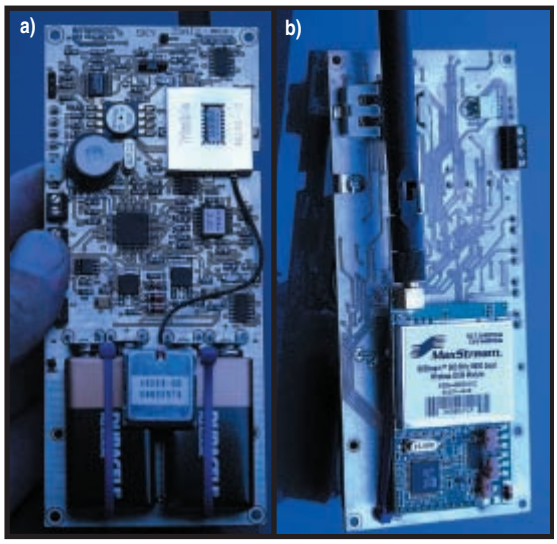


Photo 4a—*The ZSAT II flight computer includes a Trimble GPS module and GPS antenna mounted on top of 9-V batteries. The IrDA transceiver is located just below the Sky label at the top of the board. This location lines up with a window that I cut in the payload bay.* **b—***Here you can see the MaxStream 900-MHz RF module, the battery holder for GPS backup battery, and a terminal block for connecting the wireless camera.*



on the Z8F642's serial port 0, which is shared with the IrDA transceiver. A routine yet to be developed will sample the GPS inputs and switch back to look for pending IrDA data. During flight, serial port 0 is turned over to the GPS module. After the flight (pressure sample = baseline), the control firmware needs to sample the GPS input. This data is transmitted via the MaxStream RF module on serial port 1 to signal the rocket's positions on the playa.

Listing 1 is ZSAT II's main control loop. The go_for_launch flag is set when a command is sent over either the IrDA port or the 900-MHz RF link. This triggers the flight computer to start sampling the accelerometer channel looking for a Launch Detect. A sampled value over 0x5A hex tells the system that the rocket is in the air. From this point on, all ADC channels are sampled and saved in an SPI EEPROM for me to review after the flight using my IrDA PDA or computer.

Listing 1—*This is ZSAT II's main control loop. Launch Detect is determined when the accelerometer output goes above a hex value of 0x5A.*

```
*********************************************************************
Wait for "Launch" control loop
*********************************************************************
if (go_for_launch)  //Flag set by Palm IrDA port - "Arm for Flight"
    {
*********************************************************************
Get ground reference
*********************************************************************
//Pressure on ADC 0
//ACC X (up) on ADC 1
//ACC Y (Side) on ADC 2
    read_save_allsensor_data(0);    //Get base line start save at
                                    //address 0
//Save four bytes
    do
    {
    ACTL = 0xA1;                    //Start channel 1 conversion ACC X
    while (ACTL & 0x80);            //Wait for conversion to complete
    data_hi = ADHR;                 //Read X data
    }
    while(data_hi <= 0x5A);         //Wait for launch detect
*********************************************************************
We have Launch Detect
*********************************************************************
          PA0 |= 0x02;              //Turn on LED
//Launch
          addr_loc =4;              //Next address to save at
          do
          {
          ACTL = 0xA0;              //Start channel 0 conversion
          while (ACTL & 0x80);
          data_hi = ADHR;
          set_spi_wr_en();
          SPI_Write(data_hi,addr_loc);
          read_spi_status();
          do
          {
          read_spi_status();
          }while (spistatus & 0x01);
          addr_loc++;
          ACTL = 0xA1;              //Start channel 1 conversion
          while (ACTL & 0x80);
          data_hi = ADHR;
          set_spi_wr_en();
          SPI_Write(data_hi,addr_loc);
          read_spi_status();
          do
          {
          read_spi_status();
          }while (spistatus & 0x01);
          addr_loc++;
          }while(addr_loc <= 0x3FFF); //Keep reading pressure until
                                      //memory is full
          PA0 &= 0x0D;              //Turn off LED, memory full
          go_for_launch = 0;        //Tell upper layers flight is over
          }
}
```

ZSAT II TEST

I selected a J468R for ZSAT II's shakedown flight on Bootstrap Bill. I built this reload in the afternoon and loaded it into Bootstrap Bill. I then prepped ZSAT II, which also has provisions to trigger two electronic matches that can be used to release the parachute system or timing control for a multiple-stage rocket.

Two other systems came into play at this point. One was a new RF module that plugs into a Palm OS device for long-range RF control. On the launch pad, I still had my IrDA port, which has become a standard addition to all of my products. But as the rocket cleared the pad, the IrDA port didn't work. I could then switch over to this Palm-based RF handheld system.

I developed a central command station that enables people back at base camp to monitor the rocket (see Photo 5). On this mission, I loaded Bootstrap Bill on the launch pad and radioed to central command to power up the on-board camera for a video test (see Photo 6). A command sent over the RF link turned on the camera. A clear video signal was seen on the TV monitor at our base camp. Cool! There was a lot going on at this point with the system.

Bootstrap Bill flew like an eagle. In all the excitement, I radioed back to base camp for an update on the video.

"We got it! We got it!" was the enthusiastic response.

As I moved out to recover Bootstrap Bill, I asked for an update on the video but was given the order to keep recording. Extremely bad orders! By doing this, I taped over the good stuff. Well, I'm still happy that I recovered the system. It definitely works.

NEW DEVELOPMENTS

A new feature called "Talking Payload" is currently under development. After the microcontroller detects the launch condition, RF messages will be sent to a small RF ground box that will look up a programmed MP3 clip. A PA

system in the LCO area will be hooked up to an FM transmitter that broadcasts on FM band 99.9. So, when the LCO makes an announcement about a rocket, it also will be sent over the FM waves for everyone on the flight line to hear.

A second audio channel on the PA system will enable the rocket to tell the spectators what's going on inside the rocket. For example, as the rocket lifts off the pad, a message from the on-board flight computer will trigger a message to the MP3 ground station to play an MP3 clip: "Launch Detected on Pad 22."

The MP3 clips will be preprogrammed using a text-to-speech program and stored on an SD card. As the rocket hits its apogee, the talking payload system will trigger a "Whoo Hoo!" audio clip. I'll be able to output key data like the peak altitude and speed. This is just another example of a cool feature you can add with your own Zilog-based flight computer. I hope to test the system at a real launch this year.

MORE TO COME

So what's next? The big Level 3. It's a lot harder to fly a Level 3 class rocket. I'll need to plan this one with a Level 3 rocketeer. Plus everything is more costly. I still have a lot to learn as a Level 2 rocketeer. I plan on running many more test flights to make sure everything is working at 100% before I add a flight computer to a Level 3 rocket.



Photo 6—*Bootstrap Bill's payload bay has a video camera. I used my Palm OS cell phone to beam commands into the payload bay using the IrDA transceiver. The IrDA window input is shown on the right side of the payload bay. The "Remove Before Flight" safety flag removes power from the flight computer when it's installed.*

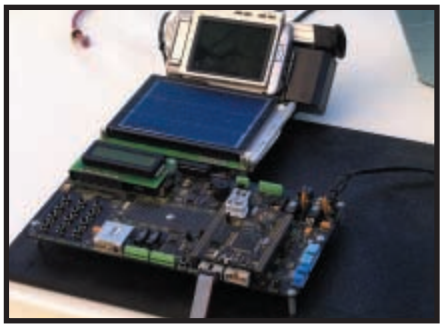


Photo 5—*I used this central command station at our base camp. It includes an eZ80 MCU, a 240 × 320 graphic LCD, a MaxStream 900-MHz RF module (mounted on back), and a video camera that records the downlink video from ZSAT II. The rocket's status and its commands can be controlled from this station.*

High-power rocketry has come a long way over the years. My passion for embedded control electronics and firmware has turned this activity into an addiction that drives me to dream up new widgets. What can I dream up next for this sport? The PDA handheld device can serve as a great tool. I plan on refining ZSAT II's basic hardware platform.

A lot of work still needs to be done with respect to the process of recovering the rocket when it lands. Adding MP3 voice prompts to the system would be a great start.

Another area I am thinking about is video (and still imaging on the ground and in the air). I envision a number of cameras working together to document my missions from numerous angles.

Blending devices and systems that weren't originally intended for use in the high-power rocket market is a fun and rewarding process. With my PDA cell phone, I can launch and recover my high power rockets. This was a technological advance that just had to be done. One small step for BootStrap Bill, one giant leap for PDAs.

Author's note: Three levels of high-power rocketry certification were set up in 1995 according to the National Fire Protection Association's (NFPA) code 1127. Some states haven't adopted it yet. Check your state's laws before attempting to launch a high-power rocket. The federal government regulates high-power rockets and motors. You must apply for and receive an FAA waiver before launching high-power rocket. You must also apply for and receive a BATF LEU permit before taking receipt of any rocket motors exceeding 62.5 g of propellant.

Steven Pope has a B.S.E.E. and holds five U.S. patents. He is a fellow at Zilog, where he focuses on embedded microprocessor applications. Steven's hobbies range from astronomy to model trains and high-power rockets. He maintains a web site with some of his projects at www.geekfreaks.com. You may reach him at stevens-designs@comcast.net.

PROJECT FILES

To download the code, go to ftp.circuitcellar.com/pub/Circuit_Cellar/2005/181.

RESOURCES

Alriss OpenClass and CanSat Projects, www.arliss.org.

Association of Experimental Rocketry of the Pacific, www.aeropac.org.

Extreme Performance Rocket Ships, www.xprs.org.

National Association of Rocketry, www.nar.org.

National Fire Protection Association Code 1127, www.normas.com/NFPA/PAGES/NFPA-1127%2802%29.html.

Tripoli, www.tripoli.org.

SOURCES

ADXL250 Accelerometer
Analog Devices
www.analog.com

High-power motor casing
Dr. Rocket
www.drrocket.com

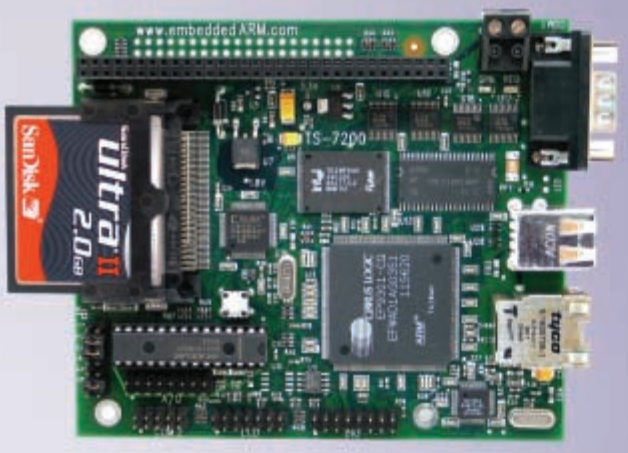
XStream wireless OEM module
MaxStream
www.maxstream.net

MPX4100 Pressure sensor
Motorola
www.motorola.com

GPS Module
Trimble
www.trimble.com

Z8F642 MCU development kit
Zilog
www.zilog.com

200 MHz ARM - PC/104



Shown with optional A/D converter, Compact Flash and RS-485

TS-7200

$149
$119 qty 100

- 32 MB SDRAM (64 MB optional)
- 8 MB Flash (16 MB optional)
- Compact Flash socket
- 10/100 Ethernet
- 2 USB ports
- 20 Digital I/O, A/D options
- 2 Serial Ports
- Support for Windows® CE

- EP9302 ARM9 processor with MMU
- Boots TS-Linux from on-board flash in 5 seconds
- TS-Linux for ARM is open source GNU/Linux
- Low power +5v @ 400mA ▪ 3.8" x 4.5"

Linux 2.4 NetBSD

Windows is a registered trademark of Microsoft Corporation

200 MHz ARM - PC/104



Shown with optional A/D converter, USB Flash drive and RS-485

TS-7250

$149
$119 qty 100

- 32 MB SDRAM (64, 128 MB optional)
- 32 MB Flash (128 MB optional)
- 10/100 Ethernet
- 2 USB ports
- 20 Digital I/O
- 2 Serial Ports
- 5 ch. 12 bit A/D (optional 13 channels)

- EP9302 ARM9 processor with MMU
- Boots TS-Linux from on-board flash
- TS-Linux for ARM is open source GNU/Linux
- Low power +5v @ 400mA ▪ 3.8" x 4.5"

The 256 MB USB drive shown holds our Debian 3.0 self-hosting development environment for ARM. Booting to the development file system allows code development to take place directly on the target hardware eliminating any issues with transfering newly compiled files from the desktop to the target device.

Cross-compiler tools are also available for Windows and Linux.

www.embeddedx86.com

(480)-837-5200
16610 E. Laser Drive #10
Fountain Hills, AZ, 85268 USA

Technologic SYSTEMS

133 MHz 5x86 - PC/104

TS-5600
$279



- PC/104
- PCMCIA slot
- 10/100 Ethernet
- SPI bus
- 2 Serial Ports
- Compact Flash
- 24 Digital I/O
- 12-bit 8 chan. A/D option

- AMD Elan 133 MHz 586 with 32 MB RAM (64 MB optional)
- PC compatible: BIOS, DOS, RTC, I/O & memory map
- TS-Linux 3.0 is open source GNU/Linux - Glibc 2.2.5
- Power +5v @ 830mA
- 4.3" x 5.6"

Shown with optional flash modules, A/D, RS-485 and Merlin cellular modem

33 MHz 386 - PC/104

TS-3400
$159



- PC/104
- Low EMI
- DiskOnChip
- DiskOnChip
- 37 Digital I/O
- 2 Serial Ports

- 33 MHz Intel 386EX with 8 MB RAM (16 MB optional)
- DOS-ROM or TS-Linux pre-installed
- PC compatible: BIOS, DOS, RTC, I/O & memory map
- 1 MB flash drive
- Power: +5v @ 500 mA

Shown with optional flash module and RS-485

PC/104 Ethernet Adapter

TS-ETH10 **$49**

- PC/104 daughter board
- 10baseT Ethernet
- DOS, Linux and NetBSD drivers
- Link and activity LED
- Header for external LED hookup
- 3.5" X 1.6"
- Jumper selection allows up to 4 boards per system



Shown with optional 16 bit connector

PC/104 Peripherals

Enclosures $85
Metal enclosure for Single Board Computers provides regulated +5 VDC from 12-38 VDC input power. Connections available at back panel. Suitable for office or factory.

TS-SER4 $89
Four PC compatible COM ports using the 16550 UART with 16-byte FIFOs, full handshaking on two ports, RS-485 option on two ports, jumper configurable.

TS-DIO24 $39
24 0-5V DIO lines, opto-22 compatible 50 pin connector, 4 outputs can drive up to 30V @1 Amp, 4 inputs are IRQ capable, I/O address jumper selectable.

TS-Modem $129
33.6K baud, jumper configurable as COM1-6, supports IRQs 3-15, V.32 V.42 compliant, standard AT commands, caller ID, 56K baud optional.

Technologic SYSTEMS

- PC/104 Computer Boards
- Enclosures
- PC/104 Peripherals
- Custom Designs
- Exceptional Support

TS-Linux v3.0

- Introducing TS-Linux version 3.0 Our latest distribution is built from open-source featuring a 2.4.18 kernel, Apache web server, FTP client/server, bash, etc. With a footprint of 15 MB or less this is a great place to start your application.
- The TS development environment compiles application code using the libraries and headers installed in TS-Linux. Applications developed on your desktop computer copy directly.
- Our User Mode Linux patched kernel runs TS-Linux 3.0 kernel and file system as an application on your desktop computer during development.

Real-Time Pulse Generator

Access to faster PIC processors inspired Tom to revisit an old pulse generator design. The result is both a practical construction project and an exercise in real-time programming.

Back when I quit my day job and became a full-time electronics consultant and writer, I had to find a way to acquire test equipment. The solution I adopted was to develop and build my own. Among other things, this gave me an excuse to write programs for the then recently introduced PIC microcontrollers from Microchip Technology.

I realized that if I could build an instrument, you could too. Thus, it became my practice, after building something, to write an article about it. My intention has been to provide others with interesting and useful construction projects and, perhaps, to teach a little about electronics and programming. Of course, if a manufacturer would like to license one of my designs, I won't turn them down.

PULSE GENERATOR 1

One of my first projects was a TTL-level pulse generator based on a Microchip PIC16C55. A pair of two-digit, thumb-wheel switches and a range switch enabled it to generate pulses from 0.1 µs to 0.99 s long. These were synchronized to the 20-MHz crystal clock that drove the PIC16C55. One problem I had to solve was how to select and generate pulse lengths in 0.1-µs increments with a processor that took 0.2 µs to execute an instruction. I designed a hardware solution that appeared in the December 1998 issue of the defunct *Electronics Now*.

For a while I sold preprogrammed PIC chips for my projects, but doing that was more trouble than it was worth, so I stopped some years ago. Interestingly, a former *Electronics Now* reader recently tracked me down and requested a pulse generator chip. So, I dug out my old code and programmed a PIC16C55 for him.

TIME TO UPGRADE?

This unexpected interest and the recent availability of 40-MHz versions of the PIC16C5*x* series prompted me take another look at my old design. Would the faster chips enable me to design a PIC-based pulse generator with less hardware and an improved specification?

The original code was probably the most intricate real-time embedded program I'd ever written. Even working from the source listing, it wasn't immediately clear what did what and how things could be changed. I ended up writing a much more straightforward program only distantly related to its predecessor.

REALLY REAL TIME

In both pulse generators, the microcontroller reads the thumb-wheel switches in real time as it generates output pulses. A rotary switch selects the time scale, and the code combines two different programs within the PIC16C55's 512-instruction memory.

Originally, the faster program generated pulses with periods from 5 to 99 µs and lengths from 0.1 to 9.9 µs. One of 10 code segments set a coarse length from 0 to 9 µs. This was followed by one of five code segments to set a fine length from 0 to 0.8 µs. The former switched a port bit from low to high; the latter switched it low again. The 15 code segments, spaced at 16-instruction intervals, occupied almost all of the 256-instruction space, which, in the PIC16C55, can be accessed by a computed `GOTO` instruction.

The inherent instruction time resolution was 200 ns, but some ingenious hardware extended the generated pulse by an additional 100 ns as required. This took a second output pulse, negative going and 0.2 µs long. Both pulses were driven low together, not by clearing port A, but by swapping port A's nonexistent upper nibble into its lower nibble. Thus, I avoided having to set the 0 flag bit. This was typical of the trickery the project demanded.

The second program handled the five slower ranges. Dividing the external clock down in decades lengthened the time scale. On the slowest range, the PIC16C55 ran from a 20-kHz clock. This program's inherent period range was 10 to 990 µs with pulse lengths from 1 to 99 µs. It wasn't pretty, but it worked.

FASTER & SIMPLER

Recent revisions of the PIC16C5*x* chips can, at least in their windowed versions, run from a 40-MHz clock rather than at the 20 MHz maximum of the original models. They can generate pulse lengths in 100-ns increments without external help. My new generator runs from a half-size 40-MHz crystal oscillator.

Using a faster PIC enabled me to improve the minimum pulse period specification from 5 to 3 µs. It also enabled me to use software rather than a hardware clock divider to make range changes. Originally, five different clock frequencies went to a front panel switch and the selected one came back down. You can get away with this at 20 MHz, but it's a bit iffy at 40 MHz.

Only two PIC pins are available for range inputs, so this version has only four ranges. I'd hardly ever used the two slower ranges, so I thought eliminating them was a reasonable compromise. The range switch sets the basic time unit to 1 µs, 10 µs, 100 µs, or 1 ms. The period can be any two-digit multiple of this unit subject to a minimum value of 3 µs on the 1 µs range. The pulse length also has two digits,

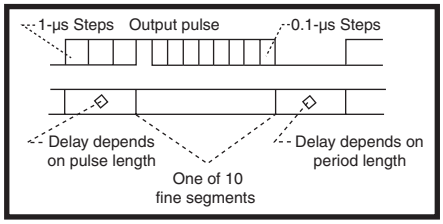


Figure 1—*On the fast range, the coarse length segments come first. (The time isn't to scale.) On the slower ranges, it was easier to reverse this order.*

but there's a decimal point between them (i.e., on any given range, the maximum pulse length is a tenth of the maximum pulse period). With a switch resolution of 0 to 99, selecting a suitable decade can set many useful combinations.

CHANGE OF PHILOSOPHY

The 0.1-µs increments are generated by a computed jump to one of 10 fine-length code segments. These are all 12 instructions long, but they're padded to 16 instructions to simplify indexing. This leaves insufficient code space for 10 coarse-length routines. I had to use a common coarse-length routine and transfer some tasks to the fine segments.

All other things being equal, a double-speed processor halves the minimum pulse period. Because the period switch sets microsecond units, the minimum period had to be either 2 or 3 µs. The number of instructions per loop would be a little higher in this version, so a 20-instruction loop just wouldn't work. A 30-instruction loop was feasible; that's where the 3-µs minimum period came from.

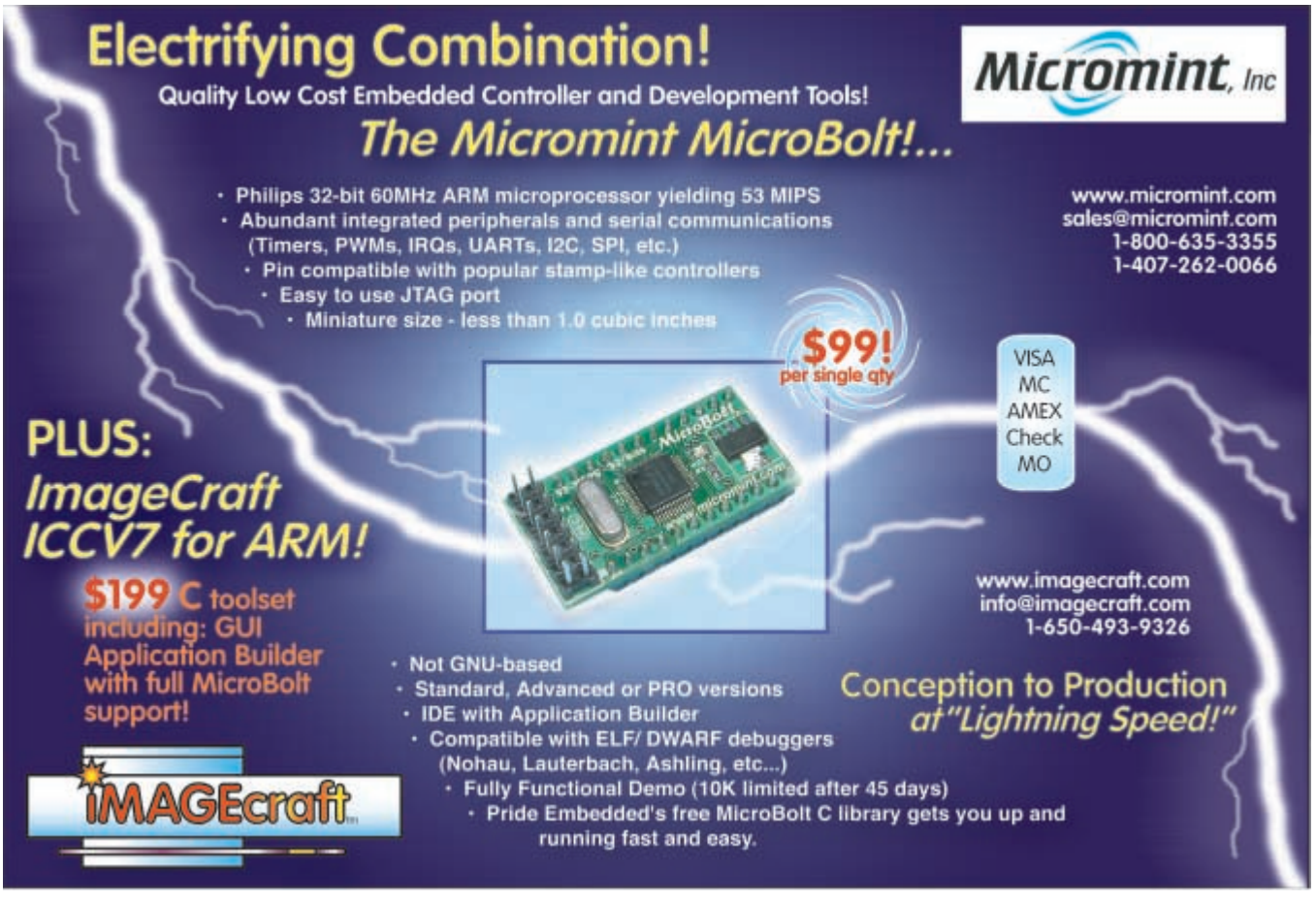
The diagram in Figure 1 is an outline of the pulse generation process. Coarse segments can have many different lengths. Fine segments all have the same length and are almost identical. Only the point where the pulse goes low varies from segment to segment. If the coarse length digit is zero, the pulse starts at the first instruction of the relevant fine length routine and ends less than 1 ms later. (The zero-length fine segment starts with a pulse-down instruction.)

After the fine code has lowered the pulse, the common code takes over. This inserts a variable off time delay with a pulse-up instruction at position 20. This is followed by the variable on time delay. If the coarse-length digit is zero, both the pulse-up instruction and the delay code are bypassed. The loop repeats by jumping to one of the 10 fine-length routines according to the fine-length switch setting.

The pulse-off time may be as short as 1.1 µs. If the period is, say, 5 µs, any length from 0.1 to 3.9 µs is valid. If you need a longer high-level pulse, you can use the inverted TTL output. For example, setting a length of 0.1 µs gives you a pulse that's high for 4.9 µs out of every 5 µs.

CONDITIONAL ARITHMETIC

My first working version of the code did arithmetic, but only when required. For example, if the coarse pulse length was zero or one, no extra on time delay was needed. Setting a pulse length greater than one added a 1-µs delay per unit and decremented the off time counter. The fine period switch was read and added during the fine length routines. Allowing for the 3-µs minimum loop cycle gave an off



time count equal to the period minus the coarse length minus 2. A decrement-and-skip-if-zero instruction tested the count and bypassed the off time delay as necessary.

The coarse period digit, multiplied by 10, had to be added to the off time counter. This calculation and addition took too long for the basic loop, so my initial solution was to do the calculation only when there was a non-zero tens digit. In that case there was inherently time available to process the digit. This calculation added 1 µs to the pulse off time, so the off time count was reduced by one to compensate. This worked, except when the period was 10 or 11 µs. Then the 1.1 µs minimum off time ought to permit setting pulse lengths up to 8.9 and 9.9 µs, respectively. The extra 1 µs of off time limited the pulse lengths to 7.9 and 8.9 µs. This introduced a small anomaly in the specification and provoked me to rewrite the code.

KICK SELF TIME

After spending some days immersed in flowcharts and trial coding, I'd concluded that a fix was impossible. Handling one special case (digit = 0) was feasible. Adding another case (digit = 1) required too many tests and jumps. Having squeezed the most I could out of the instruction set, I finally asked myself which of the chip's features were not being used. One was the index register.

Early on I'd considered using a look-up table to add digits pre-multiplied by 10. After all, why multiply in the program when the assembler can do it for you? But I dropped this idea because the usual table look-up structure—CALL X and RETURN N—would have chewed up far too much time. I'd totally overlooked the fact that the PIC's index register can address a table. The start-up code that loads the table into RAM does all the necessary multiplications.

This slip was so silly because I was already using a preloaded register to add a constant to the W register with one instruction. (The PIC16C55 has immediate logic instructions but no immediate arithmetic instructions.) Putting the coarse period digit plus 16 in the index register made the add-a-constant instruction into add-one-of-10-registers. Each is preloaded with a multiple of 10 less a 3-µs offset.

After I finished kicking myself for the wasted effort, I replaced the existing code with eight straight-line instructions without a special case or skip instruction in sight. I even had an instruction left over that I used to improve the range switching. Very sweet!

SLOWER AND EASIER

The three slower ranges, although simpler to implement using standard programming techniques, also presented some tricky problems. This time the minimum pulse length was 1 µs and the minimum period was 10 µs. The minimum off time was also 1 µs. A 9-µs pulse length in a 10-µs period is valid.

Here the solution was to put the fine-length control first. I wrote 10 µs worth of code that starts with "pulse down" and has an "Are we there yet?" instruction every microsecond. The fine pulse length countdown raises the output pulse N µs from the end of the segment. If the period is 10 µs, the same segment is repeated endlessly. Longer periods are achieved by inserting up to 98 dummy 10-µs segments. Each of these also starts with a pulse-down instruction, but it is skipped unless the coarse-length counter has reached zero. There's no need to compute the off time because the length and period counters are independent.

SMART DUMMIES

A dummy segment performs two jobs: it ends the pulse when the coarse length runs out and decides whether to return to the main segment or insert another dummy. Each dummy segment is padded by a 10-µs wait loop.

There's just one teensy fly in the ointment. If the fine pulse length is zero, the output pulse begins at the start of the first dummy segment in the time slot that's usually used for turning off pulses. That means there's a three-way choice to be made at the start of each dummy segment: turn off a pulse, turn on a pulse, or do nothing. This takes carefully counted branching code. It also requires keeping track of which is the first dummy segment. All this chews up instructions and hence time.

AND A BUSY MAIN

The 10-µs main segment is long enough to read the switches, load the length and period counters, and test if your setup is valid. There's also time in each 1-µs block of code to insert a conditional subroutine call. This adds a time delay of 9 or 99 µs to every 1 µs to provide the two slower ranges. (No, I didn't borrow this algorithm from a popular operating system.) The time delay in the middle of each dummy segment is also greatly increased on the two lower ranges.

Only the fine-length parameter is used during the main segment itself; it's decremented to see when to start the pulse. The period parameter is decremented at the end of the segment to see whether one or more dummy segments should be inserted. The coarse-length parameter is only used within a dummy segment (not in the main segment). As a consequence, the values of the period and the coarse length switches can be read anywhere during the main segment, provided their processing doesn't impede the fine pulse length computation.

As the fine length parameter is decremented within the segment, the first thing the code does is copy the length calculated during the previous segment into a working register. This leaves the first register free for a new switch input.

The time available between pulse-edge tests comes in blocks of five instructions. Processing is broken up in such a way that the content of the W register need not be preserved between blocks. (Generating a pulse edge doesn't affect it, but inserting a time delay does.) The process of computing the difference between the pulse length and the pulse period that sets the valid flag occurs after these parameters have been calculated.

The port A bit that indicates an invalid setting and disables the output may change only when the pulse is off. This requires the flag to be tested and the port A bit to be set during a 1-µs period near the start of the segment. That's the only point when the pulse is guaranteed to be off.

There was a block of five instructions free, so I added a test for a zero pulse length setting. This also set the invalid flag. As it happened, the fast code generated no output in this case, but the slow code did. If you do something

U-VERSION VFD

Noritake

Drop-in Replacement for LCD Modules

The new family of U-Version modules is compact, low power and lower in cost with a custom designed chip. This new VFD technology has an 8 and 4 bit parallel interface and enables the replacement of LCD's with Noritake U-Version VFD modules.



LCD



VFD

UPGRADE NOW...

Brightest Display
Widest Viewing Angles
Widest Temperature Range
Fits Right In
No Programming Change
Low Cost

www.noritake-elec.com/U-Version

Noritake Co., Inc. 2635 Clearbrook Dr., Arlington Heights, IL 60005 phone 1-800-779-5846 e-mail electronics@noritake.com

dumb, it's better to get no output and a warning than an unexpected output.

RANGE SWITCHING

The last little job the main segment does is test the range switch. The four-way range switch has two common contacts connected to the two port pins. The appropriate single contacts are connected to 5 V to give a binary Grey-coded input.

The fast software tests both bits and jumps to the slow software if either are a one. This uses only three instructions. The slow software does a conventional mask-and-test-if-zero to set its internal flags or to return to the fast range.

GETTING IT RIGHT

Debugging time-critical code of this nature is an interesting exercise. I added index numbers to each instruction as I wrote the code. This enabled me to check, for example, that alternative paths contain the same number of instructions and that extending the period adds the correct loop length.

The final test criteria were the output pulse lengths and periods themselves.



Photo 1—*The 74AC74 output chip supplies an extremely clean 2.5-V, 100-ns pulse into 50 W.*

Checking a period is easy enough with a frequency meter, although I needed a six-digit meter to track down some one-instruction timing errors in my original code. On the fastest range, the pulse length is never longer than 9.9 µs. A timing error of one instruction (100 ns) is easily visible on the oscilloscope. On the slower ranges, where a pulse can be from 990 to 99,000 instructions long, lengths must be checked in stages. I compared a pulse length on one range with a known pulse period on the next faster range. This enabled me to compare two time intervals against each other rather than comparing each against the possibly erroneous oscilloscope time base.

OPERATOR ERRORS

On the fastest range, a negative result in the off time register indicates that an invalid combination of length and period has been set. This lowers a port A bit that both lights an LED and disables the output driver during the output pulse. (The alarm signal is high for 0.6 µs during the pulse-off time so the LED brightness depends somewhat on your setting.)

The slower ranges had more time available for testing the setting. Here the length and period are compared directly and the output is disabled if the combination is invalid. Setting a zero pulse length is also flagged on all ranges.

OUTPUT DRIVER

A PIC chip produces extremely clean and fast output pulses. For some applications, a PIC could act as a stand-alone pulse generator. (Should only a limited number of lengths and periods be required, an 18-pin PIC16C54C/JW would suffice.) For my full-function, bench-top instrument I added an output driver chip, a 74AC74 dual D flip-flop. This supplies well-synchronized normal and inverted outputs with rise and

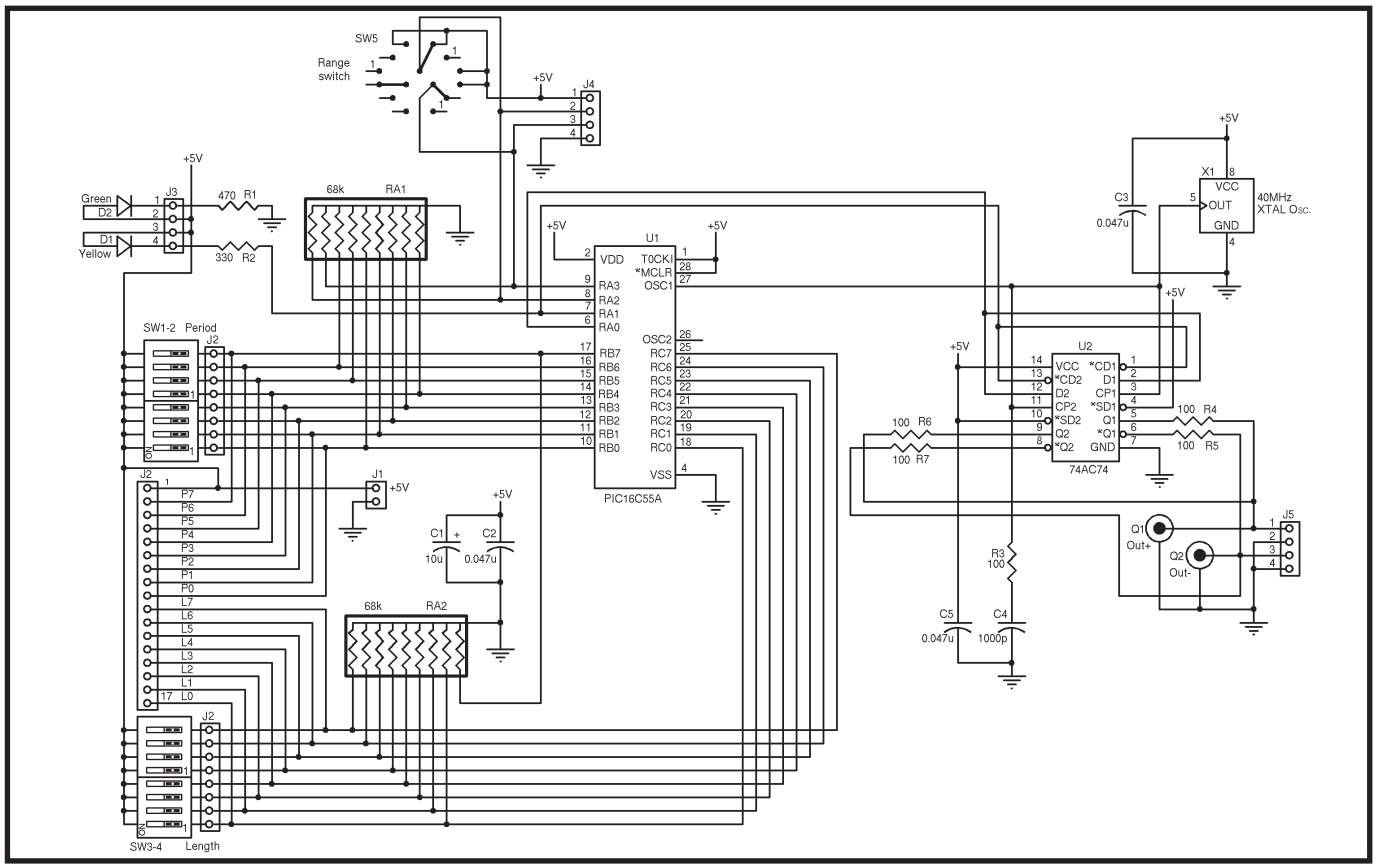


Figure 2—*The schematic may look complicated, but there are only two chips among all that wiring. When wired as shown, the on-board connections are short and direct.*



Photo 2—*I gave my old pulse generator box a bright new front panel label and completely new works.*

fall times less than 5 ns. It serves another purpose, too. If something nasty is connected to the output connector, it'll blow a $0.25 chip, not the PIC processor.

The pulse outputs swing from 0 to 5 V. I added a 100-Ω resistor in series with each flip-flop output and connected them in pairs to give at least 48 mA of output drive current from a 50-Ω source. As well as output short-circuit protection, these resistors provide series termination when driving an open-ended cable. With a terminated 50-Ω cable, you'll get 2.5-V pulses across a 50-Ω load. Note that 150-Ω resistors can replace 100-Ω resistors if you habitually use 75-Ω cables and terminators.

For digital applications, an output cable back-terminated by the generator's 50-Ω output impedance gives an acceptable full-height logic signal marred only by a little overshoot on the front and back edges of the pulses. Overshoot is undesirable in analog applications such as amplifier and filter testing because it's difficult to know how much is due to the device under test and how much is coming from the generator. As you can see in Photo 1, the overshoot disappears when a terminator is used.

Note that although an invalid setting clamps the normal output low, it forces the inverted output high. This can cause damage, for example, if the inverted output is being used to turn on a switch transistor in a DC-DC converter breadboard. The output clamping isn't perfect. I found that a few hundred millivolts of clock breakthrough appear on the output during the erstwhile pulse on time. This probably doesn't matter in a logic application, but it's something to be aware of.

TIMING IS EVERYTHING

One problem I ran into was that the relationship between the 40-MHz clock into the PIC and its output transitions isn't well defined. The output flip-flop needs a data input that arrives more than 3 ns before a positive clock edge. The PIC's propagation delay makes transitions arrive approximately 20 ns after the clock. New data arrives just in time for the next clock edge. A slow PIC and a touchy output chip could result in some output edges being generated 25 ns too late. For example, a 74ACT74 chip I tried generated 125-ns pulses when a value of 100 ns was set.

The proper solution would be to delay the clock to the flip-flop by approximately 10 ns. This is feasible, but it's an undesirable complication. Another solution would be to invert the clock, adding a theoretical 12.5-ns delay if you ignore the inverter's propagation delay. In the AC logic series, this can be anything from 1 to 7 ns.

I didn't want to add a chip, so I rewired the clock so that it went to the PIC first and then to the flip-flop. I also AC-terminated the clock at the flip-flop. This probably didn't improve the set-up time margin by more than 1 ns, but the problem went away.

CONSTRUCTION TIPS

Figure 2 is a schematic of the new pulse generator. To minimize the amount of labor, I reused my old case and gave it the fresh panel label shown in Photo 2. The circuit board and wiring are new.

This was definitely a job that needs a board with a ground plane. I mounted wire-wrap sockets on the 2.4″ × 2.6″ piece of perforated board shown in Photo 3. This has a copper plane on the underside. I drilled out each component pin location to 1.5 mm to let the sockets sit flush with the top of the board. I then lightly drilled away the copper around each pin location except the few where a pin had to be grounded. Surface-mount 0.047-µF capacitors are connected from the ground plane to the V_{CC} pins; low-impedance bypassing is essential here. A dab of silicone rubber holds the chip sockets in place. Epoxy holds the other connectors.

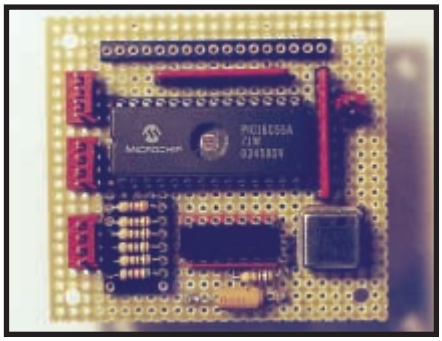


Photo 3—*Take a look at the board's layout. Five connectors link it to the output, power, and control components.*

At these operating speeds, wire-wrap is acceptable if not optimum. (I've seen wire-wrap boards run with difficulty at 150 MHz.) Connections should be kept short and direct. Regular PC sockets and enameled copper wire might be a better way to go.

Other than the twisted-pair output connections, speed-critical stuff is restricted to the circuit board. You can arrange the box layout to suit what you have available. I run test gear from an 11.5-VAC transformer. My original pulse generator had the rectifier/capacitor/regulator combination on the board. More recently, I tended to stick these components on the back of the box next to the power socket (more silicone rubber) and feed the board just the 5 V it needs.

I used a 16C55A/JW, but a 16C57C/JW will also work. The original PIC16C55 chip is too slow. I haven't tried a non-windowed PIC16C55A-20/P to see if it also runs at 40 MHz. The hex code is posted on the *Circuit Cellar* FTP site (PULSGEN2.OBJ). I also included the source code (PULSGEN2.SRC). This is in custom mnemonics, but it shows how the program is put together.

This pulse generator offers an improvement in performance over its seven-year-old predecessor. Its chief merit is that it requires fewer chips to implement a useful test instrument.

Tom Napier designs analog electronics for his clients and practices the forgotten art of code minimization in his spare time—or sometimes vice versa.

PROJECT FILES

To download the code, go to ftp.circuitcellar.com/pub/Circuit_Cellar/2005/181.

SOURCE

PIC16C55A Microcontroller
Microchip Technology
www.microchip.com

Build a Hardware Accelerator

With a little direction from Amir and Wael, you can implement a hardware accelerator within a soft-core RISC processor in a System-on-a-Chip embedded platform. The accelerator is perfect for a vision system application for a mobile robot.

We recently implemented a hardware accelerator within a soft-core RISC processor in a System-on-a-Chip (SoC) embedded platform. You can use our system to enhance the reliability of the vision systems in your robotics applications.

The current prototype includes an Altera programmable device and a Nios II soft-core processor. We designed the accelerator, which performs a morphological edge detection filter, with VHDL code.

Nios II & Cyclone

The importance of instruction set simplicity for building fast computers was recognized in the early 1980s. Since then, the direction of computer architectures has moved toward the RISC, which is a simple load-store architecture. Traditionally, only ASIC designers have had the luxury of constructing new processors with customized features and tuned parameters for specific applications. However, the recent introduction of high-speed, small-die-size, million-gate FPGAs (along with advanced synthesis and simulation tools) has created new application areas for programmable devices (e.g., the design and implementation of embedded RISC processors).[1, 2]

Most large FPGA manufacturers offer either a hard-core or soft-core RISC processor in their product lines. Xilinx's Microblaze and Altera's Nios II are among the most popular IP cores for the quick and easy deployment of RISC-based systems. We used a Nios II soft-core processor and SoC concepts to deploy our embedded system.

To demonstrate the power and flexibility of such a system, we added our hardware accelerator to the processor's core to speed up a specific image-processing operation. Image-processing operators are usually time-consuming. As good examples of heavy processing applications, they can be used to observe the speed increases gained by customized hardware. In this article we'll discuss our algorithm, which is a morphological edge-detection filter applied on an 8-bit gray scale image in the system's memory. The operator was first implemented in software and then designed as a hardware block at the gate level. In conclusion, we'll compare the results to show the speed gained by using the hardware accelerator.

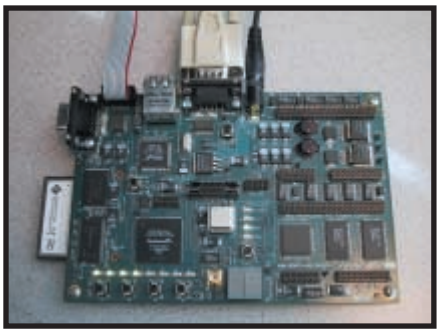


Photo 1—*The Cyclone EP1C20 evaluation board serves as the hardware platform.*

We used an Altera Cyclone evaluation kit for the actual prototype (see Photo 1). There are two connections from the Cyclone EP1C20 evaluation board to the host computer: a JTAG port to download the FPGA configuration file and an RS-232 serial port to download the test image and upload the edge-detected image. The serial port is also used to communicate with the host computer to take user commands and show the results.

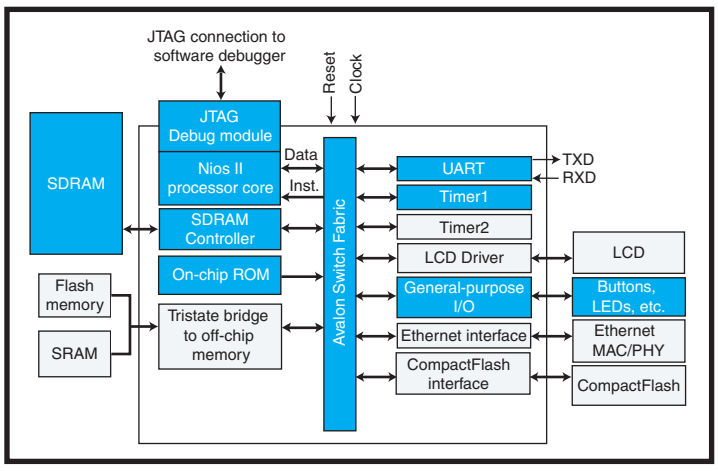


Figure 1—*The processor core communicates with peripheral controllers through a switch fabric interconnect structure called Avalon. All of the controller blocks inside the FPGA are optional. Only the components shown in blue are defined and used in this prototype.*

SYSTEM OVERVIEW

Our system consists of a RISC-based hardware platform with a Nios II designed into the EP1C20 board. The Nios II is a general-purpose RISC processor with a 32-bit instruction set, data path, and address space. It can also access a variety of on-chip peripherals and interface to off-chip memories and peripherals. We used a subset of these peripherals (e.g., an SDRAM controller, UART, JTAG, and timer) in this project.

Figure 1 is a block diagram of all the available components in the Nios II system and on the evaluation kit. The components we used are colored blue.

During the development phase of the project, we used Altera's Quartus II and SOPC Builder, which use a GNU C/C++ tool chain and Eclipse IDE as the software development tools.

All the components in Figure 1 are IP cores that come with the Nios II system. They're actually predesigned modules using gate-level resources in the FPGA chip. Despite being a black box, each module can be customized to some extent. For example, you can set the data rate in a UART module or the memory size and mapping for an SDRAM controller.

After creating all the components, we connected them to the main bus. We then used the SOPC Builder tool to create the complete image and port it to the Quartus tool as a netlist file along with the required time and space constraints. After fitting the netlist file into the FPGA, you can use the logic resources left in the chip for other purposes.

To run and test the actual program, we connected the Cyclone edition evaluation board to the host computer through the serial port. When you do this, the bitmap image is transferred with a terminal emulator software like Microsoft's HyperTerminal or Symantec's Procomm Plus Terminal. The processed image will be transferred back to the host through the same serial link. The user interface terminal is also used to take commands and show the results like the time spent for different modes of image-processing operator implementations.

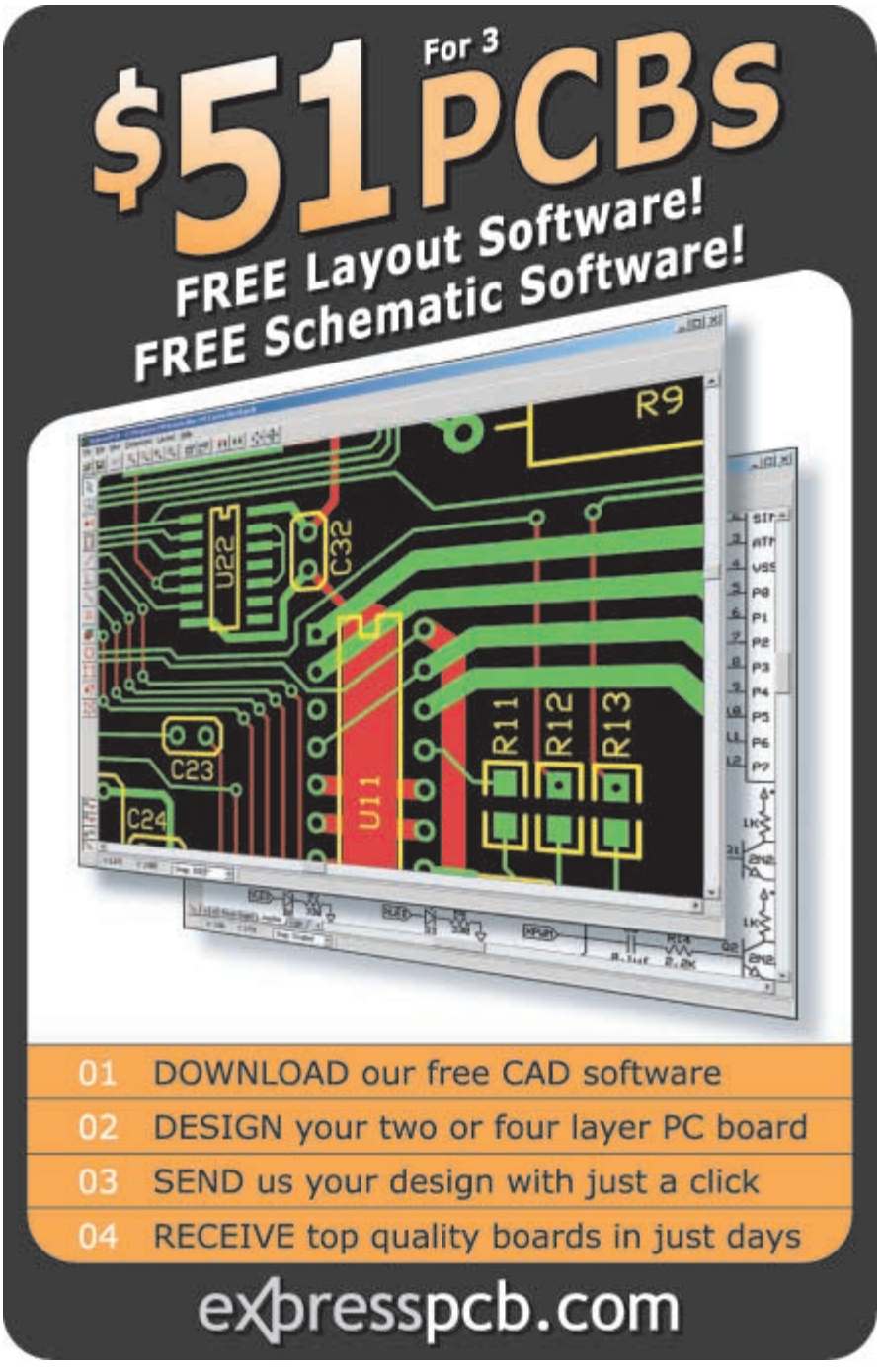
A JTAG connection can be used for downloading the FPGA configuration file as well as providing an interface to debug the code running on a soft-core processor. The program code is separate from the configuration file. It can be downloaded at the same time as the configuration file or any time later via the same JTAG connection.

The embedded Nios II processor is reset to the beginning of the program code every time the system is reprogrammed. The SDRAM chip available on the evaluation board is used for the program and data memory. The program starts by asking for an image file to be transferred from a terminal PC. After the transfer, it shows the information about the image, including height and width, and then asks you to choose either the software or hardware accelerator approach to implement the filter. After you make a selection, it runs the filter, measures the execution time, and sends the processed image back to the terminal PC.

IMAGE PROCESSING

Over the past 20 years, edge detection operators have been extensively analyzed and tested for image processing purposes. Processing an image into a binary or grayscale edge image in order to preserve useful information is a key step in a computerized vision system. The rest of the vision processes can work with the simpler format instead of having to deal with the original grayscale image directly.

Mathematical morphology, which is one of the many approaches used in edge detection algorithms, is based on



set theory concepts of shape. In its simplest form, it's a nonlinear operator that's defined as the difference between morphological dilation and erosion.[3] The dilation and erosion of a grayscale image *f*, denoted by *d* and *e*, respectively, are defined as:

$$d(r,c) = MAX_{i\in D,\ j\in D}[f(r+i, c+j)]$$
$$e(r,c) = MIN_{i\in D,\ j\in D}[f(r+i, c+j)]$$
$$D = (-1, 0, 1)$$

D is the structure element or window size that the operators are applied on; it's chosen as a 3 × 3 window in this implementation. By considering an 8-bit grayscale value representing the brightness of pixels, the dilation operator replaces each pixel with the brightest pixel in its 3 × 3 area. The erosion operator replaces it with the darkest pixel. The edges of objects are identified by taking the pixel-by-pixel difference between the two images formed by the operators. In addition to the filter's computational simplicity, it produces useful and pleasantly surprising results.

The embedded program receives an 8-bit grayscale bitmap file through a serial port. It then extracts the image's information (including the width and height), starts processing image information, and creates another bitmap file replacing the original image with the edged image pixels. The main loop for executing the edge filter looks for the maximum and minimum grayscale values in a 3 × 3 window around the current pixel. Listing 1 shows the main loop in pseudo code.

A time capture function is used before and after the main loop to measure the time spent for executing the filter. The main loop is also written in assembly code in order to avoid any unwanted overhead generated by the compiler.

The same concept is also used to design the hardware accelerator block. But instead of comparing all the pixels in the 3 × 3 window, pixels in a row are compared first and then their results go through another comparison to get the end result for the entire window. That's one of the advantages of using a morphological edge detection filter, which is considered a separable operator.

The hardware accelerator consists of two sets of 8-bit comparators and two

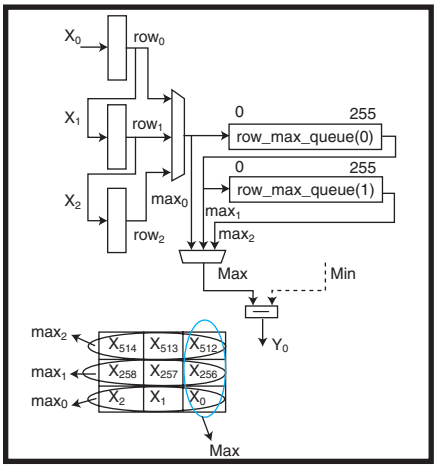


Figure 2—*X_0, X_1, and X_2 represent the current and two previous pixels in a row. max_0 is the maximum of three pixels in the current row. max_1 and max_2 are the maximum values for the two previous rows stored in FIFO queues. Max and Min will represent the maximum and minimum values in a 3 x 3 window around the pixel. Y_0 is the end result.*

sets of 256-byte FIFO memory blocks. Each set is used for computing maximum and minimum values in a 3 × 3 window around each pixel. For the sake of simplicity, only one maximum/minimum pair is shown in Figure 2. Data bytes are clocked into three consecutive latches—row_0, row_1, and row_2—each of which contains the three pixel values in a row. The result is max_0 (or min_0).

In next clock cycle, max_0 (min_0) is shifted into FIFO queues. Because each queue is 256 bytes long, the end of the queues will contain max_1 and max_2 (min_1 and min_2) from 256 and 512 cycles ago. These are the top two rows in the 3 × 3 window (assuming a constant image width of 256). Comparing max_0, max_1, and max_2 generates the final result for the maximum value of the 3 × 3 window. The final minimum result is generated in the parallel block in the same way. Subtracting maximum and minimum values eventually creates the new pixel in the processed image.

By pipelining rows and column processing, you can access and feed image pixels into the accelerator block in a linear way. You don't have to go back and forth through memory addresses. The accelerator block is created with logic resources left over from the SoC design. The Nios II can take up to 1,800 logic elements (LEs). There are 20,000 LEs in an EP1C20 FPGA, so there are plenty of resources left for other functions.

Figure 3 shows a signal simulation used in the design. data8 is the input pixel, and the START signal starts the process. The result is ready in only three clock cycles. The first clock cycle latches row_0, row_1, and row_2 and then calculates max_0 and min_0 through the comparators. Because of changes in max_0 and max_1, the maxi-

Listing 1—*Two nexted loops are used to go through the image and find the maximum and minimum grayscale value in the 3 x 3 neighborhood of each pixel.*

```
Begin_Time = get_time() ;
for all the pixels in image do
    max = current_in_pixel ;
    min = current_in_pixel ;
    for all neighbor pixels in a 3x3 window do
        if ( neighbor_pixel > max )
                max = neighbor_pixel ;
    for all neighbor pixels in a 3x3 window do
        if ( neighbor_pixel < min )
                min = neighbor_pixel ;
    current_out_pixel = max - min ;
End_Time = get_time() ;
Execution_Time = End_Time - Begin_Time ;
```

Listing 2—*Using our hardware accelerator and the related custom instruction, this modified code is the result. Only one loop is used in this version, which makes it much faster than the code in Listing 2.*

```
Begin_Time = dtime() ;
for all the pixels in image do
        current_out_pixel = CUSTOM_INSTRUCTION (current_in_pixel) ;
End_Time = dtime() ;
Execution_Time = End_Time - Begin_Time ;
```

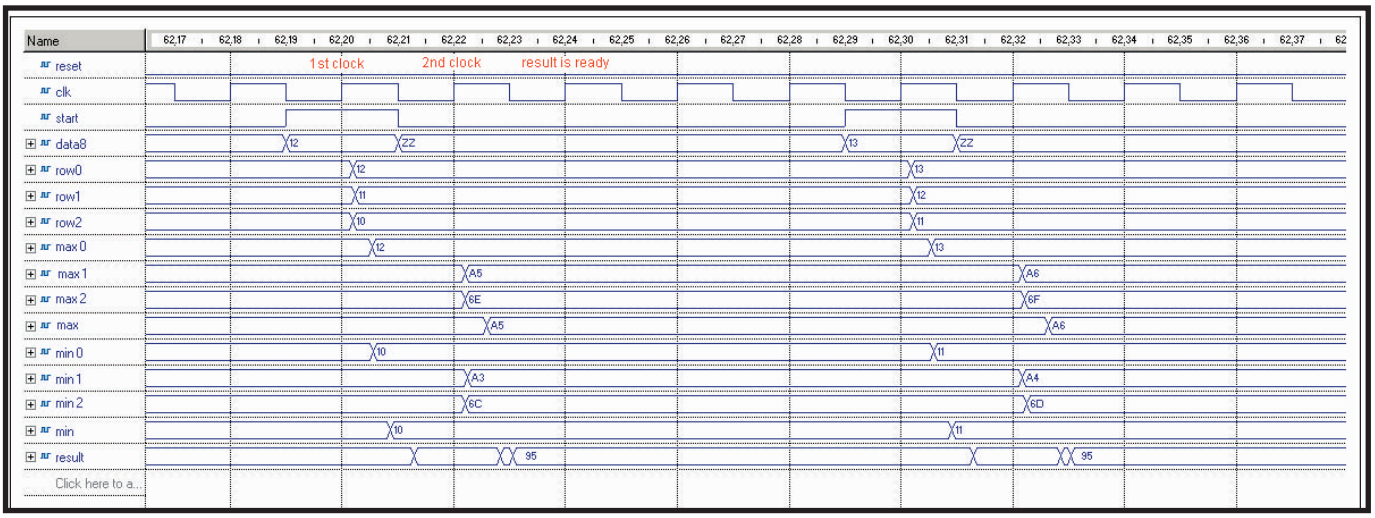


Figure 3—*data8 is the current pixel. All of the other waveforms represent the corresponding signals in Figure 2. The START signal is generated by the ALU, indicating the start of processing in the custom logic block.*

mum and minimum values change, too. It didn't cause any problems for us because we didn't sample the result until the third clock edge.

The second clock edge shifts row queues and obtains new max_1, max_2, min_1, and min_2 values at their outputs. Then, maximum and minimum values change again. The final result is sampled at the third clock edge and sent to the ALU's output buffer. The rest of the operations are handled by the processor's ALU block, which stalls for the first three cycles.

The accelerator block, which was designed in VHDL, is simulated and verified with Aldec's Active-HDL simulation tool. The logic block is integrated into the processor as a part of its ALU. The total number of clock cycles required of the add-on block is set at the system design stage. A new custom instruction, defined to be called in the main program, uses this block instead of the regular ALU path. The main loop in Listing 1 changes to the piece of code in Listing 2, which goes through the input pixels and applies the new custom instruction to each pixel value. This code was written in assembly.

CUSTOM INSTRUCTION

The Nios II's core isn't accessible, so you can't modify it to fit different applications. Nevertheless, you can define blocks of combinational or sequential logics and add them to the ALU core in the processor's data path. You can also define up to four new instructions and add them to the current instruction set. With this instruction, you can reduce a complex sequence of standard instructions to a single instruction implemented in hardware.

As you can see in Figure 4, the custom logic can be attached next to the ALU. It can perform a user-defined operation on the content in the 32-bit R_A and R_B registers, which can be any of the 32 data registers in the processor, and store the result in R_A.

The software interface to the custom instruction extracts the details of the implementation from the programmer. For each custom instruction, the Nios II IDE produces a macro that's defined in the system header file. The macro can be called from a C or C++ application as a normal function call. It also can be accessed in assembly code.

The following is a call to the custom instruction in C code:

```
*out_pixel =(char)ALT_CI_EDGE
  _DETECTOR(*in_pixel) ;
```

Here it is in assembly code:

```
custom 0, r19, r19, r19 ;
```

The total number of clock cycles required for a sequential custom logic block to perform its operation can be defined as either a fixed or variable number of clocks. For variable duration, an extra signal should be provided to indicate the end of operation. For this project we set a fixed length of three for the hardware accelerator block.

The other types of possible custom instructions include an extended custom instruction—in which different operations can be performed by a single logic block through an 8-bit selector field in the op-code—and an internal register file and external interface to communicate with logics outside the processor's data path.

RESULTS

The Nios II processor can be defined and used in three configurations: the fast, the standard, and the economy cores.

The Nios II fast core, which is designed for high-execution performance, includes separate instruction and data cache. It performs dynamic branch prediction and takes less than 1,800 LEs.

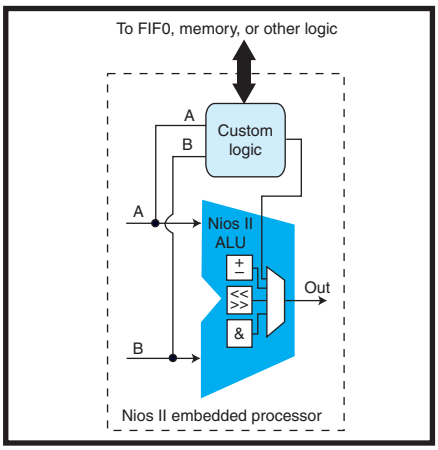


Figure 4—*When you add a custom logic block to the ALU, its output is used whenever the corresponding custom instruction is called. The custom logic block uses memory blocks in the FPGA to create FIFOs.*

	Economy	Standard	Fast
Hardware accelerator	63.698 ms	14.829 ms	13.109 ms
Software implementation	320.969 ms	94.333 ms	39.731 ms
Speed-up	5.04	6.36	3.03

Table 1—*Take a look at the speed results associated with using the hardware accelerator in different implementations of the Nios II processor. In the economy configuration, there's no instruction and data cache. The standard configuration uses only instruction cache. The fast configuration uses both instruction and data cache.*

The Nios II standard core, which is designed for smaller core sizes, has only instruction cache and no data cache. It performs static branch prediction and takes less than 1,400 LEs.

The Nios II economy core, which is designed to achieve the smallest possible core size, doesn't have any instruction or data cache. Nor does it perform branch prediction, It takes less than 700 LEs, which is only 3.5% of the EP1C20.

All three configurations have been implemented and tested for both software and hardware accelerator modes. Table 1 shows the results of applying the edge filter on a 256 × 100 pixel image. The main processor runs with a 50-MHz clock. Image data is read from an external SDRAM chip and written back to the same chip. The program code resides in SDRAM as well. Instructions must to be fetched through the SDRAM controller block.

The increase in speed gained by using the hardware accelerator varies from 3× to 6.4× for different configurations of the processor. Adding instruction cache in Standard mode reduces the delay caused by fetching instructions from external memory. Because the majority of running time in both the hardware and software approaches is spent in the main loop and in a small block of program memory, using the instruction cache dramatically speeds up the performance. Now the processor doesn't need to access the data externally for each instruction fetch. Furthermore, when it comes to small programs, it's much faster to use on-chip RAM, which is created by the memory blocks in FPGA, for program memory.

Interestingly, using cached data in Fast mode helps the software implementation more than the hardware accelerator approach. In the software implementation, half the memory accesses are related to previously accessed pixels in the 3 × 3 window, including the previous pixel and the three pixels in the top row. So, the program reuses data bytes as it goes through the image. Obviously, all the previously used data bytes are cached. This saves processor cycles. The hardware approach doesn't fetch the previous pixels; it goes through the image pixels only once. Therefore, the execution time is almost the same with and without cached data.

Cache memory plays a big role in the processor's performance. In fact, because of the principle of locality and the higher speed of smaller memories, using a memory hierarchy can substantially improve the processor's performance. According to this principle, most programs don't access code and data uniformly. They tend to run through the loops and access the same word (or other words) in the same memory block in the near future (temporal and spatial locality). No wonder using different cached strategies covers a big chunk of the performance-tuning stage of processor design.

The Nios II processor uses the direct map technique for its cached memory, with eight words per line for instruction cache and one word per line for data cache. It also uses write-back and write-allocate techniques for its data cache. The amount of cache can vary from 512 bytes to 64 KB, depending on the available resources in the FPGA.

FINE TUNING

Using hardware accelerators along with a RISC processor in heavy data-processing applications not only speeds up program execution, but also simplifies the final code. It doesn't take that much time and money to set up an FPGA-based evaluation kit as an embedded system platform. An Altera EP1C12 evaluation kit complete with the required software tools costs less than $300. On the other hand, one of the beauties of using a soft-core RISC processor in an FPGA-based embedded system is the excitement of creating your own system and tuning the parameters quickly and easily.

Amir Darvishi is a senior hardware design engineer at MCK Communications, which is part of Citel Technologies. You may contact him at amir.darvishi@citel.com.

Wael Badawy teaches in the Department of Electrical and Computer Engineering at the University of Calgary. You may reach him at badawy@ucalgary.ca.

PROJECT FILES

To download the code, go to ftp.circuitcellar.com/pub/Circuit_Cellar/2005/181.

REFERENCES

[1] J. D. Lucker and V. B. Prasad, "RISC System Design in an FPGA," Proceedings of the 44th IEEE Midwest Symposium on Circuits and Systems, v.2, 2001.

[2] B. Kleinman and B. Cox, "A High-Speed RISC CPU Using the QL16x24 FPGA," WESCON/'93 Conference Record, September 1993.

[3] J. Lee, R. Haralick, and L. Shapiro, "Morphological Edge Detection," *IEEE Journal of Robotics and Automation*, vol. 3, issue 2, April 1987.

[4] Altera, "Nios II Processor Reference Handbook," 2005, http://altera.com/literature/lit-nio2.jsp.

RESOURCES

Altera Corp., "Nios II Custom Instruction: User Guide," December 2004,www.altera.com/literature/ug/ug_nios2_custom_instruction.pdf.

F. A. Pellegrino, W. Vanzella, and V. Torre, "Edge Detection Revisited," IEEE Transactions on Systems, Man, and Cybernetics, part B, vol. 34, issue 3, June 2004.

SOURCES

Active-HDL
Aldec
www.aldec.com

Cyclone EP1C20 evaluation board
Altera
www.altera.com

HOLTEK
www.holtek.com



I/O MTP MCU
HT48EXX

Universal Multi-Programmable MCUs for Intelligent Control



Single pack development kit to get beginners up and running with Holtek devices asap!

- Superior Price/Function Ratio
- Industrial Specifications: -40°C ~+85°C
- High noise immunity and ESD protection
- Internal Data EEPROM
- Wide Operating Voltage: 2.2 ~ 5.5V
- Up to 56 I/O lines
- Compatible with OTP/Mask type devices (HT48E06 excepted)

Advanced C compiler available now!

Part No.	Program	Data Memory	Data EEPROM	I/O	Timer	Interrupt	Pins
HT48E06	1Kx14	64x8	128x8	13	8bitx1	2	18/20
HT48E10	1Kx14	64x8	128x8	19	8bitx1	2	24
HT48E30	2Kx14	96x8	128x8	23	8bitx1	2	24/28
HT48E50	4Kx15	160x8	256x8	33	8bitx1,16bitx1	3	28/48
HT48E70	8Kx16	224x8	256x8	56	16bitx2	3	48/64

Holtek starting as low as $125 USD

New range of development tools and volume production type OTP writers now available !



Order your Holtek In-Circuit-Emulator today and get 10% discount

(valid for USA and Canada customer only; offer expires on 12/31/2005; discount code: EHSIUCC)

HOLTEK SEMICONDUCTOR INC.

USA Sales Office : Holmate Semiconductor, Inc.
46712 Fremont Blvd., Fremont, CA 94538 Tel: (510) 252-9880 Fax: (510) 252-9885
www.holmate.com

Strong-Encryption for PICs (Part 1)

Triple-DES Algorithm

Triple-DES is a widely used encryption algorithm. In this series of articles, Romeo shows you how to use it with PIC MCUs. He also describes how he implemented Triple-DES to create a one-time password generator for an electric lock.

When I was in school, I found philosophers such as Heraclitus puzzling because they seemed bent on demolishing the castles of certainties that we build during the course of our lives:

You could not find out the boundaries of soul, even by traveling along every path: so deep a measure does it have. (Fragment 45, Diogenes Laertius IX, 7)

This could synthesize the internal battle between the head and the heart, between the analytical and romantic sides of our nature. I'm an engineer, so it seems as if my cold, calculating side should always prevail. However, after years of testing and sometimes designing and implementing encryption systems, I am beginning to feel somewhat dazed. I have come to realize that the same algorithm that several products implement in a platform—ranging from the mainframe to the PC—also could be implemented in one of the cheapest, low-range microcontrollers on the market.

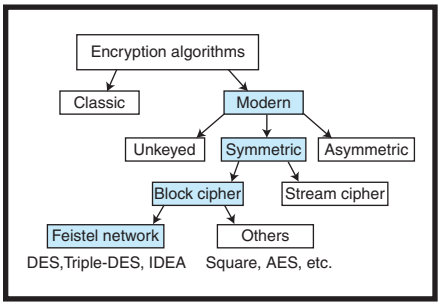


Figure 1—*Encryption algorithms: a simplified taxonomy. A large class of modern symmetric block ciphers is based on Feistel networks.*

To be honest, a comparison is impossible. Usually when implementing an encryption algorithm, speed is one of the most frequently requested characteristics. In this case, because of obvious constraints, the optimization strategy is only focused on the size of component.

Amongst the symmetric block ciphers, the Triple-DES algorithm with an encryption key of 168 bits (56 × 3) is one of the most widely used, even though it isn't particularly new. This month I'll explain how to use the algorithm for Microchip PIC MCUs. To do so, I'll introduce you to an interesting prototype application: a one-time password generator for activating and deactivating an electric lock.

Please note that while working on this project I was mainly interested in the implementation of the algorithm. The application and the protocol should be used only as a foundation for a commercial-strength application.

CRYPTOGRAPHY

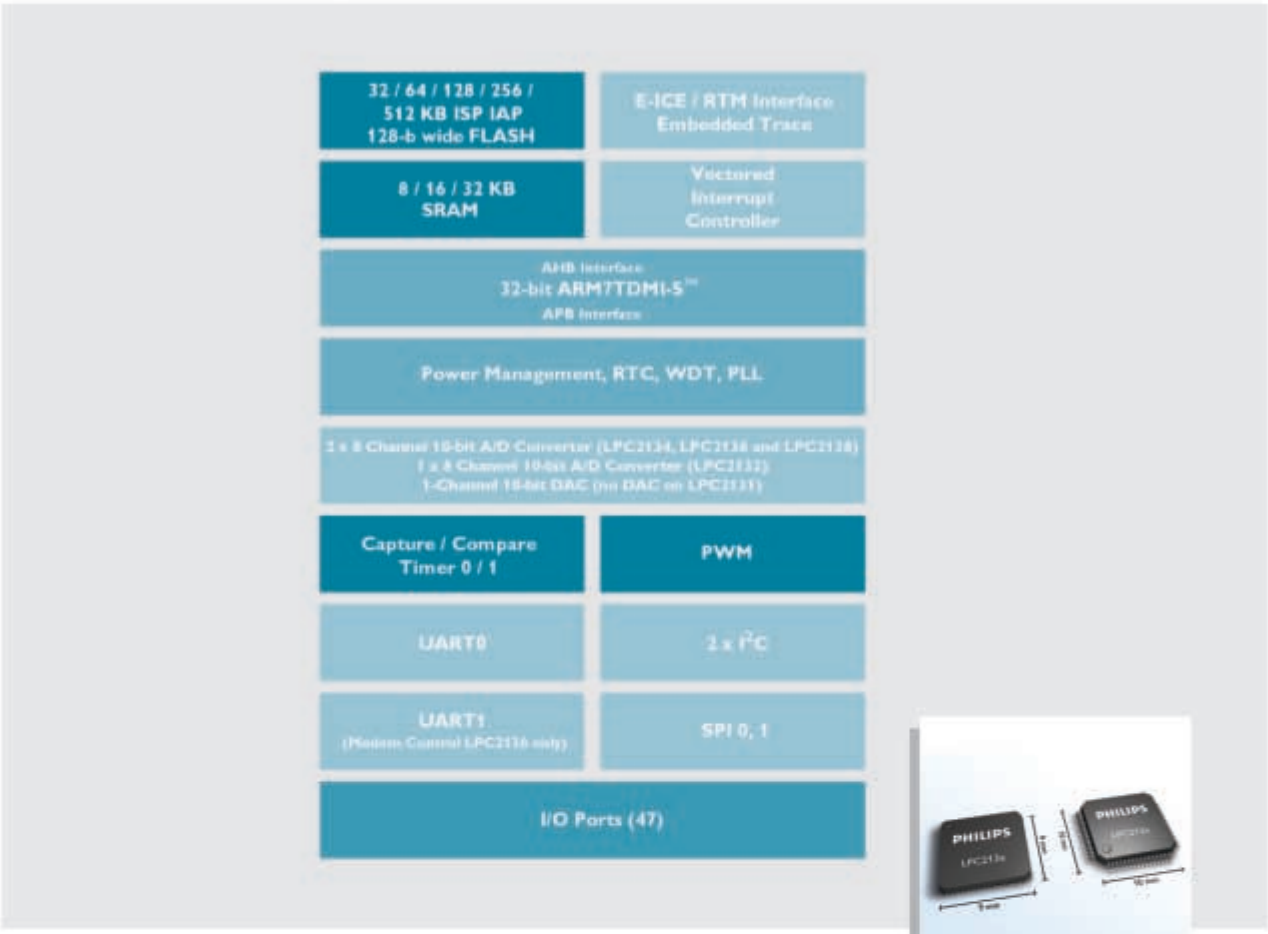
Cryptography, intended to be the art of hiding content, is a science that dates back several thousand years. Approximately 4,000 years ago, the Egyptians made use of secret writing. Julius Caesar used a simple cipher, which is now called the Caesar cipher.[1]

Classical cryptography is of little interest nowadays, except for some techniques that have evolved over the years, such as steganography and some schemes like the one-time pad (or Vernam code) that it is still today an example of a perfect cipher for being theoretically unbreakable. A lot of things have changed. Modern cryptography has a milestone that dates back to 1949 when Claude Shannon pioneered the field with his paper entitled "Communication Theory of Secrecy Systems" (see Figure 1).

Nowadays, cryptography together with cryptanalysis, which is the science of breaking cryptography, constitutes a part of cryptology, which is a branch of mathematics. Cryptography gives solutions to a number of security concerns of computer systems. When using a computer, you normally use a name and password to confirm your identity. Cryptography provides stronger methods of authentication based on digital certificates, for example. Cryptography can also protect data stored on a disk from unauthorized access (confidentiality) and detect modification (integrity). These are a few of the many examples you can read about in the literature listed in the Resources section of this article.

SYMMETRIC ENCRYPTION

When discussing encryption algorithms, the term *symmetric* means the encryption and the decryption of a text have to be done with the same key. Witfield Diffie and Martin Hellman, the two researchers who introduced the new concept of public-key cryptography, described the asymmetric encryption algorithm in their revolutionary 1976 paper entitled "New Directions in Cryptography."



We've squeezed more in... how much can you squeeze out?

The gauntlet has been laid down. Take part in the **Philips ARM Design Contest 2005** and pit your design skills against your peers. Not only will you have the opportunity to win a share of $15,000 in prizes, but you can bask in the knowledge that your design pushes the industry's fastest ARM7 family of microcontrollers to its limit.

First Prize $3000
Second Prize $2000
Third prize $1000
Honorable Mention (8 prizes) $500 each

Plus five $1000 Bonus prizes:
$1000 bonus for the best project incorporating the 2138
$1000 bonus for the best project incorporating the 2136
$1000 bonus for the best project incorporating the 2134
$1000 bonus for the best project incorporating the 2132
$1000 bonus for the best project incorporating the 2131

Deadline for entries: October 24th, 2005.

Just hit **www.circuitcellar.com/philips2005** for the competition rules and entry form.

PHILIPS
sense and simplicity

© Koninklijke Philips Electronics N.V. 2005. All rights reserved.

Public-key encryption algorithms are usually slower than symmetric-key encryption algorithms such as the Data Encryption Standard (DES) or the more recent Advanced Encryption Standard (AES). For this reason public-key encryption is mostly used for authentication (digital signatures) and for the transport of keys subsequently used for bulk data encryption by symmetric algorithms.

Regarding symmetric encryption algorithms, another important distinction must be made between block and stream ciphers. In the former, the algorithm encrypts a block of data as a whole. In the latter, it encrypts individual bits or characters of a plaintext message. When implemented in hardware, stream ciphers, which have simple hardware circuitry, are generally faster than block ciphers. Stream ciphers were used from the beginning of the electronic age, especially in the military. There's a large body of partially undisclosed literature describing them.

Symmetric-key block ciphers are a fundamental component in many cryptographic systems. They provide confidentiality, but they can be used as a building block in message authentication techniques, data integrity mechanisms, entity authentication protocols, and digital signature schemes (based on symmetric-key).

No block cipher is ideally suited to all applications. You must consider several trade-offs when choosing an encryption algorithm. The most common are speed requirements, memory limitations, and security.

Of the many block ciphers currently available, DES, Triple-DES, and DESX are widely used. AES, the new Federal Information Processing Standard (FIPS) proposed as a DES replacement, is widely known and highly regarded. MARS, RC6, Serpent, and Twofish are the other finalists in the AES competition. Their high profiles were earned in part because of the reputations of their designers.

When using a block cipher, you must consider a few things: modes of operation, the initialization vector, and padding. A block cipher encrypts plaintext in fixed-size blocks. The size is 64 in DES and Triple-DES. More recent algorithms adopt a block size of 128 bits.

If you use the Triple-DES algorithm for messages exceeding 64 bits, a problem arises: How do you arrange the encryption process?

The simplest approach is to split the message into 64-bit blocks and encrypt each one independently from the others. This mode (electronic codebook, or ECB) has the worst security properties because identical plaintext blocks (with the same encryption key) result in identical cipher text and this can help a cryptanalysis. For this reason, ECB isn't employed in most applications.

Other common modes are cipher-block chaining (CBC), cipher feedback (CFB), and output feedback (OFB). When security is a main concern, the preferred mode is CBC. In this case, in the cipher text, all the subsequent blocks depend on the previous ones.





The initialization vector (IV) isn't related to the encryption algorithm; it's related to its mode of operation. The IV is a block of random data that's used in most modes of operation to initialize the encryption process. In CBC mode of operation, changing the IV, key, or first plaintext block (e.g., using a counter or random field) results in different cipher text.

The third theme, the padding, arises when the size of the message isn't a multiple of 64 bits. In this case, splitting the message into 64-bit blocks causes the last block to be smaller than 64. For the encryption to be feasible, the last block usually has to be completed in 64 bits. The solution to this problem isn't trivial. If you simply complete the block by adding a conventional character, you won't be able to decide in the decryption phase if the characters at the end were simply added or were part of the decrypted message. Note that the message, normally the plaintext message, to be encrypted isn't necessarily text. It could also contain binary data or be entirely binary (e.g., an executable).

RFC 2040-like padding is common. Pad bytes are added to the plaintext according to a simple rule: they're all the same and set to the binary value representing the number of bytes you're adding.[2] When the block size is 64, between 1 and 8 padding bytes are included. If the length of the message is an exact multiple of 8 bytes, you'll have 8 bytes of padding set to the hexadecimal value 0x08.

Another type of padding comes from PKCS5. The short block is padded with a number of bytes where each single one represents the number of remaining bytes.[3] I won't be covering padding techniques in this article because the padding isn't needed in my application. The message for encryption is exactly two blocks of 8 bytes each.

DES ALGORITHM

The DES algorithm is the result of a joint effort between IBM and the U.S. National Security Agency to protect sensitive unclassified data. Adopted as federal standard in November 1976, the National Bureau of Standards (now called the National Institute of Standards and Technology, or NIST) published the algorithm as FIPS 46 in January 1977. NIST issues standards and guidelines and creates individual projects to provide technical assistance to federal agencies in implementing data encryption.

The standard FIPS 46 was reaffirmed in 1983, 1988, 1993, and 1999. In the 1999 standard (FIPS 46-3), the use of DES was restricted to legacy systems and triple-DES became the new algorithm of choice. At the conclusion of the 2004 review, the NIST decided that the strength of the DES algorithm was no longer sufficient to protect federal government information, so it proposed withdrawing the standard FIPS 46-3 and two related standards (FIPS 74 and FIPS 81). The withdrawal process was completed in May 2005, and the notice was published in the *Federal Register*.

Government agencies with legacy





single DES systems will be required to transition to FIPS-approved versions of Triple-DES (TDEA described in the new recommendation) or AES.[4, 5] Triple-DES with keying option 1—which is the algorithm implemented in this prototype—and the AES will coexist as approved algorithms through the year 2030 to enable a gradual transition to AES. TDEA with keying option 2 was approved to protect government information only through the next five-year review period.

The AES is a new symmetric-based encryption standard developed by NIST. Also known as the Rijndael algorithm, the AES was designed by Joan Daemen and Vincent Rijmen. It's a 128-bit block cipher with key lengths of 128, 192, or 256 bits. AES is intended to provide strong cryptographic security for the protection of sensitive information well into the twenty-first century.

In 1981, the DES was standardized with the code X3.92 as an American National Standard. It was quickly adopted worldwide as the algorithm preferred by the financial community. By using DES in banking transactions, the American Bankers Association developed a set of standards that are now evolving to support AES. Today, it's probably the most widely used algorithm for the protection of sensitive unclassified data. DES uses a 56-bit key to encrypt a 64-bit block and produce a block of the same size.

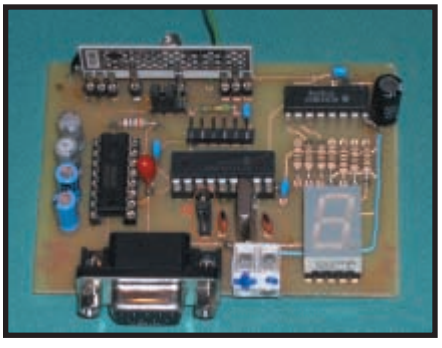


Photo 1—*You can use the jumper J2 to disable the decryption phase in the receiver and to transfer (via an RS-232 connection) the raw data to the PC for further processing.*

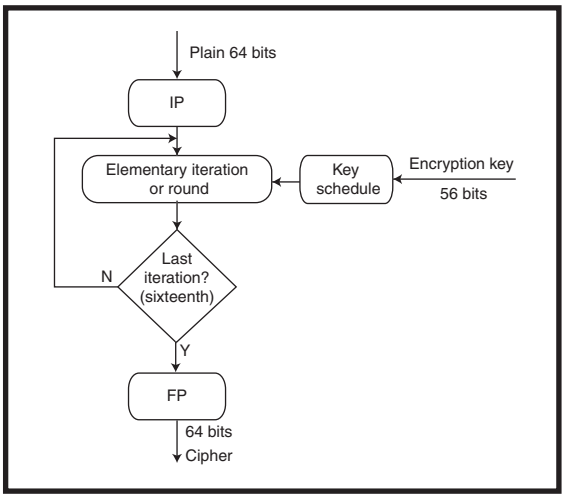


Figure 2—*The structure of the DES algorithm is straightforward.*

The DES core decomposition into fundamental modules appears as shown in Figure 2. The initial permutation (IP) changes the order of the bits of the input data. It has been proven that this block (and the last one, i.e. the final permutation) doesn't increase the security level of the overall scheme. This is the reason why you won't find these blocks in some implementations. In this case, however, the algorithm doesn't comply with the standard.

The key schedule, also known as key transformation, processes the key bits and produces the 16 48-bit sub-keys or round keys (for each of the 16 rounds of DES) that the cipher can use. These 16 sub-keys, when using the algorithm for bulk encryption, are usually pre-computed and stored in memory for subsequent use; in this implementation, because of memory constraints, they are evaluated on the fly in each round.

The fundamental block in DES is the round. DES has 16 of them in which the data is combined with the key. The scheme implemented is called the Feistel network, which was invented by Horst Feistel and used in the IBM cipher Lucifer (a precursor to the DES). It's the basis of most block ciphers published since then, including Twofish, FEAL, and others.

The fundamental component of a Feistel network is the F function, which is a key-dependent nonlinear mapping. After the IP block, the plaintext was split into two blocks of the same size (32 bits): the right (R_0) and the left half (L_0). In each round, the right block and a sub-key are input to F, the output of F is XORed with the left block, and then the two blocks are exchanged for the next round:

$$L_I = R_{I-1}$$
$$R_I = L_{I-1} \oplus F(R_{I-1}, K_I)$$

The criteria followed are simple: it's possible to create a strong encryption algorithm iterating a function, even a weak encryption algorithm by itself. An overview of DES computation path is shown in Figure 3. Two rounds of a Feistel network form a cycle; in one cycle, every bit of the text block has been (ideally) modified once.

After the last round, the left and right words are reassembled and passed through the final permutation (FP or IIP). The L/R Swap block is added just to make it clear that the last round is irregular. The final permutation is the inverse of the initial permutation.

TRIPLE-DES ALGORITHM

The Triple-DES algorithm was proposed as an alternate to DES in 1993, when it became clear that, with a growing weakness of a 56-bit key, advances in computer technology had compromised the security of the DES. A much higher level of security was needed. Each Triple-DES operation is a compound set of the DES encryption and decryption operations. A TDES encryption operation consists of the transformation of a

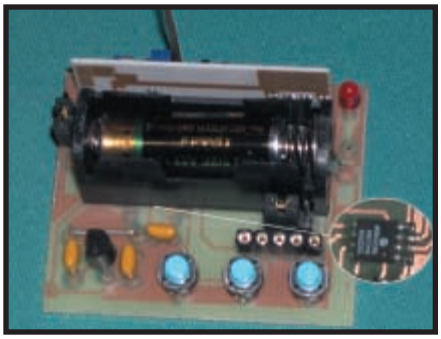


Photo 2—*The transmitter uses few components. I chose the SOIC package for the PIC microcontroller.*

64-bit data block (IN) into another 64-bit data block (OUT):

$$Out = E_{K3}\{D_{K2}[E_{K1}(In)]\}$$

$E_K(IN)$ and $D_K(IN)$ represent the DES encryption and decryption of the block IN using the K key. Note that there are several ways to use DES encryption/decryption three times. They aren't all Triple-DES and as secure.

Similarly, a Triple-DES decryption operation consists of the transformation of a 64-bit data block IN into a 64-bit data block OUT, defined as follows:

$$Out = D_{K1}\{E_{K2}[D_{K3}(In)]\}$$

The standard specifies the following options for the keys K1, K2, and K3. Regarding keying option 1, K1, K2, and K3 are independent keys. Regarding keying option 2, K1 and K2 are independent keys and K3 = K1. Regarding keying option 3, K1 = K2 = K3. In the last case, the Triple-DES algorithm coincides with the DES algorithm, providing backward compatibility. Every eighth bit of each of the keys (Kn with n = 1–3) is ignored (they can be used for parity checking), so only 56 of the 64 bits of each key inputs are considered.

PASSWORD GENERATOR

When you log on to a PC, you normally have to type a user ID and a password. If you connect to your PC through a network, a hacker could steal your legitimate passwords and use them again at a later time. There are some solutions to this problem like using SSH instead of the telnet to access the system.

The problem addressing reusable passwords was perhaps first described in the famous fable "Ali Baba and the 40 Thieves." One-time password systems alleviate the problem of hacked passwords by making the reuse of the password ineffective. As their name implies, one-time passwords can be used only once: it doesn't matter if both the user ID and password were stolen because the password is only valid for one session.

One-time passwords also can be used to reduce the dangers of bad (intuitive) passwords. There are many different one-time password systems available. Some of them require you to carry a hardware device, such as a smartcard or a special calculator. Others are implemented only in software. Some are based on paper. For example, you may be given a list of passwords, and each time you use one, you simply cross it off the list. This is the scheme my bank uses when a home user requests an on-line bank transaction over a certain limit.

All of these one-time password systems provide an important improvement in security over the conventional system. But the one-time password has applications other than accessing a PC via a network.

PASSWORD APPLICATION

My garage door's remote control came with a traditional code set up by micro switches. I changed this by using the Triple-DES algorithm to create a one-time password. When designing this application I wrote down a list of requirements: a strong encryption algorithm fully compliant with a standard; an inexpensive MCU in the transmitter; and a simple transmitter and receiver.

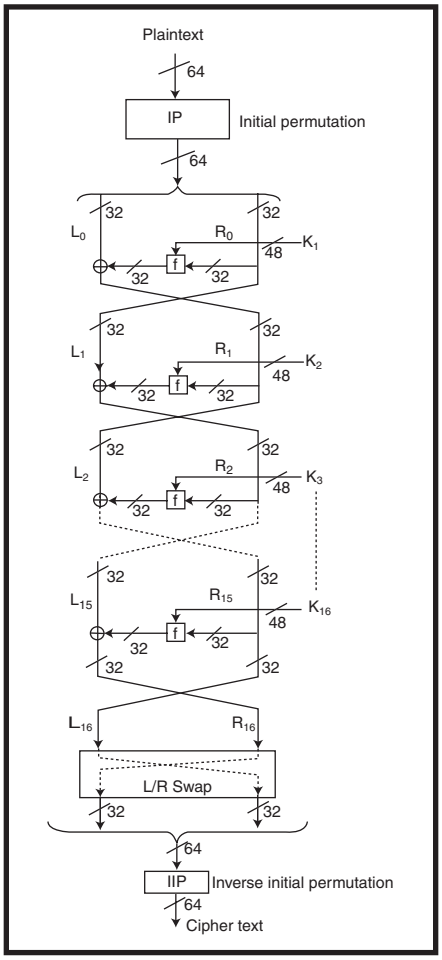


Figure 3— *The expansion of the DES computational path highlights the Feistel network structure.*

The application involves two circuits, a transmitter and a receiver, that communicate over an RF link at 433.92 MHz. You can connect the receiver to a PC through a serial connection (RS-232). I used the PC as a tool to support the testing phase of the application, but by changing a jumper in the receiver, you can transfer an important part of the processing to the PC and use the receiver as a gateway, leaving it to act as an RF receiver and Manchester decoder. Photo 1 shows the receiver.

The transmitter has a fixed number of buttons (see Photo 2). When you press a button, the input states are sent to the receiver. The receiver then decodes the message and sets the outputs accordingly.

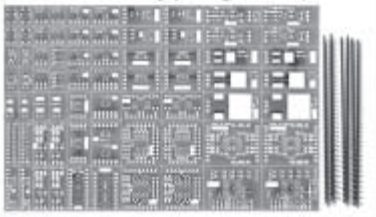
The encoder and decoder's parts are based on PIC MCUs. They're connected to a transmitter/receiver module that takes care of the transmission and reception of digital signals by radio.

Triple-DES was the algorithm of choice. Its chief disadvantage in comparison to AES is speed, but that wasn't a concern in this case. The other two requirements were to show that embedding strong encryption algorithms, even in a low-range MCU, is feasible.

HOW IT WORKS

When you press one of the buttons connected to the MCU, the transmitter prepares two blocks of 8 bytes each (payload) and encrypts them using the Triple-DES algorithm with a 168-bit encryption key (fetched from program memory), CBC chaining mode, and an IV composed of null characters (0x00). It then prepares the header and sends the characters through the RF link with a synchronous serial protocol (Manchester code) that enables the receiver to reconstruct the original message.

The inclusion of two fields (time counter and sequence number)—in the

SMT Prototyping Adapters

Snap-Apart™ PCB's with .100" pin strips
Dozens of assorted adapters on each PCB
A variety of PCB's - patterns on both sides
Eliminate the cost and delay of PCB layout / fab!
SOIC PLCC SSOP QFP DPAK
TSSOP SOT23 MSOP QSOP
SC90 D2PAK SOT89 SOT143
TSOP SC70 SC88 SC326
and many more.

BELLIN DYNAMIC SYSTEMS, INC.
www.beldynsys.com (714) 630-8024
- Rapid Development Solutions for the Technical Professional -

BYRON
HOURMAND, M.S.E.E.
Advanced Embedded Technologies
Hardware & Software
Consulting
20 Years Industry Experience
in the Design of:
- Medical Devices
- RF Transmitters and Receivers
- Security Systems
- Gate Controllers
- Automotive Electronics
- Construction Machine Controllers
- Appliance Controllers
- Remote Terminal Units for oil & gas
- Test Fixtures and more
For a free quote, please contact:
(760) 415-3208
www.AET-eng.com

PIC-SERVO
MOTION CONTROL
MOTION CONTROLLERS FOR
BRUSH, BRUSHLESS AND
STEPPER MOTORS.
• CONTROLLER CHIPS
• CONTROLLER BOARDS
www.picservo.com
JEFFREY KERR, LLC

first 64-bit block to be encrypted (by the transmitter)—guarantees that the cipher text is completely different in two consecutive transmissions. In fact, because of the properties of a strong cipher, a minor change to the plaintext or the key must result in significant and random-looking changes to the cipher text (the avalanche effect).

After achieving synchronization with the transmitter, the receiver must decide which key to use to decode the payload. This information is contained in the header (the Key Group field). In this prototype, there's only one encryption key, but you can use external EEPROM to extend the system's capability.

Following this, the receiver must decrypt the payload using the chosen key and then control the checksum fields to keep the integrity of transmitted data. But if you've decrypted the payload and the TDEA decryption routine has been successful, does this guarantee that the payload was correct (i.e., unmodified and concurrent with the transmitted data)? No. An encryption algorithm like TDEA works with binary data that transforms it into another block of binary data. If you modify and try to decrypt a block of data produced by encrypting a text, the decryption algorithm simply returns to a block of binary data that doesn't correspond with the original text.

If you want to guarantee the integrity in addition to confidentiality, you need something more. The algorithms used to compute the checksums are extremely simple; the checksum fields are 1 byte each. One is the sum (mod 256) of all the other bytes in the payload; the other is the exclusive OR.

After the check at the previous step is successful, the receiver can check the user ID (note that in the prototype there is only one user ID) and verify that the time counter TT in the payload (time measured by the transmitter) is greater than the last accepted value from that user. If this isn't the case, it means that someone is trying to reuse old sequences.

Lastly, the receiver compares TT to the current time measured by the receiver (TR). If the difference between TT and TR is within a 10-min. "window," the user is accepted. The window can be reduced or increased depending on the precision of the two clocks and the level of security you want to achieve. But this is just a simplification. The two counters depend on the time through different proportionality factors. The receiver has to do some calculations to handle the comparison. The two timer counters (in the receiver and transmitter) are 32-bit integers that are incremented every 1 s in the transmitter and every 100 ms in the receiver.

In the prototype, the transmitter's clock is internal. It isn't as stable as the one in the receiver, so I had to include a resynchronization mechanism.

If the "window" comparison step is unsuccessful, you have to push the button again. When the receiver sees two messages from the same user with two consecutive sequence numbers within 30 s (with the obvious constraint that the time must not go backward), it accepts the message and updates the user record.

An additional check (not implemented in the prototype) is to verify whether or not a specific user ID is authorized to issue a specific command. When all of the checks are completed successfully, the command in the command field is executed.

RF

Amplitude modulation is the oldest technique for embedding information on an RF carrier. With this technique, a carrier signal's amplitude is raised and lowered depending on the modulation signal. When the shift equals the amplitude of the carrier, you have on-off shift keying (OOSK). This method of AM modulation is referred to by multiple designations. Besides OOSK, other common designations are carrier-present carrier-absent (CPCA) and on-off keying (OOK).

To demodulate an OOSK signal, the simplest circuit is at least made of a diode detector and a subsequent low-pass filter, which remove the carrier, leaving only the original modulation signal. The benefits of OOSK modula-

tion are its low cost, small size, and low power consumption, which make it practical for various applications. You should, however, consider this technique's principal drawbacks: the low data rate, which depends on the oscillator's start-up time, and the poor noise immunity (noise in the receiver will interfere with data transmission). If the primary requirement were the data rate and noise performance, the FSK or PSK would be a better choice.

TOP CHOICE

DES is one of the most widely used cryptosystems. For years it has been the encryption standard recommended by U.S. governmental agencies. My prototype application shows that implementing the algorithm DES and its stronger successor Triple-DES is feasible even for low-range microcontrollers. Next month we'll take a closer look at the application.

Author's note: In this article I described a well-known IT technique that dates back to 1981. The ISO standard 9798-2 (Information Technology/Security techniques/Entity Authentication) describes the mechanism as well (unilateral authentication timestamp-based). For more information on these topics, please refer to the Resources section of this article.

Romeo Costanzo is a senior information security engineer at Telecom Italia. He holds an M.S. in electronics and has been a software development engineer and information security specialist for 16 years. You may contact him at rcostanzo@acm.org.

PROJECT FILES

To download the code, go to ftp.circuitcellar.com/pub/Circuit_Cellar/2005/181.

REFERENCES

[1] D. Kahn, *The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet*, Scribner, New Jersey, rev. 1996.

[2] R. Baldwin and R. Rivest, "The RC5, RC5-CBC, RC5-CBC-Pad, and RC5-CTS Algorithms," October 1996, www.faqs.org/rfcs/rfc2040.html.

[3] RSA Laboratories, "PKCS #5: Password-Based Encryption Standard," V. 1.5, November 1993, www.rsasecurity.com/rsalabs/node.asp?id=2127.

[4] NIST Special Publications, "Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher," 800-67 V.1, May 2004.

[5] NIST, "Advanced Encryption Standard," FIPS 197, November 2001, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

RESOURCES

D. Denning and G. Sacco, "Timestamps in Key Distribution Protocols," Communications of the ACM, vol. 24, no. 8, August 1981.

A. K. Lenstra and E. R. Verheul, "Selecting Cryptographic Key Sizes," Journal of Cryptology, November 1999.

A. Menezes, P. van Oorschot, and S. Vanstone, "Handbook of Applied Cryptography," CRC Press, Boca Raton, FL, 1996.

NIST, "DES Validated Implementations," http://csrc.nist.gov/cryptval/des/desval.html.

NIST Special Publications, "Recommendation for Block Cipher Modes of Operation," 800–38A, 2001.

B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, John Wiley & Sons, Hoboken, NJ, 1996.

C. E. Shannon, "Communication Theory of Secrecy Systems," Bell Systems Technical Journal, 1949. The material in this paper originally appeared in a confidential report entitled "A Mathematical Theory of Cryptography" (September 1945), which is now declassified.

SOURCE

PIC Microcontrollers
Microchip Technology
www.microchip.com







Hardware Synthesis Using VHDL (Part 1)

VHDL Implementation

Michael presents an XC3S200-based timing system that includes a date/time function, an alarm, and a stopwatch. He used VHDL to get the job done.

Learning something new is never easy. I remember sitting in a brightly lit room surrounded by slackened faces as a presenter droned on and on about VHSIC high-definition language (VHDL). The presenter was an expert in the field who had written a book on the subject. Although he was in demand as a lecturer, his in-depth presentation on VHDL left most of us bored and wondering how to apply what he was teaching.

I'll try to avoid putting you to sleep. My approach is simple: I'll use real-world examples. In this article I'll describe practical aspects of VHDL such as counters, decoders, ROM, state machines, multiplexers, latches, test benches, and much more using a sophisticated stopwatch application (see Figure 1). The application I'll introduce runs on a Xilinx Spartan-3 evaluation board (see Photo 1). Test benches use some constructs that can't be synthesized because they are required to produce a stimulus for testing. I'll warn you about anything that shouldn't be used in a real design. The code constructs are generic VHDL code, which you can use with any synthesis tool. I'll emphasize VHDL code rather than explain how to use a particular tool because most vendors offer extensive tutorials.

WHAT IS VHDL?

VHDL was developed by the U.S. Department of Defense as a common language for describing hardware designs at the system, component, and gate levels. Its main purpose was to enable the exchange of design data in a common format between different contractors.

VHDL is ideal for producing design descriptions that can be translated or synthesized into real hardware implementations. System-C proponents don't want this known, but VHDL is an excellent system modeling language that can describe hardware components at either a functional or gate level.

WHY VHDL?

VHDL is an IEEE standard (Std-1076) supported by all tool vendors. Professionals use it for describing complex ASICs, producing video-processing FPGAs, generating digital signal-processing applications, and much more. By learning VHDL, you will have acquired a skill that's in high demand.



Photo 1—*Take a look at the Xilinx Spartan-3 starter kit with the watch application running in Stopwatch mode. The seven-segment display is located in the lower right corner. The push buttons on the lower left control the stopwatch. An XC3S200 FPGA positioned above the display runs the watch application.*

VHDL is extremely readable with a syntactic structure that's similar to Visual Basic and Pascal. It has a powerful, expressive syntax that can define new data types and inherently support team-based programming. One company I know has dozens of VHDL designers working in numerous locations on one design.

Tom Cantrell recently wrote that the future is bright for FPGAs, which will play a large role in mainstream applications ("More Flash, Less Cash," *Circuit Cellar*, 178, May 2005). I agree with Tom, but I'll go further and predict that VHDL will become the premier technology used to define FPGA content either as output from design tools or with direct programming. In combination with VHDL, FPGAs provide a low-cost approach to defining complex hardware designs that were inconceivable only a few decades ago. Perhaps most importantly, using VHDL to define hardware is fun.

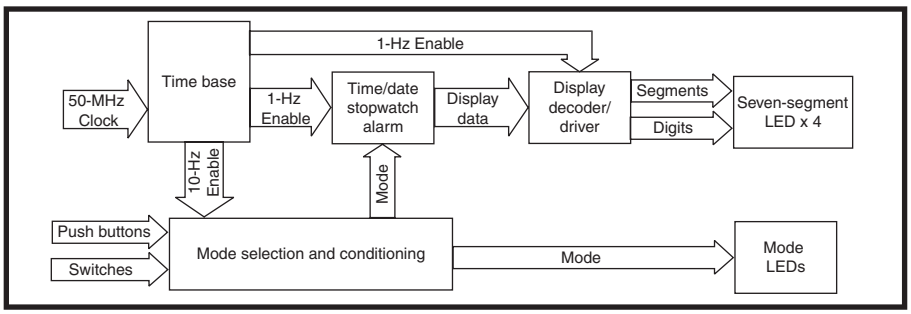


Figure 1—*The watch project consists of a time/date function, a second time zone, an alarm, and a stopwatch with lap timing. A time base generates ENABLE signals to run the timing functions, display the multiplexer, and debounce the circuitry.*

MY KIND OF TYPE

Data types are to VHDL what colors are to a painter. In this respect, VHDL has almost an infinite palette. Table 1 lists all predefined and standardized (Std-1164) data types along with brief descriptions. The only predefined data type I'll cover in this article is something called a natural number. This type represents an integer with values from zero to the largest representable integer (usually a 32-bit value). Later, I'll define some of my own types when I explain how to implement a state machine.

VHDL Type	Built in	Can be synthesized	Values
std_logic	IEEE	Yes	U, X, 0, 1, Z, W, L, H, –
std_logic_vector	IEEE	Yes	Array (natural range <>) of std_logic
boolean	Yes	Yes	True or false
integer	Yes	Yes	32 or 64 bits
natural	Yes	Yes	Integers ≥ 0
positive	Yes	Yes	Integers > 0
bit	Yes	Yes	0, 1
bit_vector(natural)	Yes	Yes	Array (natural range <>) of bits
character	Yes	No	7-bit ASCII
string(positive)	Yes	No	Array (natural range <>) of characters
text	Yes	No	File of string
time	Yes	No	hr, min, sec, ms, us, ns, ps, fs
delay_length	Yes	No	Time ≥ 0

Table 1—*Take a look at some of the data types defined in the Std-1164 and Std-1076 IEEE standards. The most commonly used are the std_logic, std_logic_vector, and natural types.*

Although VHDL includes a bit type with values of 0 and 1, this type usually isn't used for synthesis because it doesn't accurately describe the real-world environment. For example, how would you represent a high-impedance state with the bit type?

To address this problem, the IEEE introduced a new type called `std_logic`, which has values of `U` (uninitialized), `X` (unknown), `W` (weak X), `Z` (high impedance), `H/L` (weak 1/0), `-` (don't care), and, of course, `1` and `0`. The power of VHDL allows the definition of a data type to support any user-defined multi-valued logic using enumerations. Of course, to ensure VHDL portability, it's highly recommended to stick with the `std_logic` type as defined in the Std-1164 package.

An additional type from the IEEE Std-1164 package is called the `std_logic_vector`, which consists of a one-dimensional array of `std_logic` elements. The `std_logic_vector` is used to represent data buses, addresses, and any other aggregate of bits. Only extremely specific `std_logic` values are supported for synthesis purposes: `0`, `1`, `Z`, and `-`. The main reason for this limitation is that the FPGA devices typically contain only gates that respond to 0s and 1s and tristateable buffers using the `Z` value.

Later, I'll describe how to create a tristate buffer. I'll also explain how the `-` value is used when simplifying logic. Keep in mind that when creating test benches or device models, all

RabbitCores Do Just About Anything

Theatrical Acrobatics | Rocket Telemetry | Sports Broadcasting | Industrial Automation | Weather & Power Monitoring | Vehicle Tracking | Manufacturing | Security

Packed with features: The reason embedded engineers choose Rabbit Semiconductor for such a wide variety of real-world applications.

RabbitCore Features

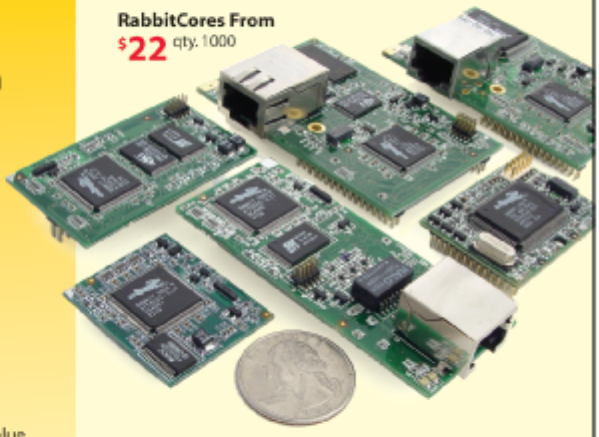
- Clock speeds to 51.8 MHz
- Up to 1 MB SRAM / 1 MB Flash
- Up to 54 digital I/O
- Up to 8 analog channels
- Up to 6 serial ports
- Ethernet available with royalty-free TCP/IP stack
- Wireless protocols available
- Complete application and development kits available
- Security and other software modules available
- Complete kits from $129

Get Your Development Kit Today!
Includes RabbitCore, prototyping board, and complete development software (IDE). For a limited time, get a free copy of *Embedded System Design Using the Rabbit 3000 Microprocessor* with purchase.

www.rabbitcoremodulekits.com

Free Book $49 Value

RabbitCores From $22 qty. 1000



RABBIT Semiconductor

2932 Spafford Street, Davis, CA 95616
Tel 530.757.8400

9841

of the values of the std_logic type can be used to simulate pull-up and pull-down resistors, undefined states, and logic contention.

OPERATORS

VHDL supports several overloaded operators: logical (and, or, nand, nor, xor, and not, which are applicable to Boolean or bit operations); shift (sll, srl, sra, rll, and rrl, which are only available in some libraries); relational (=, /=, <, <=, >, and >=), arithmetic (+, -, ×, /, mod, rem, **, and abs); and concatenate (&, e.g., two arrays are joined to make a single array).

Any package that introduces a new data type will also overload these operators to enable the manipulation of the new data type and perform conversions to and from existing types and the new data type.

PACKAGES & MORE

I've already cleverly sneaked in the term "package" without really introducing what it means. A package is an encapsulation mechanism used to group related types, constants, and functions or subprograms. Packages are frequently precompiled when the originator prefers not to disclose the source code, but they also can be provided in the form of source code.

The IEEE Std-1164 package is an example of a package supplied with complete source code. I won't describe how packages are created, but I'll refer to functions and types defined in some of the IEEE standard packages. In a real-world project, a package would typically contain the common type and constant declarations and functions required in multiple components. For the purposes of this article, however, the IEEE standard packages will suffice.

Each VHDL component consists of a single entity and one or more architectures. In hardware terms, an entity defines an IC's pins, while the architecture describes the IC's implementation or model. The simplest sort of architecture contains only instantiated components and the wiring interconnects that join the component pins to each other. More complex architectures define detailed functional elements such as counters, registers, state machines, etc.

GETTING STARTED

Listing 1 shows a VHDL implementation of a decade counter somewhat similar to the venerable 74LS90. The first line introduces the standard IEEE library. The next three lines (beginning with use) make visible specific packages contained in this standard: std_logic_1164, std_logic_arith, and std_logic_unsigned. The .ALL postfix indicates that all of the package's contents are made visible within this context. It's also possible to qualify the export of a specific package declaration using this same dot notation.

COUNTER ENTITY

The counter pins are declared in the entity section of the listing near the top. Five signals are defined: clk, enable, reset, q, and carry. Like pins on an IC, signals can have three basic modes (or directions): in, out, or inout.

in mode signals can be read only. out mode signals can be written only. inout signals can be read or written. buffer mode, a lesser-known mode that can be read or written to, is used primarily for carrying information out of a module. I don't recommend using buffer mode, however, because many synthesis tools have trouble with it.

With the exception of q, all other signals are single bits of the std_logic type. The q signal is your first encounter with a bus or std_logic_vector. Typically, bused signal vectors are defined with a bit ordering of n downto 0, which means that the leftmost bit n is also the most significant bit. The opposite bit ordering (from 0 to n) is also possible, but it isn't used too often. The q bus has a width of 4 bits.

COUNTER ARCHITECTURE

In the architecture block, the behavioral name distinguishes this architecture from the different implementations. Because my only focus is on producing real hardware, I won't cover multiple architectures.

The div10_ex architecture name must be the same name as the entity declaration name and tie this architecture to the entity. The architecture's body is contained in a begin/end block with the architecture's name echoed after the end.

Listing 1—*This typical VHDL decade counter/divider implementation features an asynchronous reset, a clock enable, and a carry output.*

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity div10_ex is
      port ( clk   : in  std_logic;
           enable  : in  std_logic;
           reset   : in  std_logic;
           q       : out std_logic_vector(3 downto 0);
           carry   : out std_logic);
end div10_ex;
architecture Behavioral of div10_ex is
  signal ctr : std_logic_vector(3 downto 0);
begin
  q <= ctr;
  carry <= ctr(3);
  process (clk, reset)
  begin
      if reset = '0' then
          ctr <= (others => '0');
      elsif clk'event and clk = '1' then
          if enable = '1' then
              if ctr = "1001" then
                  ctr <= (others => '0');
              else
                  ctr <= ctr + 1;
              end if;
          end if;
      end if;
  end process;
end Behavioral;
```

BitScope
Pocket Analyzer

USB Oscilloscope & Logic Analyzer

8 Channel 40MS/s Logic Analyzer
Capture digital signals down to 25nS with arbitrary trigger patterns.

3 Input 100MHz Analog DSO
Classic Analog Scope using a standard x1/x10 BNC probe. Additional inputs on the POD for dual channel operation.

8 + 1 Mixed Signal Scope
True MSO to capture an analog waveform time-synchronized with an 8 channel logic pattern triggered from any source.

Real-Time Spectrum Analyzer
See the spectrum and waveform of analog signals simultaneously and in real-time

Waveform Generator
Load up to 32K arbitrary waveform and replay via the onboard DAC (10MS/s) or a digital pattern from the POD (40MS/s)

The new generation Scope for the age of microelectronics.



Turn your PC or NoteBook into a powerful Scope and Logic Analyzer!

See inside your circuit in the analog and digital domains at the same time to make tracking down those elusive real-time bugs much easier.

Pocket Analyzer combines a high speed sample-synchronized storage scope and logic analyzer with a programmable waveform and logic pattern generator. Also included is an integrated real-time spectrum analyzer and powered "Smart POD" expansion interface so you've got all bases covered!

About the same size and weight as a Pocket PC, this USB powered BitScope needs no bulky accessories. It's the perfect low cost "go anywhere" test and debug solution.





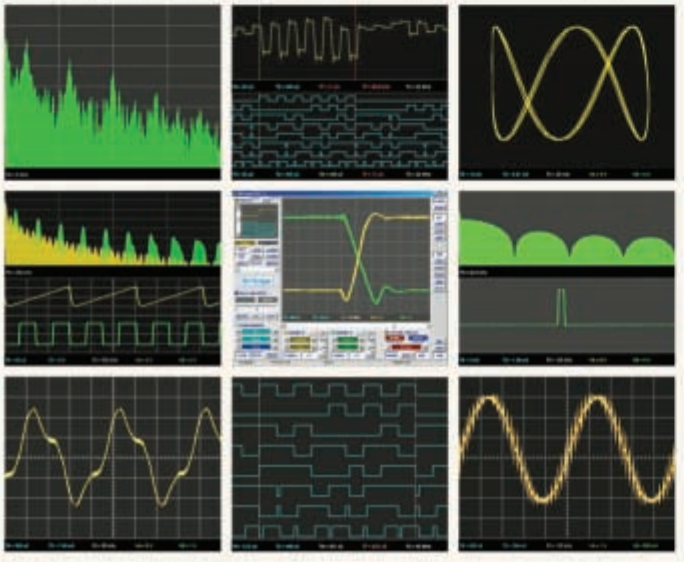
Standard 1M/20pF BNC Input
200uV-20V/div with x10 probe
S/W select AC/DC coupling
S/W select 50ohm termination
Arbitrary Waveform Generator

BitScope "Smart POD" Connector
8 logic channels, 2 analog channels
Dual channel capture from POD A/B
Async serial I/O for external control
Logic Pattern generator 32K 40MS/s

BUS Powered USB 2.0 Device
Single USB cable to your PC
Compressed data transmission
Simple ASCII control protocol
BitScope Scripting Language

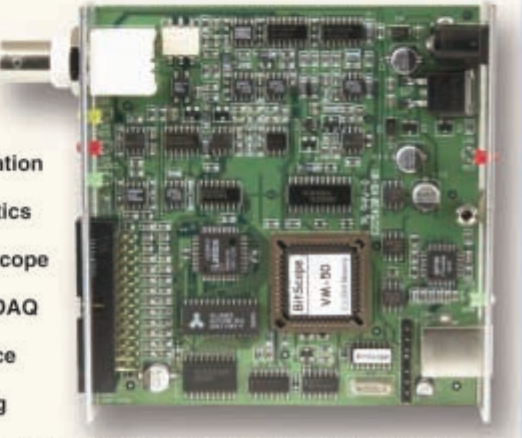
External/Passthru Power Supply
Auto senses an external supply - removes power load from USB for use with unpowered hubs. Supplies up to 500mA via POD

BitScope and your PC provide an array of Virtual Instruments



BitScope DSO 1.2 software for Windows and Linux

- R&D
- Education
- Robotics
- Lab Scope
- Fast DAQ
- Service
- Debug



BitScope Pocket Analyzer uses highly integrated Surface Mount technology to provide functionality you would expect from scopes many times the size and price. Its programmable Virtual Machine architecture means new functionality can be added via software. For custom Data Acquisition, export directly to your spreadsheet.

www.bitscope.com

Between `is` and `begin` is the declarative region where all constants, types, variables, signals, functions, and component declarations must be made. In this case only a single `ctr` signal is declared with a width of 4 bits. The `ctr` represents the internal decade counter state.

Two signal assignments appear outside the decade counter process just after the `begin` keyword of the architecture. In effect, each of these signal assignments is contained in a process of its own with a sensitivity list containing the signal on the right side of the signal assignment.

In terms of hardware, these signal assignments connect wires between the internal counter state and the external pins identified on the left side of the signal assignment. The reason you can't directly use the signal `q` as the state is because its port is defined as a write-only (or out) port. This is a common occurrence with VHDL `out` ports. In fact, some people attempt to cheat and define `out` ports as being `inout` or buffer. Don't do this. Let the data direction be your guide in defining the direction of ports. If the data is truly one-directional, use either the `in` or `out` port. Only bidirectional data flows (e.g., data buses) require `inout` ports.

DECADE COUNTER PROCESS

In the begin/end block is a process block. Processes are used to encapsulate a particular behavior of the design (the decade counter behavior in this case). The `clk` and `reset` arguments following the process name form the process's sensitivity list. This process is activated only when signals in the sensitivity list (`clk` or `reset`) change state. Sensitivity lists are used mainly in optimizing simulations so that process activation can be minimized to speed up the simulation. Synthesis tools monitor the sensitivity lists for completeness; they issue warnings if signals are missing, but they don't require them.

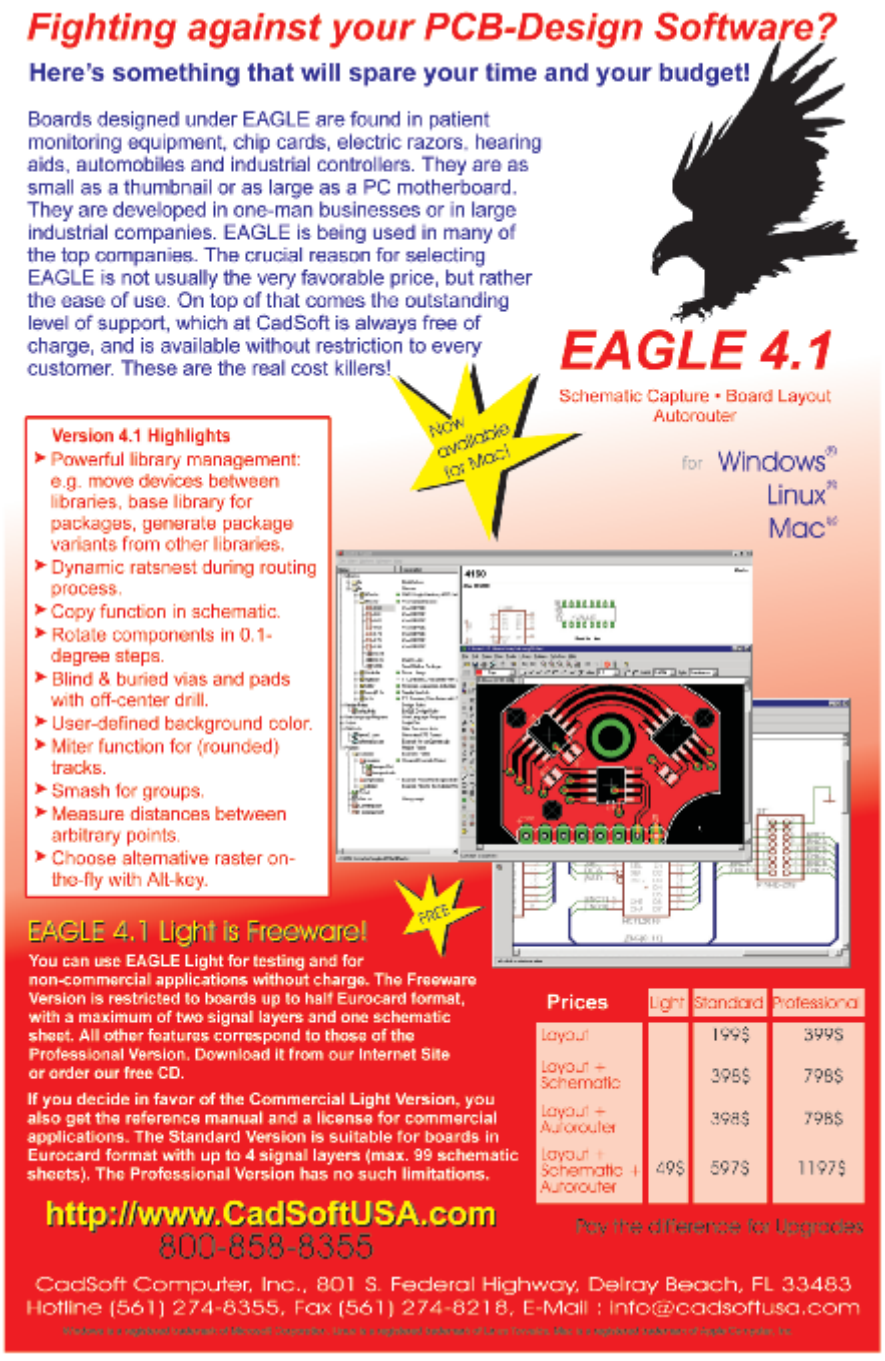
If you don't ensure that the sensitivity lists are complete, you may be surprised with some unusual simulation results. You haven't really used VHDL until this sort of simulation bug bites you.

Within a clocked process, you need to include the clock and all other dependent signals outside the clocked region of code in the sensitivity list. The clocked region in this example is demarcated by the `elsif` and `end if` statements.

The asynchronous part of the decade counter is after the first `if` conditional. The `reset = '0'` condition activates the statements following `then`, which, in this case, clears the `ctr` state.

The <= operator is a signal assignment. The signal assignment is different than the variable assignment. Signals are assigned concurrently; variables are assigned sequentially. This may sound like semantics, but it's actually an extremely important point to master. Signal concurrency is a reflection of the fact that all signal transitions in the real world may occur simultaneously. Consequently, all signals within a VHDL project change state simultaneously.

The strange-looking `(others => '0')` construct is a short-form technique for saying that all the array elements are assigned the 0 value. This



notation is a named association in which elements of an array are assigned by an index number or by the default name others.

You also could explicitly assign array elements with 0 => '0', 1 => '0', 2 => '0', and 3 => '0'. A significant advantage of the others notation is that the array initializer can be independent of an array's size.

Vectors can be initialized using a constant bit vector such as 0000. This approach reduces the amount of writing when initializing bit vectors. Extremely large bit vectors can be abbreviated even more by prefixing x for hexadecimal, o for octal, and b for binary so that 101010000001 can be written as x"A81" or o"5201". The default bit vector radix is assumed to be base two.

The clk'event following the elsif makes use of a signal attribute. Refer to the *Circuit Cellar* FTP site for a list of some common VHDL attributes supported by most synthesis tools. The event attribute triggers the following statements whenever the attached signal changes state. The clk = '1' condition means that the statements within the clock block, from then to end if, become active for every rising edge of the CLOCK signal. Synthesis tools look for this code sequence and produce clocked constructs in such cases. The std_logic_1164 package defines two functions, rising_edge and falling_edge, which duplicate the rising edge statements' functionality. I'll address these functions with examples.

Basic counter functionality is defined by the central if, else, and end if statements. The if enable = '1' then clock enable statement provides a gated clock implementation to apply clock pulses to the counter only when the ENABLE pin is at a logical 1. Then, the counter state is incremented by one (an implicit conversion takes place to convert the one to a std_logic_vector) as long as the counter isn't equal to 1001 or nine, which represents the final decade counter state. When this final count is reached, the counter will perform a synchronous reset on the next rising clock edge.

I know this may seem like a lot of work to get a decade counter, but it's all downhill from here. Armed with these basic concepts, you can design just about any counter or divider.

Figure 2 (page 44) shows the synthesized FPGA structure that results from this VHDL code. The primitives are based on those available in the Xilinx Spartan-3. As you can see, another benefit of VHDL is that you can express a design in a generic language that has no ties to a particular piece of hardware. You can resynthesize this same VHDL code for an Actel FPGA or any other vendor's device, including ASICs! Figure 3 (page 44) is a sample simulated output for approximately 10 clock cycles.

GENERIC ARGUMENTS

Generic arguments are powerful mechanisms available in VHDL for defining general-purpose entities that you can tailor to suit different requirements. For example, Listing 2 contains a generic counter implementation that's easily customized by passing a couple of arguments to transform it into practically every different counter or divider you'll need for my watch project!

Two generic arguments, width and max_cnt, which are located at the top of the entity declaration, are the keys to this flexibility. The former represents the counter's bit width. The default width is specified as eight. The max_cnt argument specifies the maximum count value that the counter can reach. The default max_cnt is 255. Default arguments enable you to create an instance without having to specify the generic arguments. As long as you're happy with the default generic arguments, neither the synthesis tool nor the simulator will complain.

The first place you'll see a change from the original decade counter is in the definition of the q signal. In the first decade counter implementation, I hard-coded an upper limit of three. Here in the port declaration, q's upper limit is set to the value of width-1. Thus, if you were to instantiate this counter with a width of 12, q's range would be 11 down to 0, thus creating a 12-bit counter width.

Listing 2—*The general-purpose synchronous VHDL counter/divider implementation has generic parameters to specify the counter bit width and maximum count value. An asynchronous reset, a clock enable, and a carry output complete the design.*

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity divn is
   generic          (width   : integer := 8;
                     max_cnt  : integer := 255);
   port             (clk     : in std_logic;
                     reset    : in std_logic;
                     enable   : in std_logic;
                     q        : out std_logic_vector(width-1 downto 0);
                     carry    : out std_logic);
end divn;
architecture behav of divn is
   signal cnt : natural range 0 to max_cnt;
begin
   carry <= enable when cnt = max_cnt else '0';
   q <= conv_std_logic_vector(cnt, width);
   counter: process (clk, reset)
   begin
       if reset = '0' then
            cnt <= 0;
       elsif rising_edge(clk) then
            if enable = '1' then
                if cnt = max_cnt then
                    cnt <= 0;
                else
                    cnt <= cnt + 1;
                end if;
            end if;
       end if;
   end process;
end behav;
```

Another change relates to the data type used for the counter. The original decade counter used an `std_logic_vector` type. For the generic counter, however, I used a subtype of the natural number type for the `cnt` type. The subtype is declared implicitly in the cnt signal declaration as `natural range 0 to max_cnt`. This means that the cnt signal can take on any value from zero to `max_cnt`.

Why did I make this change? It certainly could have used an `std_logic_vector` as a generic argument, but I prefer working in base 10, so it made sense to use a natural number counter. There isn't a hardware penalty for this choice. Plus it makes the design more readable. As a consequence, you need a conversion function (`conv_std_logic_vector`) to convert from the natural number cnt to the `std_logic_vector` result for q. The conversion takes place as part of the signal assignment where the cnt state is wired to the q output. The first argument is the natural or integer signal to be converted to a `std_logic_vector`. The second argument is the desired width of the resultant `std_logic_vector`. This conversion function is built into the IEEE std_logic_arith package.

The remaining code should look extremely similar to the decade counter (with the exception of using natural numbers in places where vector constants were previously required for initialization and maximum count comparison). Another change is the use of the `rising_edge` function to define the clock edge used for event transitions. This makes the code more easily understandable.

The final change has to do with the way you produce the carry output signal. In the decade counter, you simply use the most significant bit as the carry signal. But when creating a chain of synchronous counters, you might sometimes create counters in which the upper bit remains high for longer than one clock cycle. Thus, you could get two or more clocks carrying through to the next counter in the chain.

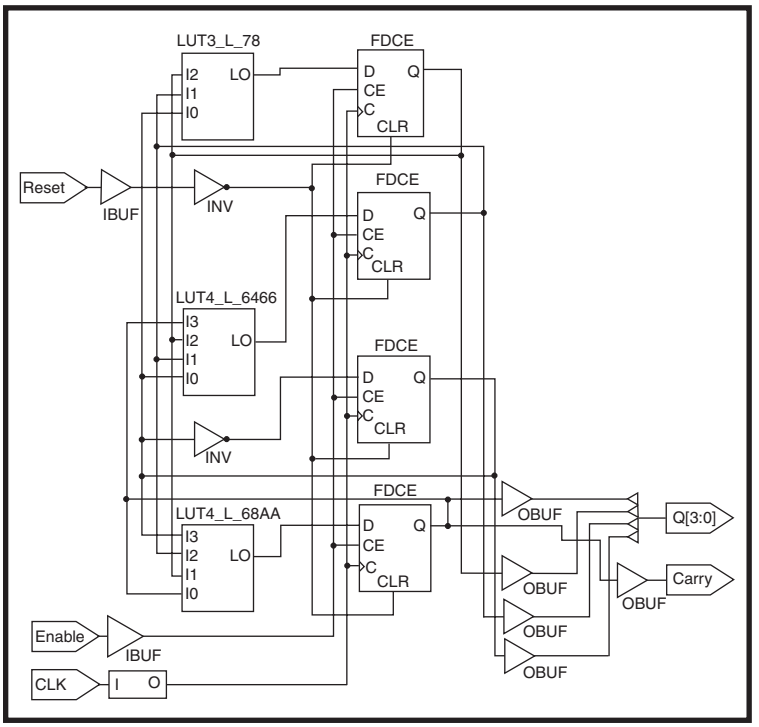


Figure 2—*The Spartan-3 primitive blocks comprise the decade counter. The LUT3 and LUT4 blocks are three- and four-input look-up tables. The FDCE blocks are D flip-flops with clear and enable inputs.*

Because you'll want only one carry clock generated each time the counter reaches its maximum value, that's what I encoded in the signal assignment for the carry output. Note that this is a conditional signal assignment using the `when` conditional statement. The ENABLE signal is passed through to the `carry` whenever the `cnt = max_cnt` condition is satisfied; otherwise, in the `else` clause, a logical 0 value is assigned to the `carry`.

FIRST TEST BENCH

You must create a test bench to test whether or not the generic counter actually works. A test bench is analogous to a hardware bench. It contains instruments such as signal generators, oscilloscopes, and logic analyzers, which are connected to the circuit being tested. The VHDL equivalents of these instruments are shown in Listing 3.

A test bench is nothing more than an entity/architecture pair with the circuit being tested instantiated in the declarative region of the architecture. You can see the generic `divn` component declaration following the `is` of the architecture. Signal declarations for each of the `divn` port elements follow the component. The actual instantiation is found in the architecture body beside the `uut:` label. Each signal passed or received from the `divn` port uses a named association with the port signal name on the left and the local signal on the right of the => symbol.

The primary stimulus required is the clock, which is defined in the process block labeled "clock." Initially, `clk` is set to a logical 0 value.

The following `wait` statement pauses the process for 500 ns. The `wait` statement should never appear in any code that can be synthesized. After the first `wait`, `clk` is set to a logical 1 value and the process waits another 500 ns. After the final `wait` time has elapsed, the process begins executing from the

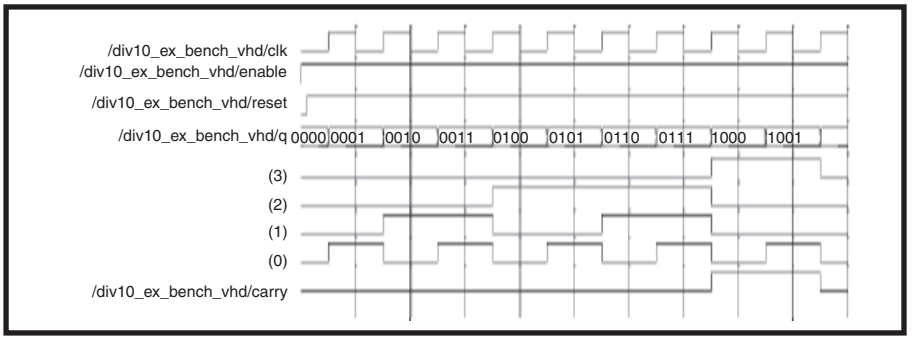


Figure 3—*The top clock waveform drives the simulated decade counter. The q output is shown below in both a vector format and broken down into each of the four counter bits.*

initial statement following the begin statement. In effect, this defined a clock with a 50% duty cycle and a 1-µs period.

The second process, which is labeled tb, provides the required reset stimulus. Like clk, the reset receives an initial value of logical 0. After 100 ns, it's set to logical 1. The ENABLE signal is tied permanently to a logical 1 for the purposes of this test. The final wait statement with no arguments pauses the second process indefinitely. This is an effective method of executing the contents of a process only once. Remember, wait statements can't be synthesized and therefore can't appear in any VHDL intended to be rendered as a hardware component.

As you can see, by defining one or more processes and using wait statements, it should be possible to synthesize any stimulus required for driving a test component's inputs. More elaborate test benches can also include functional component models of elements such as memories, processors, clocks, and data acquisition systems provided by many IC vendors. The simulator running the test bench acts as a logic analyzer and provides access to all the tested model's inputs and outputs.

Listing 3—*The small test bench generates CLOCK and RESET signals to drive the divn counter implementation. The test bench instantiates a decade counter from the divn generic counter.*

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;
entity divn_bench_vhd is
end divn_bench_vhd;
architecture behavior of divn_bench_vhd is
--Component declaration for the unit under test (UUT)
   component divn
      generic  (width    : integer;
                max_cnt  : integer);
      port     (clk      : in std_logic;
                reset    : in std_logic;
                enable   : in std_logic;
                q        : out std_logic_vector(width-1 downto 0);
                carry    : out std_logic);
   end component;
--Inputs
   signal clk :  std_logic := '0';
   signal reset :  std_logic := '0';
   signal enable :  std_logic := '0';
--Outputs
   signal q :  std_logic_vector(3 downto 0);
   signal carry :  std_logic;
begin
--Instantiate the unit under test (UUT)
   uut: divn
      generic  map       (width => 4,
                          max_cnt => 9)
      port map           (clk => clk,
                reset => reset,
                          enable => enable,
                          q => q,
                          carry => carry);
   clock : process
   begin
      clk <= '0';
      wait for 500 ns;
      clk <= '1';
      wait for 500 ns;
   end process;
   tb : process
   begin
--Wait 100 ns for global reset to finish
      reset <= '0';
      enable <= '1';
      wait for 100 ns;
      reset <= '1';
      wait; -- will wait forever
   end process;
end;
```

START EXPERIMENTING

Thus far, I've covered a lot of VHDL territory: data types, attributes, entities, architectures, counters/dividers, generic components, test benches, instantiated components, and a variety of statements. Next month I'll investigate state machines, encoders/decoders, multiplexers, and ROMs. I'll show you how these elements can be combined to make the stopwatch application in Figure 1. Until then, download some VHDL tools and have fun experimenting with the examples in this article.

The complete VHDL source code is available on the *Circuit Cellar* FTP site. If you're using the Xilinx tool set, I also included a project that contains all the same source code.

Michael Griebling is a professional electrical engineer with a B.A.Sc. in electrical engineering. He has been active in the aerospace industry for more than 20 years. Michael enjoys designing firmware, hardware, and software. You may reach him at mg@comp-inspirations.com.

PROJECT FILES

To download the code, go to ftp.circuitcellar.com/pub/Circuit_Cellar/2005/181.

RESOURCES

Std-1076-2002, *IEEE Standard Language Reference Manual*, IEEE, Piscataway, NJ, 2002.

Std-1164-1993, *IEEE Standard Multivalue Logic System for VHDL Model Interoperability*, IEEE, Piscataway, NJ, 1993.

Xilinx, *Spartan-3 FPGA Family: Complete Data Sheet*, DS099, January 2005.

——— *Spartan-3 Starter Kit Board User Guide*, UG130, Xilinx, October 2004.

SOURCES

ISE WebPACK 7.1i and XC3S200 Spartan-3 FPGA starter kit
Xilinx
www.xilinx.com



by Jeff Bachiochi

Simulate First

Simulation Techniques for Testing New MCUs

Simulators are designed to help you debug your code. But as Jeff will show you, you can also use a simulator to get a feel for a new microcontroller and its peripherals. The idea is to play around with an MCU before spending money on it.

Newbies are always asking me, "How can I get started with microcontrollers? I don't have a lot of money, but I have plenty of time to invest." A good place to start is your local bookstore. Not a buck-a-book shop, but a store that has a good selection of technical documentation (preferably a store that carries *Circuit Cellar*). Although most books written today are how-to manuals for high-level computer languages, probe around until you find the section of the store that pertains to hardware, the shelf that deals with processors, and the few books that cover microcontrollers.

An alternate approach is to play. Yup, play around. Often it's free. Have you ever found yourself using a screwdriver on a new toy prior to even turning it on? I have. I always want to know more. This often means playing around with a new toy. If your new toy is a simulator, you can learn about microcontrollers just by playing with it.

A simulator, which is an application running on your PC in this case, enables you to execute an MCU's code without actually having to have the hardware in front of you. Your giga-powered PC is made to look like a microcontroller.

Most microcontrollers are embedded as part of a larger device and contain a CPU, memory, and I/O. Think of a microcontroller as nothing more than a set of registers or storage containers manipulated by program instructions. The registers may be used as program memory to hold code instructions or data memory to hold temporary values. They may be part of the CPU, so you can use them to manipulate data, or you can directly interface them with another function (e.g., real-world I/O or counting, timing, and communication). You'll need to review the documentation that describes the functions in order to understand how each is implemented in the device.

This illustrates why you must spend some time choosing a particular part based on the functions required for your application. Most manufacturers have a product selector guide that lists the functions included with each part. But you don't need to get involved with that just to get a feel for how to write and simulate some simple code.

SPIDER-MAN

If you don't know where to start, search the 'Net. If you're already familiar with an MCU manufacturer, visit its web site and search for development tools. Many large companies are too busy to develop their own tools just to give them away, which is a bit shortsighted if you ask me. However, you can't blame them for believing they can best serve the design community by putting all of their effort into their hardware. This is especially true because there are so many good software companies out there creating support software.

For instant gratification, check out the companies listed at the end of this article. These companies have made it their business to offer free tools for some of their products. Focus your attention on the assemblers and simulators, which aren't necessarily just for you non-swimmers out there. I still write most of my applications in assembler and verify my routines with a simulator.

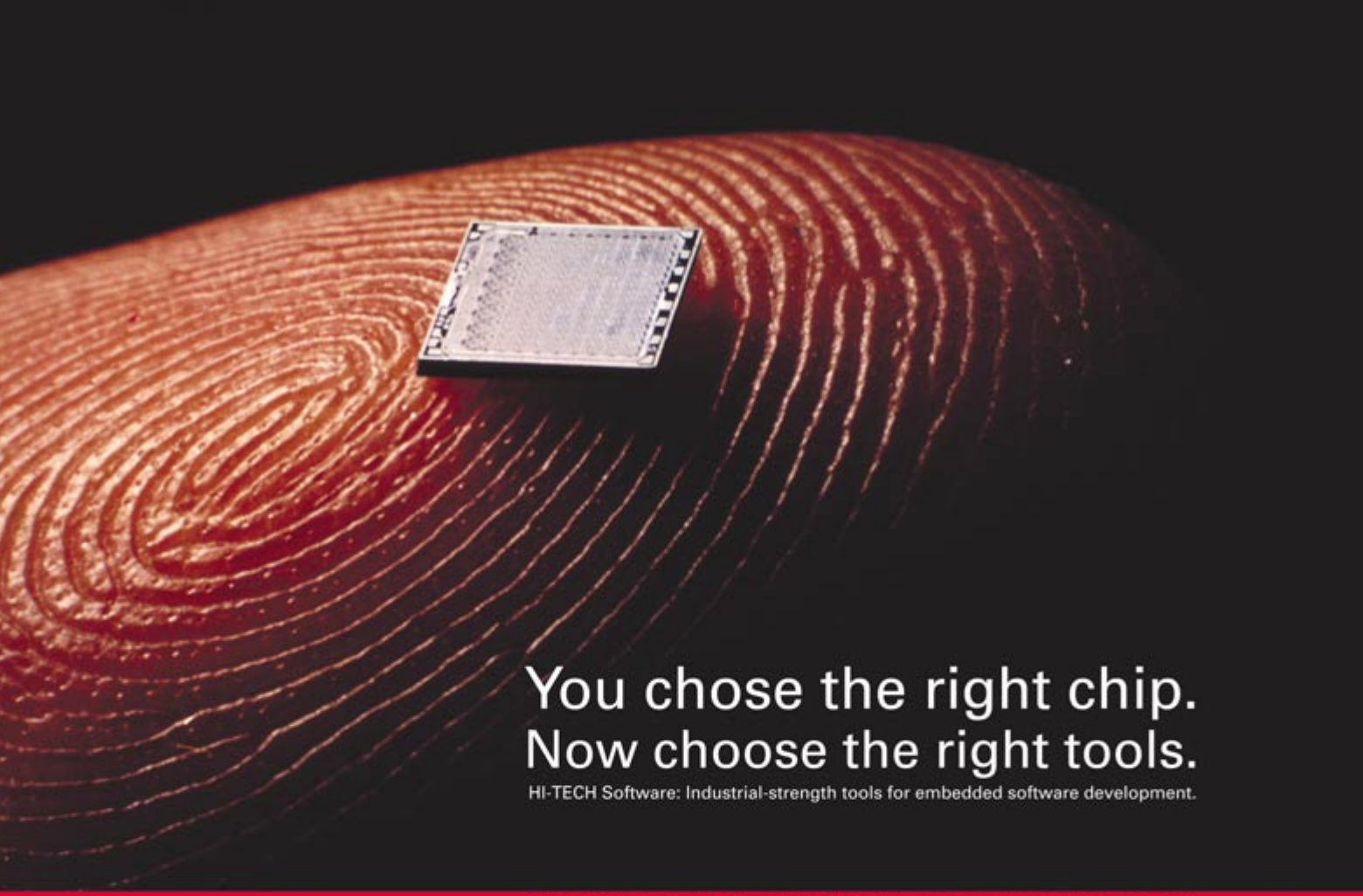
Many tools come as part of an integrated development environment (IDE), which enables access to each tool from a familiar framework that keeps track of where the tool's components and the files you've create are kept. It's important to read the IDE's documentation in order to familiarize yourself with the process of creating and simulating code execution for one of the supported MCUs.

JUMP IN

I used Microchip Technology's MPLAB IDE for this project. Why? It isn't that Microchip is better than the other manufacturers at writing an IDE application, but this IDE supports the entire line of Microchip parts. This means you aren't limited to using this IDE with only select Microchip parts.

Downloading and installing MPLAB should take less than 1 h. At that point, you'll start with a clean slate (if you haven't run the application before). Click Project Wizard on the Project pull-down menu. The Wizard will lead you through the steps of creating a new project. The program will set up the IDE with all of the information it will need to work with the code you'll write.

I used a PIC16F873A microcontroller for this project because it has a good selection of commonly used functions. Choose the default tool set: MPASM, MPLINK, and MPLIB.



You chose the right chip. Now choose the right tools.

HI-TECH Software: Industrial-strength tools for embedded software development.

Compact, efficient code for ARM, PIC and 8051 processors

Whatever processor family you are targeting, whether it is the ARM, PIC, 8051 or one of the other chip families we support, HI-TECH tools can help you write better code and bring it to market faster. Our reputation is proven, our tools are first rate, and our support staff is the best in the industry.

HI-TECH ARM-C:
The HI-TECH ARM-C is a high-performance C compiler for the powerful RISC microcontrollers based on the ARM architecture, delivering excellent code density and reliability.

HI-TECH PICC:
HI-TECH PICC is the overwhelming choice for developers on the Microchip PICMicro family - supporting every one of the extensive range of PICMicro devices.

HI-TECH 8051C:
The 8051 family is the most successful 8-bit microcontroller family in the industry, available from over 40 manufacturers in a huge range of chips. HI-TECH 8051 C is a field-proven ANSI C compiler supporting over 800 different microcontrollers.

HI-TECH C is also available for several other popular embedded microcontrollers.

You thought long and hard about which chip to use for your next project. It's a crucial decision - choose the right architecture and you just might create a product that changes the world. Choose the wrong one and your project may be off to a slow start. The next critical decision is what tools to use. You know you want compact, efficient code. You know you need reliable and dependable tools from a company with a stellar reputation and first-rate technical support. And you know that it must be a company that has proven that customers can rely on their tools to succeed.

For over two decades HI-TECH Software has delivered the industry's most reliable tools for developing efficient and compact code to run on the most popular embedded processors. Used by tens of thousands of customers including General Motors, Whirlpool, Qualcomm, John Deere and many others, HI-TECH's development tools and world-class support staff have helped serious programmers to create hundreds of successful embedded products.

Find out more or download a demo at http://cc.htsoft.com/

HI-TECH Software LLC 6600 Silacci Way Gilroy, CA 95020 USA
Ph: 800 735 5715 **Web:** http://cc.htsoft.com/

HI-TECH SOFTWARE

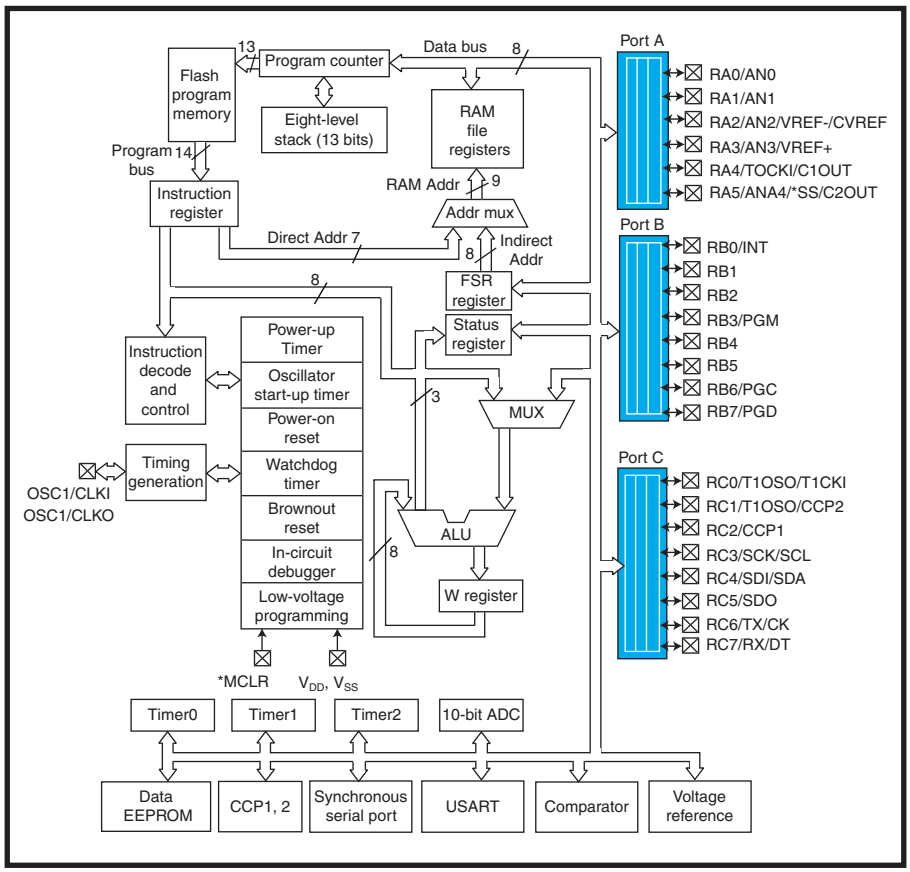


Figure 1—*Note how the internals of the PIC16F873A relate to one another (usually including I/O and peripherals). The PIC16F873A has 4 KB of program flash memory, 192 bytes of data memory, and 128 bytes of data EEPROM.*

(MPASM is Microchip's assembler that turned my code into the instructions executed by the microcontroller.) Next, define the project name and folder where it's kept.

I usually add a project folder to the installed MPLAB IDE folder and then add my stuff to it. This keeps everything in one area. Because I hadn't written any code yet at that point, I clicked past the following screen, which is for adding (prewritten) source code to the project.

After I had created a project, MPLAB was happy, but I hadn't really done anything. One of the most difficult things when getting started is actually getting started. A code template is a good place to begin.

Click Open in the File pull-down menu, navigate to the MPASM Suite\Template\Code folder, and then choose the file corresponding to your MCU (f873atemp.asm). A file will then load to your workspace. Click Save As, rename the file, and save it to your project folder. (FTB181 was my file name and the name of the project folder.)

Following this, add the file to the project. In the upper left corner is a window (FTB181.mcp) showing a list of potential file types. Right-click on SourceFiles and added FTB181.asm to the project. Next, click Make in the Project pull-down menu. MPLAB will save the project file and use MPASM to assemble the file and display any errors it found. Doesn't that sound easy?

QuickPIC

What are basic things you need to know about the PIC16F873A? First you must know what's inside it (see Figure 1). There are three main parts. On the right are ports A, B, and C, which are the direct I/O. On the bottom are a number of peripheral function blocks. All of the registers are accessible to the data bus. The flash program memory is in the top left section. The user RAM is in the top right section. Most of the remaining blocks make up the CPU, whose program counter retrieves instructions from the program memory and places them into the instruction register.

The instruction decoder determines which operation is to be performed based on the instruction. Although most instructions are executed in a single cycle, there's actually a two-cycle pipeline. During the first cycle, an instruction is fetched; it's executed during the next cycle while the next instruction is fetched.

The program memory is 14 bits wide and divided into 2,000-word pages, as shown in Figure 2. Code execution begins at the reset vector (0x0000). Interrupts are directed to the interrupt vector (0x0004). The 14-bit instruction size limits how much memory can be directly accessed within a single instruction (11 bits), thus the 2,000-word page size. For MCUs with more than one page, other pages are accessed by preloading the PCLATH register.

Every register accessible within the MCU has a defined location in RAM, which, like program memory, is also divided. A bank is 128 bytes of RAM. As you can see in Figure 3 (page 51), general-purpose user RAM is located at the high end of each bank (starting at address 0x20 of bank 0). You can select a bank with the register select bits in the STATUS register.

QuickASM

What's an assembler? The CPU

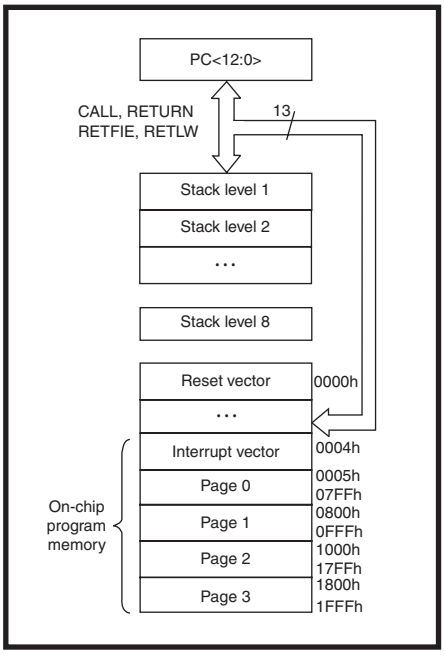


Figure 2—*Every microcontroller is a self-contained computer including ROM, EEPROM, or now flash memory-based code memory space.*



Everywhere you imagine. RENESAS

MCU Solutions Matched To Your Design Imagination

RENESAS IS THE #1 MCU SUPPLIER IN THE WORLD. **www.renesas.com**

M16C Advantages

- Advanced 16-bit and 32-bit CISC cores that perform at up to 32 Dhrystone 1.1 MIPS
- Banked-register architecture optimized for fast context switching
- Wide range of high-performance flash memory: 8 KB to 512 KB
- Best-in-class EMI/EMS noise immunity
- Designed to be pin- and function-compatible between families and packages
- Extensive peripheral integration: everything from POR and LVD to CRC counters

Low-cost Renesas Toolchain

- Fully integrated suite of hardware and software tools
- Highly-optimized C compiler and assembler — FREE (64K limited version): Download latest version from web (www.renesas.com/skp)
- Low-cost Starter Kit (target board, USB debugger, complete toolchain), starting at $49
- Easy online access to application notes, example code, software development tools

Get your product to market faster using Renesas Interactive!

Renesas INTERACTIVE

Evaluate, test, and learn about Renesas products in an interactive online environment.

www.renesasinteractive.com

SKP	Device	C Compiler, Assembler, IDE	Cost
SKP8CMINI13	R5F21134FP	FREE (64K limited version)	$49
SKP8CMINI17	R5F21174FP		$49
SKP16C26A	M30260F8AGP		$49
SKP16C28	M30280FAHP		$49
SKP16C62P	M30626FHPFP		$49
SKP32C83	M30835FJGP		$119
SKP32C84	M30845FJGP		$49

Special pricing is in US dollars from Renesas authorized North American distributors only, and is subject to change.

Popular Devices

Series/ Group	Part Number	CPU (Bits)	Package	Freq. (MHz)	Dhrystone 1.1 MIPS (Max)	Program Flash (KB)	Data Flash (KB) (10K Rewrites)	SRAM (Bytes)	Serial I/O Channels	I²C	I/O	A/D 10-bit	Ring Osc.	LVD POR	Timers	Price *
R8C/17	R5F21172SP	16	20 Pin SSOP	20	9	8	2	512	1	1	15	4	X	X	3+WDT	$1.90*
R8C/13	R5F21134FP	16	32 Pin LQFP	20	9	16	4	1K	2	–	24	12	X	X	4+WDT	$2.38*
M16C/26A	M30260F8AGP	16	48 Pin LQFP	20	13	64	4	2K	3	1	39	12	X	X	8+WDT	$4.76*
M16C/28	M30281FAHP	16	64 Pin LQFP	20	13	96	4	8K	4	1	55	13	X	X	9+WDT	$5.90*
M16C/62P	M30624FGPGP	16	100 Pin QFP	24	16	256	4	20K	5	3	87	26	X	X	11+WDT	$8.25*
M16C/62P	M30626FHPGP	16	100 Pin QFP	24	16	384	4	31K	5	3	87	26	X	X	11+WDT	$10.05*
M32C/84	M30845FJGP	16	144 Pin LQFP	32	32	512	–	24K	5	5	124	34	X	X	11+WDT	$16.88*

* Suggested 10,000 piece price. Please contact your local Renesas distributor for volume pricing.

Saelig brings you easy-to-use cutting edge products!

Tinytalk Data Logger



Tinytalk - Low cost, low unit weight, and 1800 readings. Not for harsh environments.

TK-0014 **from $100!**

USB Cables



USB cables cost an arm and a leg in stores - here they are much cheaper but just as good! USB A/B cable choose from 3', 6', or 10' long.

USB 3-foot A/B Cable **from $6!**

I/O thru Ethernet



BITlink - easily construct control systems communicating through Intranet/Internet. BIT2000 for process control, building monitoring, data logging, alarm systems and other industrial uses.

BIT2000 **from $399!**

16-Ch Data Logger



DL 7000 - New SBC for embedded data logging. 16 analog inputs with adjustable input ranges; 6 digital I/O lines, 7" X 3" X 2" - low cost, small size, low power, and high reliability.

PicoScope 3206



PS3206 - is an impressive 200 MS/s 8-bit PC scope adapter - 10 GS/s for repetitive signals! No power supply needed, simple USB 2.0 connection.

PP263 **only $1519**

USB Temp Logging



DLP-TH1- low cost, self-powered USB-based digital temperature/humidity sensor. Easily monitor temp/humidity /dew.pt. from PC's USB port.

DLP-TH1 **from $99.95!**

Data Logger



VL100 - 2" x 3" battery-powered analog & digital loggers store events, volts, current, pressures for weeks. Download to PC and review graphic results or Excel spreadsheet.

VL100 **only $190!**

NEW! USB ADC 11



USB ADC-11 - USB-connected/powered 11 ch voltage data logger - fast sampling rates and with digital outputs for control or alarm.

PP239 **only $180.50!**

MP3 on Demand



DAD10 - MP3 message/music-on-hold for PABX or standalone audio delivery. Stereo output is 2W into 4 ohms, but consumes only 70mA from an 8-15V supply with no speaker load.

DAD10 **only $199!**

Auto Kit



Automotive Kit - automotive Diagnostic kit for car electronics performance fault-finding. Measure and test virtually all auto electrical components!

PP205 **from $1139!**

Tinytag Transit



Tinytag Transit - A range of low cost temperature data loggers specially designed for monitoring conditions during the shipment of small packages. Replaceable battery.

TG-0050 **from $60**

16 Analog I/O



PCI230 high-speed multifunction PCIboard - 16 analog inputs, 2 analog outputs, counter-timers, etc. Plug'n'play PCI 2.1. Screw terminal for easy connection.

90989383 **from $531!**

NEW! Hercules Mobile



Hercules Mobile - Portable PC for Healthcare Applications battery-powered, in a completely sealed, IP65 washable aluminum housing. 10.4" LCD touch screen.

96050074 **from $5236.55!**

Wireless Control



WMB-USA - Wireless controller board uses 16 MHz Atmega 8535-16 for 16 MIPS. Built-in serial RF link for low cost wireless control, mobile robots and data logging applications.

WMB-USA **only $89!**

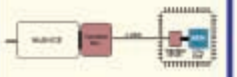
Internet Logging



SWI-300 - Remote data logger - enables non-technical users to install, setup and operate a remote data logging and alarm systems.

SWI 300 **only $559.95!**

JTAG in 1 Pin!



J-Link - Traditional JTAG testing uses 5 valuable i.c. pins. Revolutionary new J-LINK system requires only one pin and two resistors, and reduces chip consumption.

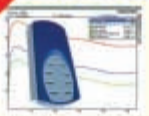
GO/NOGO Module



FloPSI - Programmable GO/NOGO indicator module compares input voltage to a defined threshold window. LED shows green/GO; red/NOGO.

FloPSI 1010 **from $13.95!**

NEW! USB TC-08



USB TC-08 - USB connected and powered 8-ch thermocouple data logger. Samples at up to 10 rdgs/S with built-in CJC for -270 to 1820 degC.

PP222 **only $474!**

Low Cost SBC



SBC4010 - Excellent value SBC compact control board. Powerful multitasking processor, in-field FLASH programmable. 1 x RS232, 1X RS485.

SBC4010 **from $126!**

Video Motion I.C.



VMD-19 - video motion detector module - automatically detect motion of objects within a video signal! For security cameras, VCRs, video switchers, quad TV processors, etc.

VMD-19 **only $32.79!**

Tinytag Encapsulated



Tinytag Encapsulated - Potted data loggers enable temperature recording in harsh conditions. Single channel temperature recorders.

TG-3100 **only $115!**

Tinytag Ultra



Tinytag Ultra - A range of low cost temperature and humidity data loggers designed for monitoring indoor environmental conditions. Custom design.

TGU-0017 **only $115!**

NEW! Ether-IO



Ether-IO - UDP/IP-controlled 24 X digital I/O board. 3 x 8-bit TTL ports each independently programmable. Connects to any TCP/IP Ethernet network.

Ether-IO 24 **from $99!**

Tinytag Extra



Tinytag Extra - Low cost temperature and humidity data loggers housed in robust, waterproof cases. New waterproof cases.

TGX-3080 **from $130!**

Intrinsically Safe Logger



Intrinsically Safe - *NEW!* A range of temperature and humidity recorders with ATEX accreditation. Sing and dual channel temperature and humidity recorders.

TGIS-0017 **from $255!**

NEW! Ventrix 25FSS



Ventrix 25FSS - new 12V-powered fanless 1GHz VIA Eden PC system with no moving parts - ideal for automobile applications. 2GB CF storage 10.8" x 7.2" x2" aluminum housing.

97040789 **from $1044!**

Motion Detector



Watchit - turns VCR into an intelligent video motion recorder. Add to existing security system for auto-recording movements (intruders!) on any VCR

VMD19-M **only $155.79!**

NEW! AudioWave



AudioWave - Software turns your Soundcard into a Sweep Generator! Easy-to-use digitally-controlled LF-signal-generator, 1Hz-20kHz. Great for service calls!

AudioWave **from $59.95!**

NEW! FrontDesign



FrontDesign - software for designing professional-looking front panels - with a host of drawing functions. Print, mill or engrave, or export final at 600 dpi.

FrontDesign **from $79.95!**

NEW! Digital ProfiLAB



Digital ProfiLAB - Easy-use Logic Sim/Hardware Control software - without programming! Circuit, logic and front-panel simulation, and hardware control, experiments.

Digital-ProfiLAB **from $79.95!**

NEW! sPLAN



sPLAN - easy-to-use CAD package for quickly and neatly drawing electronic and electric circuit diagrams. Beginners can create a perfect-looking diagram quickly!

sPLAN **from $79.95!**

First time customers

$10.00 off your first order

Use offer CP103

On orders of $40 or more

(Not available with any other offers)

"I received the USB kit today, and 5 min later I was up and running with my USB test program. That was easy!" (J.C.)

Saelig Co. Inc.

p: 585-385-1750 f: 585-385-1768

www.saelig.com/ad/cc805.htm • info@saelig.com

ALSO: SCALABLE LED DISPLAY PANELS, TEMP-HUMIDITY MONITORS, THERMOCOUPLE P.C. ADAPTERS, ENVIRONMENT MONITORING SYSTEMS, EDUCATIONAL SCIENCE PROJECTS, GRAPHICS SOFTWARE, AutoCAD PROGAMMING COURSE, USB-PIC BOARDS, FLASH PROGRAMMERS - IF YOU DON'T SEE WHAT YOU NEED MAYBE WE CAN FIND IT FOR YOU? - ASK FOR SALES!

grabs an instruction to execute by reading a memory location and transferring the data into the instruction register. The width of the instruction register is an indication of the number of potential different instructions a CPU can recognize and act on. Each value has the potential of a different operation, although not all values are necessarily used. Some instructions perform specific tasks without additional information like the sleep instruction. Not all MCUs that have a sleep function use the same value for the instruction. In fact, the same manufacturer may not use the same values for all of its processors. Microchip manufactures MCUs with 12-, 14-, and 16-bit instructions using values of 0x003, 0x0063, and 0x0003, respectively, for the sleep instruction.

If your program consists of only the sleep instruction, you'll need to use a value of 0x0003 or 0x0063 (depending on the MCU) and program your MCU with the value at the reset vector location 0x000. The first computers were programmed this way (i.e., by figuring out the proper value for each instruction by hand).

Assembly language involves a shorthand notation of sorts to describe an instruction. Most instructions are a mnemonic (an opcode with optional operands). The mnemonic sleep instruction doesn't require additional operands. If you use mnemonics to describe a set of instructions to execute, an assembler will translate them into values based on the type of MCU you're using. This way the same code can be assembled for different MCUs, creating a different hex file.

The hex file is a list of values that corresponds to the instructions you wrote using assembly language mnemonics. This is what will be programmed in your microcontroller's memory. Today's flash memory can be rewritten over and over. ROM or one-time-programmable (OTP) MCUs have to be thrown away if a change in code is needed.

Some instructions require additional information along with the mnemonic: `CLRF,PORTA`. This instruction clears a register. The register cleared is indicated by an operand that follows the mnemonic. Here the assembler needs to know the MCU being used as well as where the port A register is located. You can include a file at the beginning of your code to assist the assembler in translating your code correctly. Usually included with the assembler, this file contains a list of the names of all of the registers and the specific addresses of each within the MCU.

As the instructions get more complicated, you'll see why the assembler is a real time saver. Moving or operating on data can require multiple operands a source and destination. Sometimes these are implied by the instruction itself, such as moving data in or out of the working register (WREG). The data could be in the form of a constant or value from another register. The CPU's program counter initially points to the reset vector location and increments through the program memory as each instruction is executed. Your program must occasionally branch either conditionally or unconditionally to a new area.

Calculating the correct value to add to the program counter so the next instruction comes from the right place is a real pain. Assembly language enables you to use labels. When your assembler assembles your code, it

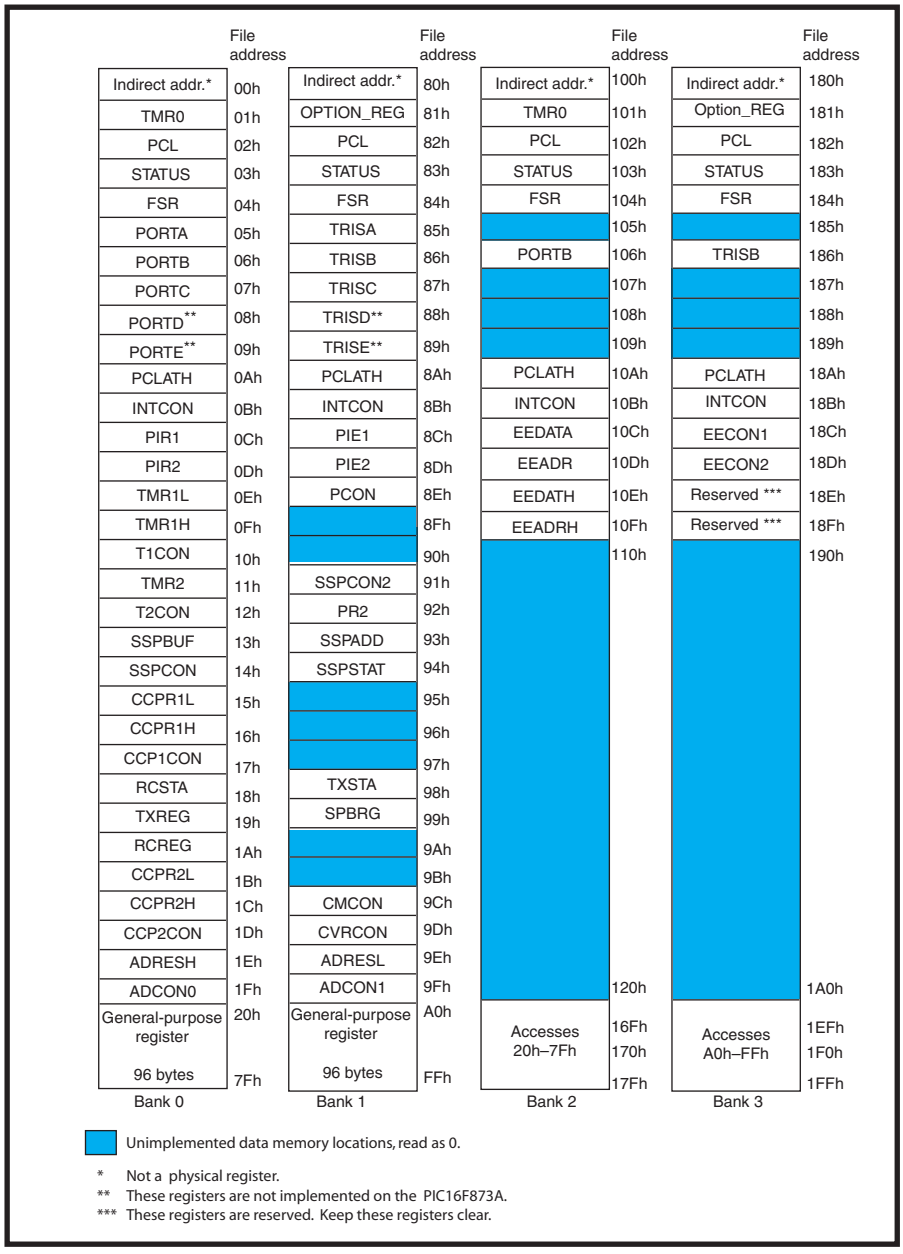


Figure 3—*Registers in a microcontroller are actually RAM memory. They include some general-purpose RAM for user variables.*

takes care of all the tedious calculations.

Assembly language is the bottom rung on the computer language ladder. Just as technology is forever improving hardware, higher-level programming languages (e.g., BASIC, C, and Java) are constantly being created. Because these higher-level languages normally require a certain amount of memory overhead, you may find yourself forced into using assembly language. In any event, you'll always use an assembler because even the high-level languages compile their code into assembly language, which is then assembled into a final hex file (or similar output file with another format).

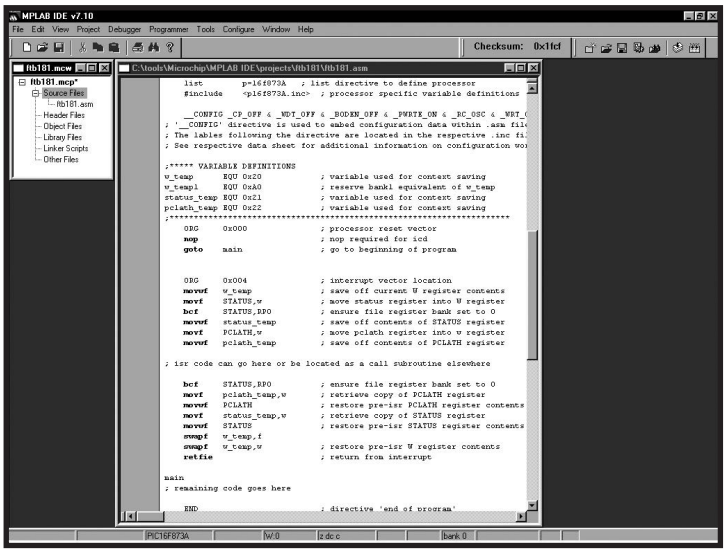


Photo 1—*Templates for each MCU give you a place to begin by including some necessary initialization stuff. Comments help explain what's included and where to put your code.*

TEMPLATE FILE

Let's take a look at the template file I added to my project (see Photo 1). This gave me (and MPASM) the minimum code necessary to assemble the file. First a look at what MPASM receives from this file.

The list option `p=16f873a` tells the assembler what kind of microcontroller is being used. The `#include <p16f873a.inc>` directive instructs the assembler to grab the p16f873a.inc file and assemble it (as if it were physically inserted in the project file at this point). The file, which is in the MPASM Suite folder, defines all the available registers, the configuration bits, and the amount of RAM available for this processor. Now I don't have to worry about defining the registers, finding the register and configuration bits values, and locating the RAM. For instance, you don't have to know that the Z flag in the Status register is bit 0x02 of register 0x03. You can just call it `STATUS,Z`!

Next, the `__CONFIG` directive includes a list of configuration bits with their on or off statuses (enabled or disabled). These include things like the kind of clock you will supply as the time base. It might be an actual clock source or just a component that when combined with the internal amplifier creates an oscillator (xtal, resonator, or resistor/capacitor).

A few variables are also defined for you; they're assigned to the lowest registers (RAM) available in the MCU (see Figure 3). These particular variables are only needed when an interrupt routine is used. (More on interrupts in a minute.)

The `ORG 0x000` directive tells the assembler where to place the next portion of assembled code in program memory. Now MPASM will assemble something. Instructions are usually in the form of a mnemonic with optional operands. The first instruction is NOP. This is a do-nothing instruction that doesn't require an operand. The assembler translates this to the 0x0000 value and stores it at address 0x000 (as directed by the previous `ORG` statement).

The next instruction is `goto main`. The `goto` mnemonic requires a destination operand (the `main` label in this case). Because the assembler has not come across any definition for `main` at this point, it cannot translate this instruction into a value. Thus, it plops down a place-holder at the 0x001 address and remembers to come back to it on a second pass.

Again, you come to an `ORG` directive, which instructs the assembler to place the next assembled code at address 0x004. This happens to be the interrupt vector, or the place to which an interrupt is directed. An interrupt is a routine that must be executed on demand. A timer value is reloaded if an overflow has occurred.

Interrupts can occur any time your program is executing. When an enabled interrupt occurs, your code execution stops and the CPU branches to the interrupt vector where you have a chance to do something special before the CPU returns control back to your normal code execution. Because an interrupt routine can use the same registers as the normal program, it can corrupt data in the register if it's used in the interrupt routine. To prevent this, save the original register content used by the interrupt and your normal code before executing the interrupt routine. Replace it before you leave.

The next six instructions assemble addresses 0x004 through 0x009. These instructions save the W register to the previously defined w_temp register. Because you can come to this interrupt code while the bank select bit is set to 0 or 1, you actually need two places to save W. In this case, you can use the first register in either bank: 0x20 (used if the bank select is 0) and 0xA0 (if the bank select is 1).

The W register is the most frequently used register. After it's saved, you can use it to save the rest. The Status register is first moved to the W register so that when you force the register select bit to a 0—which is the bank for saving the Status and all other registers—its original value is preserved. It's the original value because by forcing the register select to page 0, you'll clear the RP0 bit in the Status register. Without saving the original value to the W register, there will be no way of restoring status to the proper register bank when the interrupt routine finishes.

Next, you'll see a comment showing where any interrupt code should be placed. Following any interrupt code, the next eight instructions will

restore the three registers back to their pre-interrupt values and exit the interrupt. There is no interrupt code in the template at this point.

Lastly is the label `main`, which was referred to in the `goto main` mnemonic. The assembler knows that the next instruction will be located at the 0x012 address, so it now relates any `main` label with program memory address 0x12. The `END` statement indicates the end of the program and lets the assembler go back and substitute values for those instructions that may have used labels (unresolved labels will cause assembler errors).

QuickSIM

I don't always test my code. Sometimes I just program the microcontroller, stick it in the hardware, and power it up. I don't have much luck doing this. It usually fails because of a dumb mistake. If you have a PCB to plug your MCU into and an emulator or in-circuit debugger, you're golden. You can follow your code until it fails. But if you don't have any of that, you're stuck—stuck, that is, if you don't have access to a simulator.

A simulator is a PC application that simulates MCU operation. The simulator creates a virtual MCU and enables you to crawl inside and investigate how executing your program code will affect all of the MCU's registers. Like the assembler, the simulator needs a few pieces of information to configure itself, the MCU (to identify the registers needed), and the speed of the clock or oscillator (to enable the timing of instructions to be calculated).

Not all simulators offer the same tools. Besides the basics of running, stepping, and halting code execution, some simulators offer simulations of external forces on the MCU's inputs. Because many applications take action based on external events, the stimulus enables the virtual inputs to be virtually stimulated. This can be as simple as setting or clearing an input; it can be as complex as simulating a serial communication bit stream using a stimulus file.

A stopwatch function that can be reset presents an elapsed execution cycle count. Using the clock frequency, it calculates the actual execution time. This enables you view the accuracy of critical timing functions like the bit timing of a software UART function.

By combining functions like Watch Window (for viewing selected registers) and Breakpoints (a selected point in your program where the simulation is automatically halted), you can monitor variables of interest at specific times within the program. This helps to zero in on improper operation.

One of the most powerful functions is Trace, which is a log of every line of code as it's executed. It includes information on the source and destination registers of each instruction and how their values may have been affected by the instruction. Microchip's trace buffer can hold 8,192 instructions. It can halt an executing simulation on overflow or be used as a FIFO ring buffer, holding just the last 8,192 instructions.

CODE TIME

Now you know enough to use the simulator and debug some code. Let's simulate the template as it stands before adding any new code.

Click Select Tool from the Debugger pull-down menu and choose MPLAB SIM. Execution commands are available in this pull-down menu, Fx shortcut keys, and the menu bar (arrows). Before executing the template code, click Settings on the Debugger pull-down menu. In the OSC/TRACE tab, enter the clock frequency (4 MHz) and enable the Break on Trace Buffer Full box.

Clicking the Run button in the menu bar (or pressing F9) will run the simulator. You shouldn't see much happen, but the Run button should quickly flash as the Pause button is highlighted. The simulator will start executing code and then stop.

After you click Simulator Trace from the View pull-down menu, the Trace Window will pop up. You'll find that it has filled and halted program execution, but you won't have even put in any code yet. What's going on?

Use the right-hand scroll bar to get to the beginning of the trace buffer. The first line executed will be the `NOP` instruction followed by a `GOTO main` or `GOTO 0x12`, which the assembler will eventually calculate as address 0x12 when it finds the `main` label after assembling the 12 previous instructions to addresses 0x00 through 0x011 (see Photo 2).

The disassembled listing won't show anything after the `RETFIE` instruction. The `main` label, although recognized and used by the assembler, won't have any mnemonic, so no code will be generated at this location. From this address on the MCU, it will show its default (cleared/erased) value of 0x3FFF. Note that this value equates to instruction mnemonic/operand ADDLW 0xFF, which adds the constant 0xFF to WREG.

You'll see in the trace window how the WREG (DA [W]) data initially equal to 0x00 is continuously incremented by 0xFF (DD [0xFF]). The simulator won't know it has run into the blank section of the MCU.

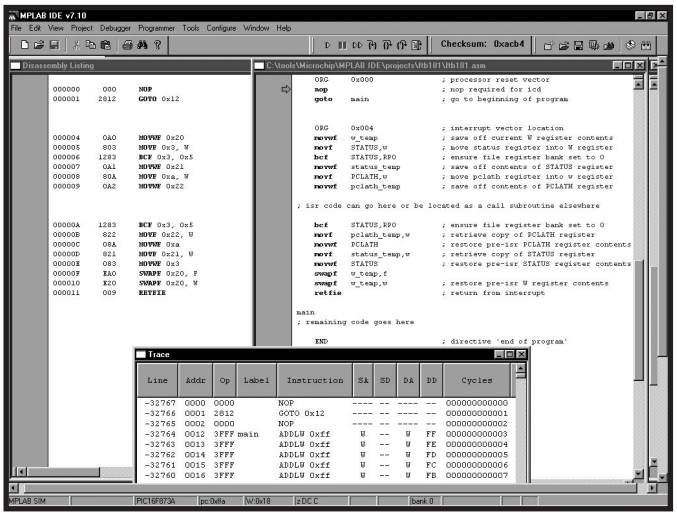


Photo 2—*The simulator gives you the ability to single-step through a program and monitor how each instruction affects the MCU's registers. The Trace function can log each executed instruction. This enables you to follow the program flow and find those unexpected detours that can crash your code.*

It will just keep on executing until the trace buffer is full and halts execution. You can prevent this if you put some code at the main label.

In the ftb181.asm window, add the variable count to the variable definitions with the next available user RAM location 0x23. Next, add the following four lines of code after the main label:

```
incf  count,f
movf  count,w
movwf PORTC
goto  main
```

The first line will grab the value from the count source register, increment it, and then put the incremented value back in the count destination register, where f indicates the use of the source register and w indicates the use of the WREG. Following this, the value will be plucked again from the count source register and moved to the WREG destination register. Now the value in the WREG register will be moved to the PORTC destination register to light some LEDs on the port C I/O pins.

Finally, the already familiar goto main instruction will jump (this time backwards in memory space to the 0x12 address previously established by the assembler when it came across the main label). This loop will continuously increment the count value once each time through the loop. If you set up a Watch Window for the count and PORTC variables, stick a breakpoint at the goto main instruction, and then run the program, the simulation will stop each time it reaches this breakpoint. You'll notice that count is incrementing and PORTC isn't.

If you look at TRISC in the special function registers (or add it to the Watch Window), you'll see it has a value of 0xFF. This is the direction register for port C I/O that defines whether the individual I/O pins are inputs (1) or outputs (0). Because all the bits are ones, every I/O bit is set as an input. By clicking the TRISC value, you'll be able to change it to all outputs (from 0xFF to 0x00). Running the simulator again will show that PORTC now reflects the count value.

Because all I/O ports default to inputs when the MCU is reset, you must add instructions to set TRISC as all outputs prior to executing the instructions in the main routine. I generally replace the initial goto main instruction (back at the reset vector) with goto init and place the init label just before the main label. Any registers that need to be initialized (e.g., TRISC) can be placed between the init and main labels.

Assembling the code and running the simulator again doesn't increment port C. You think you clearly have a CLRF TRISC instruction after init, yet TRISC doesn't get set to a value of zero. If you notice TRISC's address (0x87), you'll find that it is in bank 1. Unless you remember to change banks (RP0 in the Status register) before trying to write to the TRISC register, you'll actually clear the register 0x07 (port C) in bank 0. Don't forget to change back to bank 1 when you're done, or at least be aware of where you are the next time you use a register. This is a tough rule to remember whenever you're using PICs. Always be aware of your bank location and adjust accordingly.

You might choose to use the banksel directive instead of manually setting and clearing bank bits, especially when multiple bits are involved. I kept it simple (see Listing 1).

LAST HURRAH

Before I unleash you into the world of simulation, let's take a look at the process of using an interrupt. Many peripherals in the MCU have the ability to generate an interrupt. The mechanism for using interrupts is the interrupt flag, the interrupt enable, and the global interrupt enable. If you want to use a peripheral's interrupt, you must enable its interrupt by setting the peripherals interrupt enable bit, which is found in the INTCON and PIEx registers.

After you've initialized and enabled all of the peripherals you need, you can set the global interrupt bit. The global interrupt enable (GIE) bit is automatically cleared when the execution flow is diverted to the interrupt vector (and enabled by the RETFIE instruction at the end of the interrupt routine). This prevents additional interrupts from interrupting the interrupt. When a peripheral wants to interrupt the normal execution flow, it raises its interrupt flag. If the peripheral's interrupt enable bit is set (and if the global interrupt bit is set), the CPU will redirect execution to the interrupt vector. The template I used saves a few registers, makes a place for the code to reside, and then restores the registers prior to returning to where execution was interrupted.

The code entered thus far increments a value just as fast as the MCU will execute the loop of code. Let's slow this down by requiring 1 ms between each increment. You can place a lot of code in the main loop so that it finishes the loop in 1 ms, or you can use a timer to measure 1 ms and let you know when to increment the count variable. If you use Timer1 as a 1-ms source in a polled mode (no interrupt), you must initialize a few timer registers and a constant. Add this code to the ftb181.asm window (see Listing 2, page 56).

Timers overflow when they reach their maximum count (in this case 0xFFFF), so you must create a constant (an unchanging value) to reload into Timer1 so it takes 1,000 µs to

Listing 1—*Adding these few instructions to the template creates a basic* count *loop. The number of the instructions in the loop creates the actual timing of the loop.*

```
init
    bsf     STATUS,RP0     //Change to bank 1
    clrf    TRISC          //Make all output bits
    bcf     STATUS,RP0     //Change to bank 0
main
    incf    count,f
    movf    count,w
    movwf   PORTC
    goto    main
```

Introducing our New Suppliers

with NEW Products & NEW Technologies

Lattice Semiconductor Corporation



Chipcon





freescale™ semiconductor

Delivering the latest with the greatest.

With a NEW Catalog Every 90 Days, Mouser is your source for **NEW Products, NEW Suppliers,** and **NEW Technologies.** Mouser is the catalog distributor focused on the needs of engineers. We deliver the latest technologies with the great service and support you need.

SEMICONDUCTORS | PASSIVES | INTERCONNECTS | POWER | ELECTROMECHANICAL | TEST, TOOLS & SUPPLIES



Supplier Authorized Distributor

New Products
New Suppliers
New Technologies
New Catalog Every 90 Days!

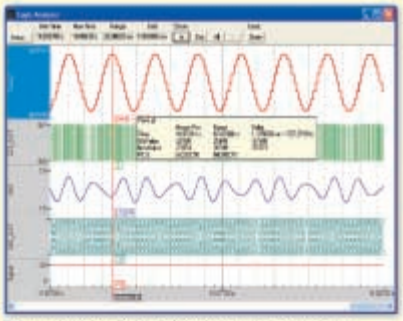
MOUSER ELECTRONICS
a tti company

(800) 346-6873
www.mouser.com

Mouser® and Mouser Electronics® are trademarks of Mouser Electronics, Inc.

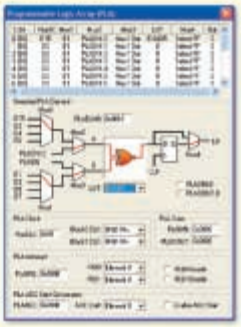
KEIL SOFTWARE

Keil Professional Embedded Development Solutions provide the tools you need for your 8051, 251, C16x, and ARM projects. Tools from Keil help you efficiently create, test, & complete your embedded projects.



The µVision3 IDE/Debugger supports virtual target simulation and includes a software logic analyzer that graphically displays variables and I/O registers.

The µVision3 Development Environment provides complete device simulation and includes peripheral dialogs that aid development of software drivers.



µVISION 3

Supported Architectures

ARM7

Keil development tools for ARM support a wide variety of devices from the most popular chip vendors.

C166

Keil development tools for the XC16x, C16x, and ST10 support over 50 devices from Infineon and STMicroelectronics.

Cx51

Keil development tools for the 8x51 support over 600 different derivatives including devices with extended address space and enhanced instruction sets.

Contact us today for a Free Demo CD

800-348-8051

www.keil.com

count up to 0xFFFF and overflow to 0x0000. I chose a 4-MHz oscillator, so one instruction cycle is equal to 1 µs. If you need a 1,000-µs delay (1 ms), then the calculation is easy. Set up Timer1 to tick using the instruction cycle clock (osc/4): one tick, 1 µs. The fudge factor is zero for now.

After the code added to the `init` section loads the timer with a constant calculated by the assembler from the constant definition section, it turns on Timer1. In the `main` section, poll for Timer1 to overflow by repeatedly checking Timer1's overflow bit. When Timer1 overflows, ignore the `goto main` loop, proceed by resetting Timer1 for another 1,000-µs count, and use your original code, which increments the `count` variable (and PORTC register) and jumps back to `main` (polling for another Timer1 overflow).

After assembling the new code (and with a breakpoint still at the last `goto main` instruction), you can use the stopwatch to measure the execution time between breakpoints. Remember to reset the stopwatch before each run after the breakpoint stops the execution. After you're in the loop, the stopwatch will show the same amount of elapsed time, but it won't be 1 ms because it will take time to set up the timer reload values and execution cycles have been expended.

Listing 2—*By using a timer, the actual timing is independent of the program execution. The timer also enables the timing to be changed easily because it's based on the comparison of values (or overflowing) and not on how many instructions are in line.*

```
After the variable declarations, add constant definitions.
*********************************************************************
fudgefactor     EQU     D'0'
onemS           EQU     0x10000 - D'1000' + fudgefactor

*********************************************************************
At the end of the init code, add the following
*********************************************************************
   movlw   HIGH onemS    //Use the high byte of the onemS constant
   movwf   TMR1H         //As the high byte of the timer1 reload value
   movlw   LOW onemS     //Use the low byte of the onemS constant
   movwf   TMR1L         //As the low byte of the timer1 reload value
   bsf     T1CON,TMR1ON  //Enable the timer
*********************************************************************
After the main label, add the following
*********************************************************************
   btfss   PIR1,TMR1IF   //Skip next if Timer1 overflows
   goto    main          //otherwise goto main
   bcf     PIR1,TMR1IF   //Clear the overflow flag
   bcf     T1CON ,TMR1ON //Disable the timer
   movlw   HIGH onemS    //Use the high byte of the onemS constant
   movwf   TMR1H         //as the high byte of the Timer1 reload value
   movlw   LOW onemS     //Use the low byte of the onemS constant
   movwf   TMR1L         //as the low byte of the Timer1 reload value
   bsf     T1CON,TMR1ON  //Enable the timer
```

Listing 3—*The timer raises a flag (sets a bit) when the required count is achieved (or an overflow occurs). You can choose to continually watch for this and change the program flow when it happens.*

```
btfss   PIR1,TMR1IF    //Skip next if timer 1 overflows
goto    main           //otherwise goto main
```

Listing 4—*An alternative to polling for the timer flag is to let the flag initiate an interrupt. If the timer interrupt is enabled when the timer flag goes high, execution can be automatically and temporarily redirected to the interrupt vector area. You can place code here that determines which device raised its flag and what to do about it. The program flow then will be directed back to where it was before the flag was raised.*

```
bsf    STATUS,RP0    //Change to bank 1
bsf    PIE1,TMR1IE   //Enable Timer1 interrupts
bcf    STATUS,RP0    //Change to bank 0
bsf    INTCON,PEIE   //Enable peripheral interrupts
bsf    INTCON,GIE    //Enable global interrupts
```

Use a fudge factor value to automatically adjust the reload value in the constant calculation. This way you will be able to change the 1,000 µs to 500 µs and use a real number without having to worry about adding the fudge factor to the value. Use a value of `D'10'` for the fudge factor to compensate. Because the polling loop takes three cycles (one for the poll and two for the `goto`), the timer will actually overflow on one of the `goto` cycles. This can throw off the attainable accuracy by two cycles. Rerun the simulation and check the new timing with the stopwatch.

Now let's remove the timer code from the `main` routine and make it an interrupt event. Remove these instructions (poll loop) from the `main` section (see Listing 3). Move the rest of the code in `main` (except the last `goto` instruction) to the interrupt routine (where it says "isr code can go here"). Add the Timer1, peripheral, and global interrupt enable instructions to the end of the `init` section and add an NOP just after `main` (see Listing 4).

The `NOP` in the `main` section (just before `goto main`) gives the `goto` somewhere to go to instead of itself. (This prevents a tight single instruction loop.) Next, clear the previous breakpoints and add a breakpoint to the `movwf PORTC` instruction in the interrupt routine. Now assemble and run the simulation. Because there are more instructions between the interrupt vector and the breakpoint, you'll need to increase the fudge factor.

The program stops each time it executes the interrupt routine. The code placed there will update the `count` and `PORTC` values each time Timer1 overflows. This operation is completely independent from the code executing in the `main` loop, which right now is an NOP and `goto`. If you look at the trace window after the code has stopped for the first time, you'll see the initialization being executed (at the beginning of the trace buffer) and then a bunch of NOP and `goto` instructions (the main loop executing over and over) until Timer1 overflows. At that point the execution will branch to the interrupt vector where the interrupt code is executed and the breakpoint is encountered.

TAKE OVER

I'm passing you the reigns at this point. You can add your own code to this program. You can experiment with other instructions and investigate the use of the other peripherals in the microcontroller. You'll find the answers to many of your questions in the pull-down Help section. Read, read, read. There are documents available for every microcontroller and tool in the MPLAB IDE.

Although simulators may have been designed as tools to assist you with the process of debugging code, they're the perfect tools for learning how microcontrollers and peripherals work. There are certain real-world interfacing operations that can't be simulated, but there's nothing like writing some code and seeing it execute.

Play for a while. You'll be ready to plunk down some cash to bring your ideas to life with hardware after you've built up your confidence in your programming ability. Go forth and simulate.

Jeff Bachiochi (pronounced BAH-key-AH-key) has been writing for Circuit Cellar *since 1988. His background includes product design and manufacturing. He may be reached at jeff.bachiochi@circuitcellar.com.*

PROJECT FILES

To download the code, go to ftp.circuitcellar.com/pub/Circuit_Cellar/2005/181.

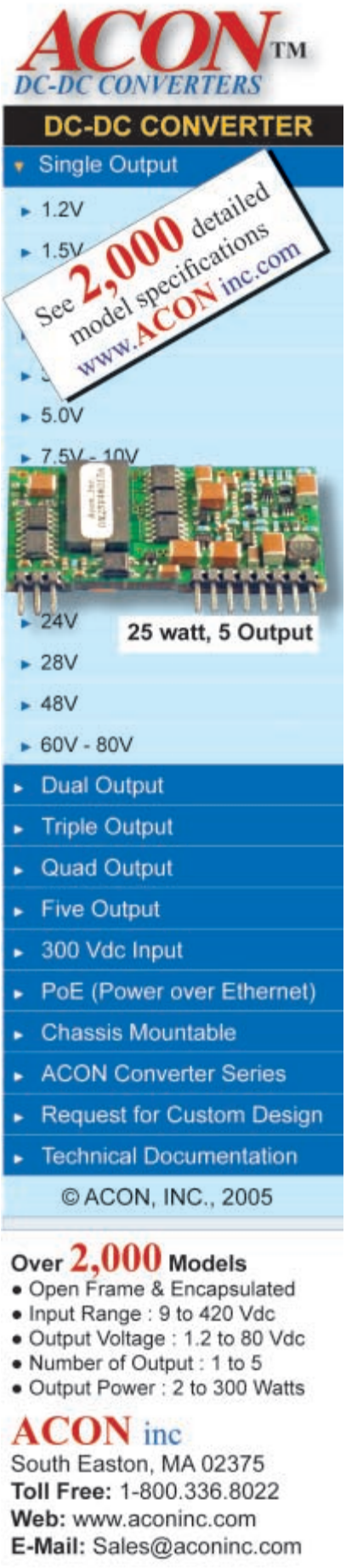
RESOURCES

Development tools for MCUs, Texas Instruments, http://focus.ti.com/docs/tool/matrix.jhtml?FamilyId=4.

SOURCES

AVR Studio
Atmel
www.atmel.com

MPLAB IDE and PIC16F873A microcontroller
Microchip Technology
www.microchip.com



You Have The Power

- **Fastest Rabbit ever with 51.6 MHz clock speed for data crunching and program execution.**
- **Up to 2000 mA of on-board power to drive your system.**
- **Balance features and cost with FLEX manufacturing. Pay only for what you need!**



Speed Up Your System and Lower Design Risk With The 'FLEX Ready' PowerCore Module

The PowerCore is a complete Rabbit® system featuring our fastest Rabbit 3000 microprocessor, available on-board power supply, analog features, Ethernet, and a rugged A/D system. The PowerCore is "FLEX Ready" and can be configured with the exact options you need, and we quickly manufacture it. Development Kits include hardware, the Dynamic C development platform, and extensive libraries.

FLEX™ Ready

Configure Your PowerCore Online **www.PowerCoreFLEX.com**

From $28 qty. 5000 | Kits From $129

Learn More about PowerCore

It's easy to get started with a complete development kit that includes your PowerCore and development tools. For a limited time get a FREE Rabbit design book with your kit.

www.PowerCoreFLEX.com



RABBIT Semiconductor

2932 Spafford Street. Davis, CA 95616 Tel **530.757.8400**

Solutions That Work

PSoC Pulse Generator

Brian built a digital pulse generator for his home several years ago. However, when the HP pulse generator he uses at work died, he hit the bench and designed a replacement for it, using a Cypress PSoC-based version of his original design.

The Hewlett-Packard pulse generator at my office served me well for more than 20 years. But the potentiometers and multipole range switches on the analog design eventually wore out. Unfortunately, all of the front panel's potentiometers and switches were discontinued custom PCB-mounted parts. So, I recently replaced the ailing generator with a homebrew unit like one I built eight years ago around the now discontinued Am9513 system timing controller ("Pulse'05 Pulse Generator," *Circuit Cellar*, 78, January 1997).

The Am9513 was the Cadillac of timer chips in its day. But I couldn't use it for this project. Apart from being obsolete, it could only count at a maximum of 6 MHz. I need to be able to generate pulses below 100 ns, so I needed a faster timing device.

I use many different MCU families, so I thought it would be easy to find MCUs with high-speed timer blocks to perform the timing functions I needed. I first checked out the 50-MHz Zilog eZ80 Acclaim! module. No luck. Although the CPU motored along at 50 MHz, the fastest clock the timer blocks would accept was the CPU clock/4, or 12.5 MHz. My old standby, the Atmel AVR family, was no help either; it didn't have enough timer blocks, and the maximum clock rate was 8 MHz. That's why I went with a Cypress PSoC. I had recently taken a closer look at the family, and was impressed its availability of configurable digital and analog blocks.

Range	Resolution	Clock source
0.01–600 s	0.01 s	CNT8_1 (VC3 /100)
0.1–6000 ms	0.1 ms	VC3 (SysClk/2500)
0.1–1200 µs	0.1 µs	SysClk × 2
20–60000 ns	20 ns	SysClk × 2

Table 1—*The pulse generator's main clock rate, as well as Output1 and Output2 pulse widths and delays, can be set with these four ranges.*

I typically work on mixed-signal projects. The newest PSoC devices sport analog function blocks that are of a high enough quality to be used in the scientific applications I routinely work on. As luck would have it, they contain digital blocks that can function as 8/32-bit timers/PWMs with clock rates up to 50 MHz. This enabled me to duplicate my old design using a $6 CY8C27443 (see Photo 1) instead of the pricey Am9513 and supporting MCU.



Photo 1–*The PSoC Pulse Generator fits into an 8″ × 5″ × 5″ cabinet about the same size as the HP generator it's replacing. All parameters are adjusted using the two rotary encoders mounted on the upper left side.*

FUNCTIONALITY

For my project, I needed to be able to generate logic-level pulses with a programmable repetition rate and pulse width. The slowest period/pulse widths might extend into the hundreds of seconds. The fastest pulses I require might need a resolution of approximately 20 ns. Furthermore, I wanted to be able to generate three independently adjustable, synchronized signals.

Specifically, this design generates one programmable square wave clock as well as two additional output pulses, each of which has a programmable delay from the rising edge of that clock as well as a programmable pulse width. Table 1 shows the ranges and resolution of the output signals.

To make the unit more versatile, I added a gate input, which enables the external signal to gate on and off the generator's output. I also included an inverting driver for each output signal, thus enabling either active-high or active-low pulses to be generated. The PSoC digital outputs can source 10 mA and sink 20 mA at 5 VDC, which is sufficient drive for my purposes. The inverting drivers are 74128s. These fairly uncommon devices can drive 50-Ω loads, but a 7402 NOR gate is pin-compatible, and could be substituted if 50-Ω drive capability isn't needed.

The front panel user interface isn't fancy, but it does the trick. A two-line LCD module displays the parameter currently under adjustment. One rotary encoder module acts as the parameter Menu selection knob. The second rotary encoder (Value) adjusts the value of the selected parameter. On most timing scales, the range of the pulse timing is 60,000:1. With such a wide range, setting a particular value with a rotary encoder can be tiresome, so I added a coarse/fine switch that adds a 100× faster adjustment rate.

CY8C27443

The PSoC family comprises devices in packages ranging from the 8-pin SOIC/DIP up to the 100-pin TQFP.

The CPU core in all PSoC devices is the M8C, which is a somewhat plain 8-bit Harvard architecture CPU with few registers and a relatively small instruction set. Most PSoC devices also include one or more 8-bit multiply/accumulate blocks (MAC), which can perk up the M8C's 4 MIPS rating for some signal processing applications. Don't get too preoccupied with the M8C CPU. It's the versatility of the analog and digital blocks that make the PSoC family shine.

Apart from three TTL glue chips, the Cypress CY8C27443 microcontroller handles all of the pulse generator's functions. The 28-pin CY8C27443 contains 24 I/O pins, 12 of which can be optionally configured as analog inputs and four of which can serve as analog outputs.

All PSoC devices incorporate flash memory. The CY8C27443 has 16 KB of flash memory and 256 bytes of RAM. In-circuit serial programming is easily achieved with the $35 CY3210-MiniProg1 USB mini programmer. You can splurge on a full in-circuit emulator or production programmers if you wish.

The strength of the PSoC architecture is in its configurable digital and analog system blocks. Figure 1 shows the layout of the digital blocks, which are arranged in several rows containing four columns. The CY8C27443 contains two rows. The newer CY29*xx* family of PSoC devices doubles the digital block complement to four rows. The digital blocks are basically generic modules that can be configured into functions such as timers, counters, PWMs, and random sequence generators.

What isn't obvious in Figure 1 is that two of the digital blocks (DBB*xx*) perform only these basic tasks, while the other two (DCB*xx*) are communications blocks. The communications blocks, in addition to being able to perform the basic tasks, can be configured for UART, SPI, I²C, and IrDA communication.

Interconnection options between the various digital modules aren't unlimited, but there is a reasonably good choice of connection options based mainly on the geometric relationship between the modules. Similarly, there's no big cross-point switch allowing any digital block to connect to any external I/O pin. However, the choices are reasonably good, so using some imagination for the placement of the different functional blocks will generally yield a design that can be routed.

What really sets PSoC devices apart are their analog blocks, which are made up of a number of conventional (i.e., continuous-time) op-amps, as well as a number of switched-capacitor blocks. The continuous-time amplifier blocks are much like regular op-amps, but they're augmented with various mux switches to modify their gain and configuration. For example, the blocks can be connected to form differential input instrumentation amplifiers, PGAs, etc. I'm not about to toss out my Burr-Brown high-performance amplifiers when measuring thermocouples and strain gauges, but the continuous-time amplifiers in PSoCs are fine for general use.

The highly configurable switched-capacitor blocks are used to create SAR, delta-sigma, and integrating ADCs, for example. They also can be strung together to form integrators and various filter topologies.

Figure 2 shows the layout of the analog blocks in the PSoC devices. Whereas the digital blocks were organized in rows, the analog blocks are organized in columns, with the larger PSoC devices containing four columns instead of two. Each column consists of one continuous-time block and two switched-capacitor blocks.

All of the blocks within a specific column have common characteristics. For example, they must share the same clock signal. As with the digital blocks, the interconnections among the analog blocks are limited: there is a finite number of routing paths from any one block to the others. In practice, it's sometimes necessary to route an analog block's output signal to an external I/O pin and then jumper it to another I/O pin. This enables it to be routed to the input of the analog block being used as the next stage.

Most MCUs contain medium-resolution ADCs (generally 10-bit, SAR-type converters). In this respect, the PSoC is more versatile because its configurable blocks can be configured to provide SAR, delta-sigma, and integrating ADCs with resolutions from 6 to 14 bits. Additionally, while most MCUs lack on-board DACs, the PSoC can be configured to provide 8- or 9-bit DACs using the switched-capacitor blocks. Alternately, high-resolution (albeit slower), 12- to 16-bit DACs can be obtained using a PWM block and an analog modulator block. Even though the basic architecture of the analog

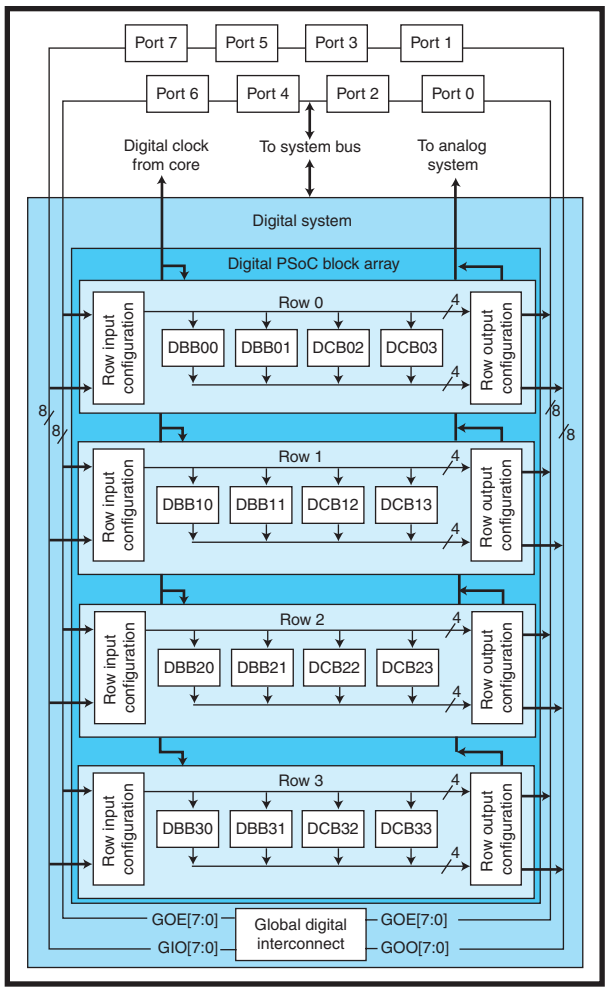


Figure 1—*The PSoC's digital blocks are organized in rows, with each row containing two basic blocks (DBBxx) and two enhanced communications blocks (DCBxx). Currently, only the newest CY8C29xx family contains all four rows shown here.*

and digital blocks has stayed the same as the PSoC family has evolved, astute engineers are constantly coming up with better ways to configure them for higher-performance user modules.

PSoC devices have extremely versatile clocking options, which is what prompted me to consider them for this project. An external clock oscillator or a 32-kHz watch crystal, which is multiplied internally by a PLL to generate a 24-MHz system clock, can drive PSoCs. But, PSoCs will also function using an internal 24-MHz RC oscillator. Unlike the RC oscillators in most MCUs, the accuracy of a PSoC's internal 24-MHz clock source is extremely good at ±2.5%, so many applications don't need an external crystal. For this project, however, I needed an external crystal oscillator to ensure highly accurate pulse periods and widths.

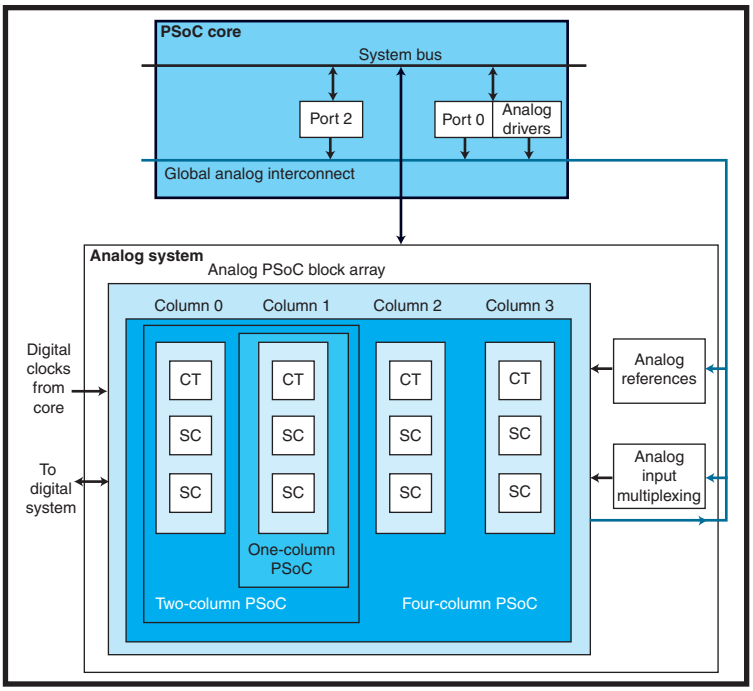


Figure 2—*The PSoC's analog system is organized in columns, with the smaller devices in the family containing only one column. Each column contains a single continuous-time block (CT) and two switched-capacitor blocks (SC).*

The PSoC contains three separate programmable prescalers that provide three independent clock sources for use with the various digital and analog blocks. These prescalers can be used independently or cascaded to generate clock rates down to 366 Hz, all derived from the 24-MHz internal main oscillator. There's also a clock doubler circuit that generates a 48-MHz clock from the 24-MHz internal main oscillator. For this project, I wanted a 50-MHz time base for the pulse generation circuits, so I used an external 25-MHz oscillator module to supply the PSoC's system clock and got the 50-MHz time base from this clock doubler. At 5 V, this 4% over-clocking presented no problems.

I like the power supply flexibility of the PSoC devices. They can operate between 3.3 and 5 V, but they also contain a built-in charge pump to allow them to work with power supply voltages as small as the 1.2 V available from a single NiCd cell. The only external exponents required to implement this charge-pump are a small inductor, a capacitor, and a diode.

PSoC DESIGNER

The process of configuring the generic digital or analog blocks to perform a specific function would be daunting if it had to be done by hand. Cypress has taken a lot of the sting out of the process by including a broad range of user modules with the PSoC Designer software.

The initial design phase is performed using the Device Editor portion of the software. You must simply pick a user module (e.g., DeltaSig8 ADC) from a pull-down list, cycle through the valid physical placements within the row/column structure until you find one that suits your design criteria, and then place it. Following this, the program will bring up a configuration window that will enable you to specify the parameters to suit your particular needs (e.g., gain, signal source, and clock source). What's equally impressive is that after you've chosen a user module, one click will bring up an online datasheet with links describing the module.

You can use the Designer to customize the configuration of your PSoC device in two distinct ways. The power-up default configuration is defined in the Device Editor. However, any block can be reconfigured on the fly under program control by merely changing the values contained in the various registers associated with that particular block. In my project, for example, all three of the 16-bit timing blocks get their clock at run-time from different sources, depending on whether the units parameter is set for seconds, milliseconds, microseconds, or nanoseconds. This is performed by routines in my code that modify the necessary device registers.

Taken to the extreme, it's possible to use a concept called dynamic reconfiguration with PSoC devices. In this case, you first draws up a layout for the power-up default PSoC configuration. Next, you must choose Add Loadable Configuration from the Device Editor menu; this clears the slate and enables you to draw another layout that can, if necessary, redeploy any of the PSoC analog and digital blocks into different user modules.

After the application is compiled, the default and any other loadable configurations are coded and loaded into flash memory. At run-time, depending on external stimuli or user input, the PSoC device is able to pause, morph itself into a totally different device, and handle the external events appropriately. It can, of course, also return to its original configuration when necessary. I suspect I could have morphed my pulse generator into something else, like a sine wave generator. There was certainly enough flash memory space left over to handle the functions needed for both instruments.

Although a newcomer to the PSoC family, I was quickly impressed with the speed and user-friendliness of the PSoC Designer software. Virtually everything can be done within this program. It bears mentioning, though, that if you plan to use PSoC devices, the PSoC Designer software works

best with a 19″ display with 1,024 × 1,280 pixel resolution.

As you configure the device in the Device Editor, the available fields enable you to label the user modules. The Pinout Window on the right side of the screen is color-coded to highlight the I/O and analog/digital assignment of each pin; it also displays the descriptive net names that you've assigned in the I/O port definition window.

When you're finished designing the chip's configuration, you can use the PSoC Designer to generate a customized datasheet for the PSoC you're using. This slick, highly detailed document is full of diagrams and tables. It looks like something you'd compose by hand (if only you had the time!). The program includes several nice touches like this.

After you're through with the Device Editor, the program will generate all of the start-up code needed to set up the PSoC the way you've configured it. When you switch to the Application Editor phases of the PSoC Designer application, all the device configuration and start-up code will be readily accessible. This is handy because there are numerous registers involved in the process of configuring the digital and analog blocks into the selected user modules. Loading the registers at reset is achieved by block transfers of flash memory-based data into the various registers.

The PSoC Designer application provides you with well-commented source code files that clearly show how every register is loaded. An easy way to figure out the code needed to tweak user modules (when needed at run-time) is to manually set out the various scenarios in the Device Editor, jump to the Application Editor, and peruse the psocconfigtbl.asm file in the Library Source folder to find the proper register settings. I would then use these settings in my code. Again, this is a case of the programmers taking the time to provide additional features to make it easier for the end user.

The PSoC Designer IDE is designed to work with its own built-in assembler, but it also integrates an ImageCraft C compiler. The PSoC Designer software, including the assembler, is free. The ImageCraft C compiler is actually part of the package, but you must activate it by purchasing a license separately.

Whenever I start using a new MCU, I prefer to do my first few projects in assembly language, because it forces me to examine the architecture of the MCU and its on-board peripherals more closely. Therefore, I wrote this program in assembly language.

USER MODULES

Each user module included with the PSoC Designer program comes with its own online datasheet. You can access the datasheet in several ways, the easiest being to right-click on the user module from within the Device Editor.

The Resources section of the datasheet specifies the number of digital and analog blocks used by the user module. It also shows how much flash memory is needed to hold both the module's initialization code and the ISR routines needed to make it function at run-time. Here you can decide whether or not there are enough blocks in the PSoC to handle all of the user modules you think you'll need without actually placing all the modules in the Device Editor.

There's a block diagram of the user module in each datasheet. Figure 3 is a block diagram of the 16-bit counter modules I used in this design. The counter/timer/PWM blocks are similar to those found in other MCUs, but they also feature the ability to be easily cascaded to form up to a 32-bit counter. They can be clocked at the aforementioned high rate. They also feature the ability to terminate their count at either the "less than" or "less than or equal to" conditions, as determined by a register bit setting. I used this feature in another PSoC design to provide a high-resolution PWM by using the pulse-swallowing technique commonly used in PLL divider chains.

The Timing section of the datasheet isn't required for all user modules, but it's important for the counter, timer, and PWM user modules. It includes detailed timing charts showing the timing relationships between the various inputs, outputs, and system clocks. Similarly, the Specs section contains different types of information depending on whether the block is an analog or digital block.

The Placement section is an important part of the datasheet. Although the PSoC is made up of arrays of analog and digital blocks, all of which are configurable, it turns out that there are preferred placements for some of the analog blocks. If a user module is made up of numerous blocks, it's often important that the clock sources for all of those blocks be the same. The physical placement of the block on the die can affect timing slews. These fine points, if applicable, are detailed in the Placement section.

Also appearing in this section is a map of the generic blocks to the symbolic names used in the API section. The symbolic names of the mapped PSoC blocks are posted on the *Circuit Cellar* FTP site.

The API section contains all the details of the various API functions associated with that user module. When you are finished designing the PSoC configuration in the Device Editor and proceed to the Application Editor, a set of APIs is automatically generated for each module that you place. The APIs generated are either C or assembly language, depending on which language you chose when you started the project. The datasheet, however, shows both the C prototype and the assembly language calling conventions.

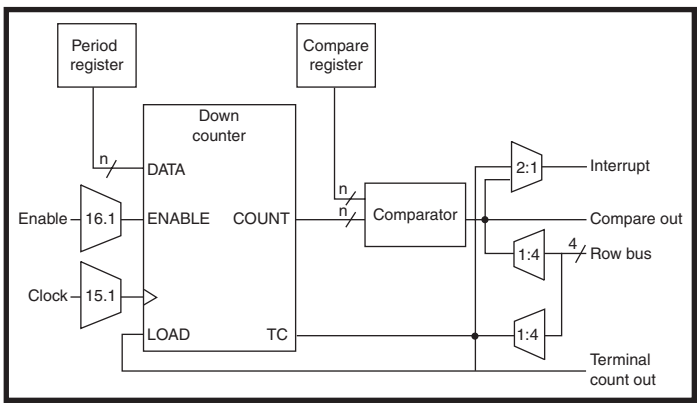
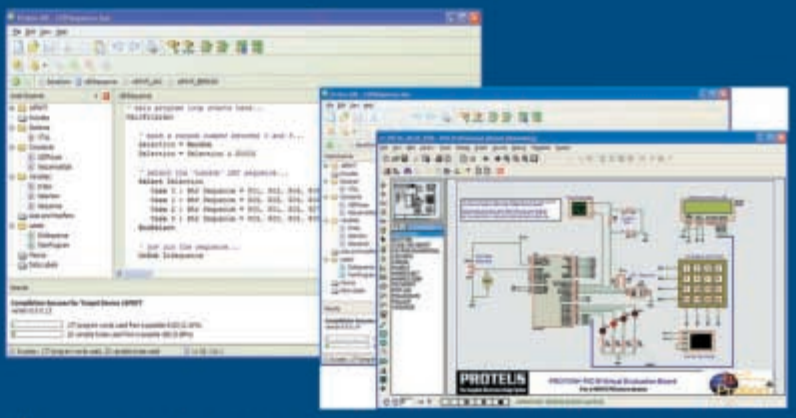


Figure 3—*I used a Counter16 user module in this project.*

PIC Programming Made Easy!

Proton+ PICBASIC Development Suite



Next Generation IDE

Proton IDE is a professional and powerful visual Integrated Development Environment (IDE) which has been designed specifically for the Proton Plus compiler. Proton IDE accelerates product development in a comfortable user environment without compromising performance, flexibility or control.

- *Code Explorer*
- *Compiler Results*
- *Programmer Integration*
- *Integrated Bootloader*
- *Real Time Simulation Support*
- *Serial Communicator*
- *Online Updating*
- *Plugin Architecture*

FREE DOWNLOADABLE DEMO

SAVE TIME WITH END TO END INTEGRATIONS

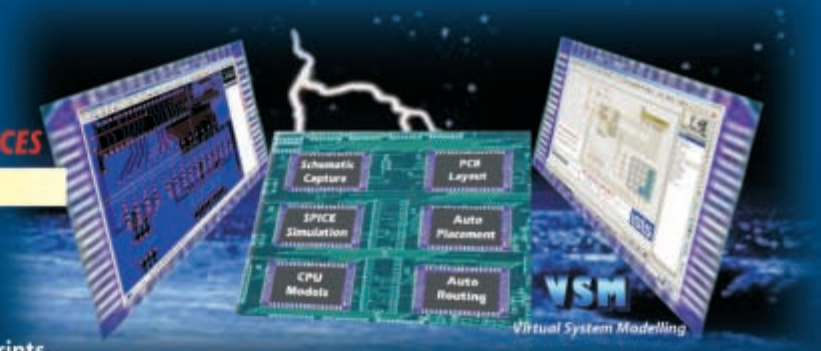
NEW IDE - New IDE makes development using Proton+ even faster and more intuitive!
COMPILER - The popular Proton+ compiler has enhanced support for I²C, SPI, Dallas 1-wire bus, RS232, X10, Compact Flash Memory Cards and USB.
VIRTUAL SIMULATION - Simulate your project in RealTime using the integrate Proteus Virtual PIC Boards.

Visit www.r4systems.com to see our latest projects using **Proton+**

PROTEUS

The Complete Electronics Design System

EASY TO USE CAD TOOLS AT FANTASTIC PRICES



Schematic and PCB Layout

- Powerful and flexible schematic capture.
- Auto-component placement.
- Rip-up and Retry PCB routing.
- Polygonal gridless ground planes.
- Library of over 8000 schematic and 1000 PCB foot prints.
- Bill of materials, DRC reports and more.

Mixed Mode SPICE Circuit Simulation

- Berkeley SPICE3F5 simulator with custom extensions for true mixed mode and interactive simulation.
- Six virtual instruments and 14 graph based analysis types.
- 6,000 models including TTL, CMOS and PLD digital parts.
- Fully compatible with manufacturers' SPICE models.

Proteus VSM - Co-simulation and debugging for popular Micro-Controllers

- Supports PIC16 & PIC12, AVR, 8051, HC11 and ARM micro-controllers. Latest version includes 40 new PIC18's.
- Co-simulate target firmware with your hardware design.
- Includes interactive peripheral models for LED and LCD displays, switches, keypads, virtual terminal and much more.
- Provides source level debugging for popular compilers and assemblers from HiTech PICC, Crownhill, IAR, Keil and others.

PCB AutoRouting

- Proteus PCB design includes an interface to the Electra Gridless autorouter.

"For the cost of the software compared to the productivity gains, I consider Proteus to be pivotal in the commercial viability of my company and by far represents the best value for money of anything Tempus possesses."

ROB YOUNGS, Tempus Consulting

FREE DOWNLOADABLE DEMO

Save Time. Save Money.

Proteus Starter Kit – $199

Complete Systems from $449

"This is clearly superior in every respect."

R4 SYSTEMS INC.

www.r4systems.com
Tel: 905•898•0665 info@r4systems.com

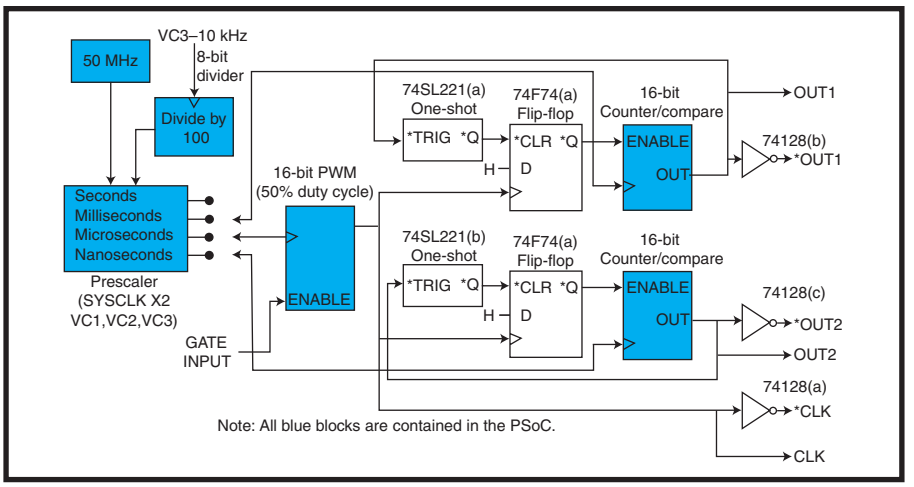


Figure 4—*Take a look at the pulse generation portion of the circuit. To get the versatility I wanted, I resorted to using two TTL glue chips.*

It's nice to see that the datasheet also includes sample code, which provides both the necessary initialization routines as well as a sample of the module's use. Again, both C and assembly language samples are provided.

The last section is the Registers section. In many cases you'll want to tweak a user module. This may involve bypassing the user API functions and going directly to the PSoC registers to modify a user module's behavior. The Register section contains details about which registers are used with that module. You'll probably still have to check the technical reference manual for details, but at least you'll know the name of the register to look for.

WIRE IT UP

Like almost any microcontroller-based project, it takes far less time to wire the project than it does to write the MCU program code. This is particularly the case with PSoC devices because you are doing most of the actual wiring in software when you configure the device to suit your project. I built this project using the invention board from the PSoC High Integration Challenge 2004 as a programmer. It was my only development tool apart from the PSoC Designer software.

Figure 4 is a block diagram of the pulse generation portion of the circuit. The time base for each of the three timing circuits is selected in the Prescaler block. Table 1 shows the clock source for each of the four timing ranges. The 16-bit PWM block generates the main 50% duty cycle clock, which is sent directly to the output BNC, and which is also used to synchronize Output1 and Output2.

For every single period of the main clock, I wanted the ability to generate two additional pulse outputs, in sync





with this clock, but having independently controllable pulse widths and delays. Doing so required gating the two PSoC 16-bit counter/compare units that I had configured. Gating is performed by a 74F74 dual flip-flop (one section for each channel). The clock pulse output sets this flip-flop on its rising edge. On the falling edge of the Output1 (or Output2) pulse, a 74LS221 one-shot is triggered. The one-shot's *Q output resets the flip-flop.

I found that the 74LS221 was required for the short pulses on the nanosecond range because the simpler solution of asserting the flip-flop's *CLR input via a capacitor was unreliable in that case. Each of the three outputs feeds a 74128 inverting a 50-Ω line driver to provide active-low pulses in addition to the PSoC's active-high outputs.

Figure 5 (page 67) is a schematic of the project. Apart from the pulse generator circuitry shown in Figure 4, the only other circuitry apart from the PSoC device itself consists of the power supply and a front panel user interface. I used a 2 × 16 LCD that operates in the 4-bit mode. Cypress provides a user module for an LCD, which basically loads the necessary code to implement an LCD API. When you add the LCD user module, all you have to specify is which port it's connected to and whether or not you want bar graph functionality.

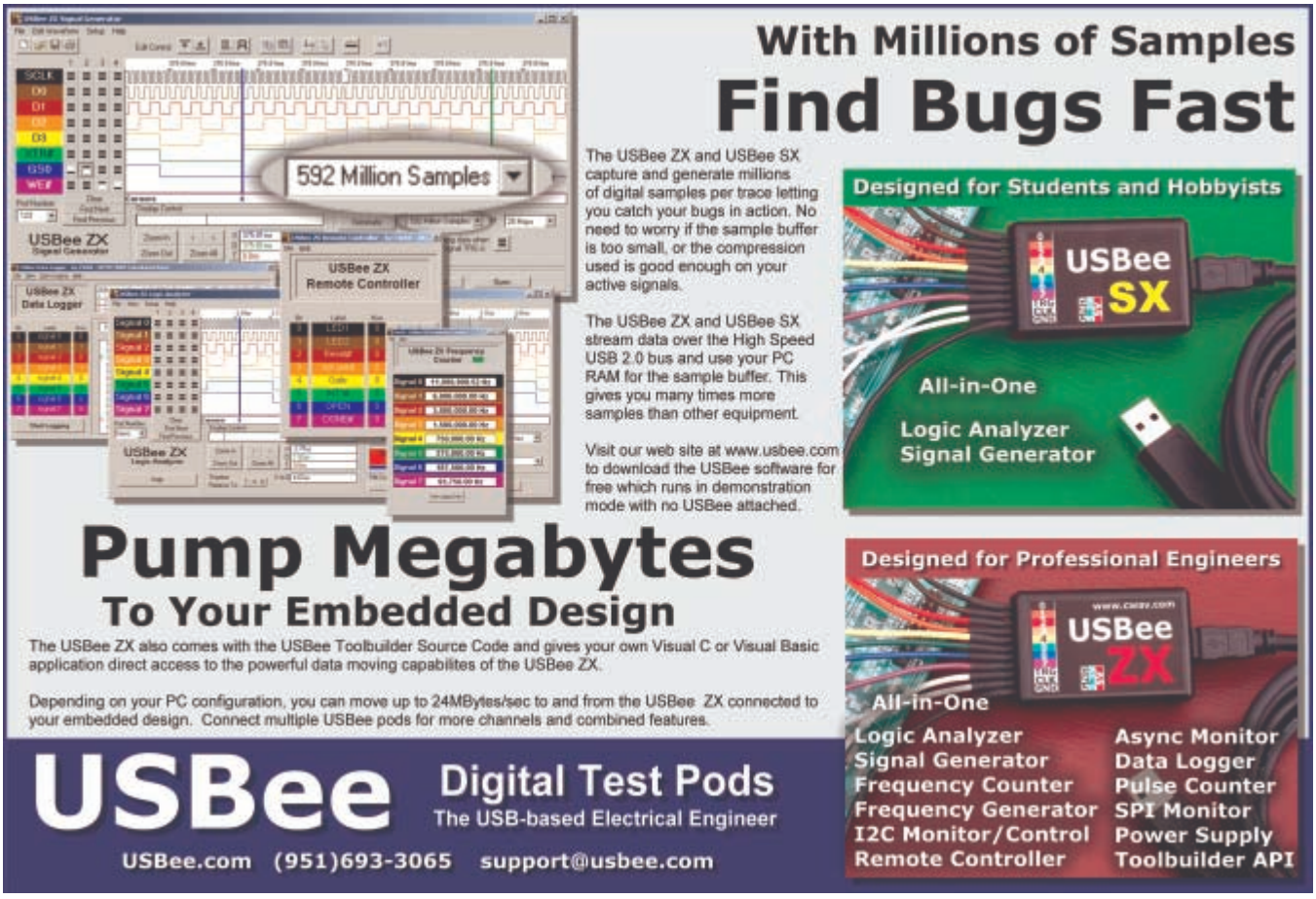
I used two Bourns ECW1D-B24-BC0024 rotary encoders for user input. These small devices are smooth and quiet. PSoC devices don't contain dedicated circuitry to decode quadrature encoders. Two PSoC digital blocks (per encoder) can be coerced into performing this function with some added software. (Refer to Edwin Olson's "Implementing Hardware Quadrature Phase Decoders.")

I didn't have any digital blocks left over, so I used the inherent "interrupt on change" function available on all of the PSoC's GPIO pins along with an ISR to handle this function entirely in software. I used pull-up resistors and debounce capacitors on the encoder lines. Note that all rotary encoders don't work the same way. You must use the specified Bourns models (or their equivalents), or the software probably won't work.

The power supply is a simple full-wave-rectified configuration followed by a 7805 three-terminal regulator. I used a center-tapped transformer and a bridge in case I decide to add bipolar analog circuitry. In that case, I'll just have to add an additional filter capacitor to provide a negative power supply.

While designing the prototype powered by an adjustable bench supply, I encountered some strange rebooting episodes until I noticed that the PSoC's built-in low-voltage monitor was kicking in below 4.8 V and resetting the unit. As soon as I made sure my bench supply's voltage was at least 5 V, the problem disappeared. In the Global Resources window, you can easily change this threshold to other voltage levels if you're using a different power supply voltage.

I used a 25-MHz crystal oscillator module for the PSoC clock to give quartz-accurate output pulses. Before adding this oscillator module, I experimentally trimmed the internal RC clock to 25 MHz by varying the value



in the 8-bit IMO_TR register. The PSoC technical reference manual tries to scare you out of doing this, but I accomplished it easily enough. However, the accuracy of the RC clock over temperature wasn't really good enough for what I needed, so I used the crystal oscillator module.

FIRMWARE

The firmware was written in assembly language, so it's pretty lean and mean. According to the map file produced by PSoC Designer, 772 bytes were to configure all of the user modules. My code used up another 2,112 bytes, which is less than 20% of the capacity of the 16-KB flash memory. Because I wasn't using the C compiler, I had none of the math routines needed to translate the various user parameters into the values required by the various counters and prescalers. Instead, I extracted various routines from several math-related application notes posted on Cypress web site.

The code consists of a main line routine as well as an ISR that looks after the process of reading the two rotary encoders. Because the rotary encoder handler is interrupt-driven, the encoders are responsive at all times.

At reset, the main line program performs the PSoC device configuration, displays a sign-on message on the LCD, and then waits for you to move the rotary encoders. Each rotary encoder has each of its two phases connected to separate GPIO pins, but only one of those GPIO pins is set to generate an interrupt on the falling edge. Whenever an encoder is moved, an interrupt occurs. The ISR first checks to see which of the two encoders was moved. Following this, it's a matter of sampling the state of the other encoder phase signal to determine whether the move was clockwise or counterclockwise. Placing the capacitors on the encoder lines cleans up most of the contact bounce noise, making the ISR routine somewhat simpler.

The encoder ISR routine sets global variable flags to indicate to the main line program whether it was the menu encoder or the value encoder that moved. Depending on which range is in effect for the parameter being adjusted as well as the setting of the Coarse/Fine switch, the actual number added or subtracted from that parameter's value will vary.

All of the ranges have resolutions comprising one or more decimal points, except the nanosecond range. Although this seems to imply that floating-point math was used, that's not the case. Everything was done using integer math, scaling, and formatting tricks for the LCD readout. If the menu selected is one of the three units menus, the ISR will also look after the process of setting the appropriate variables.

In response to changes in the flag variables brought on by the encoder ISR, the main line program will perform the necessary actions to select the proper clock source for each of the three pulse generators. It will also load in the proper values to select the necessary period, pulse width, and delay. Thus, for each click of the value encoder, the pulse generator associated with the menu selection will be reloaded immediately. Observed on an



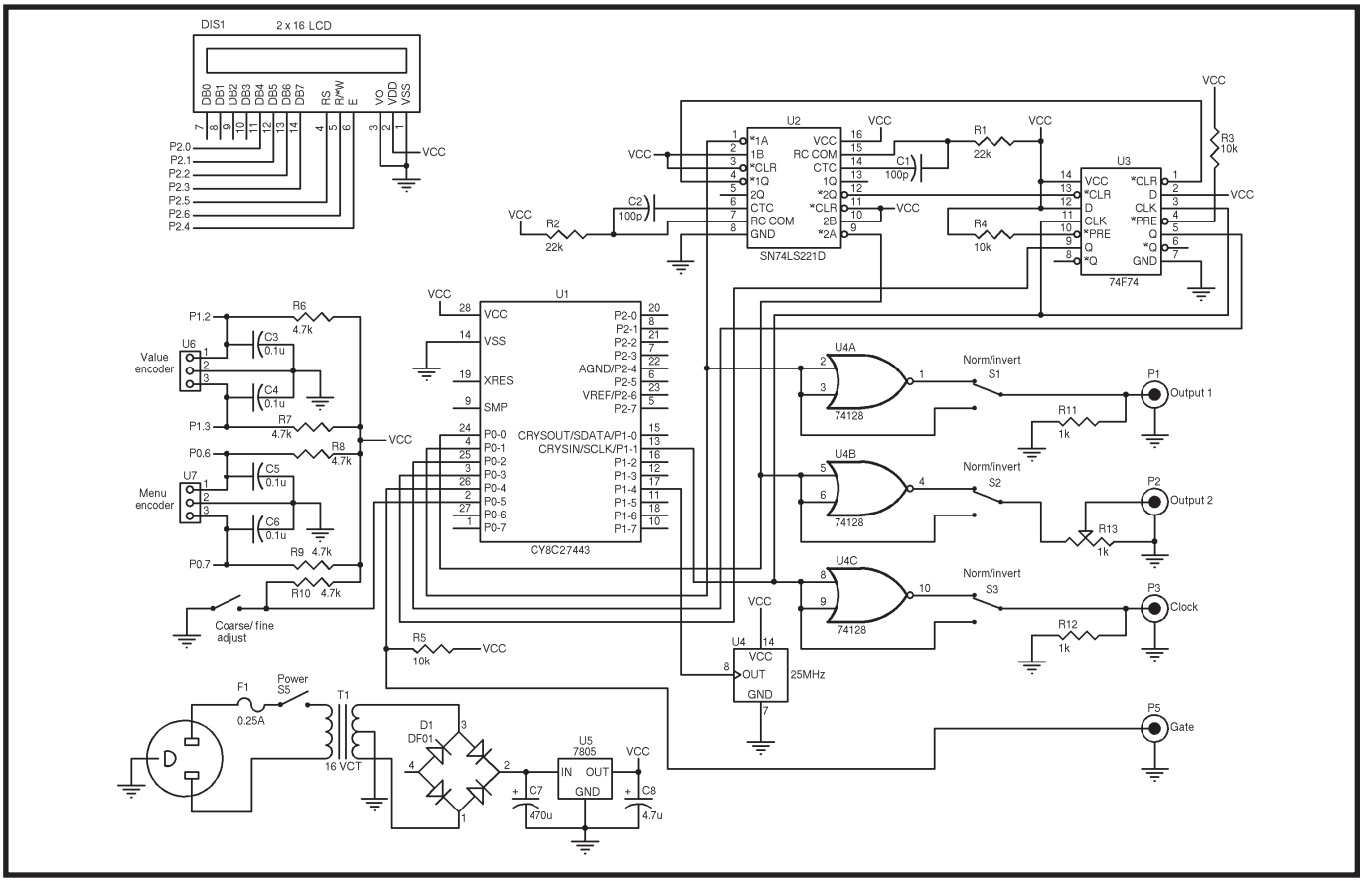


Figure 5—*Virtually all of the pulse generation circuitry resides in the PSoC device. I needed to add two TTL glue chips to allow for more versatility in pulse synchronization.*

oscilloscope, the pulse outputs will follow this fine-tuning in real time.

The LCD is updated to reflect changes. The first line displays the parameter name. The second line shows its current value, the time units, and an indication of the Coarse/Fine switch's setting.

Because I included the delay function for both Output1 and Output2, a bit of a complication occurs if I set a delay of zero. This happens because of the way that the timer blocks are built; they can't count a zero. Given the design's high resolution, entering the smallest possible delay value results in practically the same thing as no delay at all (in most cases).

The only other trade-off was the clock output pulse's duty cycle. The actual duty cycle is half the value loaded into the PWM16 block's period register. If the period requires an odd number to be loaded in the register, then the integer division by two will result in a small error in the 50% duty cycle. Because there's such an overlap in the different ranges, you can always go to a range where the period number is so large that the integer division error is negligible.

THE TRAILING EDGE

Well, that's about all there is to say about this project. Although this particular PSoC family was new to me, the development process went smoothly. This is attributable in large part to Cypress's excellent PSoC documentation, as well as the smooth operation of the PSoC Designer development software.

The tiny Invention Board programmer from the PSoC High Integration Challenge 2004 worked well in developing this project. But now that I expect to use the PSoC in upcoming designs, I bought a PSoC ICE-Cube real-time in-circuit emulator for development purposes.

Brian Millier is an instrumentation engineer in the Chemistry Department at Dalhousie University in Halifax, Canada. He also runs Computer Interface Consultants. You may reach him at brian.millier@dal.ca.

PROJECT FILES

To download the code, go to ftp.circuitcellar.com/pub/Circuit_Cellar/2005/181.

RESOURCES

H. W. Cintra, "Math Programs," AN2113, Cypress, 2004.

M. Ganesh Raaja, "Binary to BCD Conversion," AN2112, Cypress, 2003.

E. Olson, "Implementing Hardware Quadrature Phase Decoders," AN2145, Cypress Microsystems, 2003.

SOURCES

ECW1D-B24-BC0024 rotary encoders
Bourns
www.bourns.com

CY8C27443 PSoC and PSoC Designer IDE
Cypress Microsystems
www.cypress.com

C Compiler
ImageCraft
www.imagecraft.com



Reach for Interface Control

Add a GUI to a ZF8042A-Based Design

Fred needed to add a GUI to a ZF8042A-based application, so he turned to Reach Technology's SLCD controller. The controller offloads GUI support tasks and communicates with the ZF8042A by way of a high-speed serial connection.

Reaching a breathing human being on the other end of a technical support line can be difficult. And if you succeed and actually get through to a real person, you'll probably end up with egg on your face because he'll point out a ridiculously obvious solution to your problem.

Egg surely would have been on my face last week had I broken down and called Zilog for help with something I was stuck on. Fortunately, I bit the bullet and worked through the problem on my own. That sure felt good. Plus, at the end of the day, I had attached a touch-enabled GUI to an eight-pin Zilog ZF8042A microcontroller.

ZILOG TWILIGHT ZONE

Normally, development kits are well tested and the demo code has gone around the block before it's released with the hardware. We've even opened some Zilog development kit packages together in the past. We never had any problems. Zilog development kit hardware is a real bargain. Zilog usually throws in a C compiler and a fancy gadget to program and debug the target microcontroller silicon. Well, all of that held true with the latest Zilog development kit as well.

Take a close look at Photo 1. There aren't enough components on that board to create a type of problem that I can't solve with solder. This little puppy is roped off like an art exhibit. In the upper-right corner, there's a classic power supply piece based on an LT1086CT-3.3 voltage regulator. Just below the voltage regulator is a Sipex-driven RS-232 port complete with its four charge pump capacitors but without the required RS-232 IC power rail bypass capacitor.

Moving farther south is a 74LVC04 inverter that's used to supply only jumper-selected logic levels to enable or disable the development board's RS-232 or IR devices using J1 and J2, respectively. The single-line debug connector, the Reset switch, and associated circuitry are also located in this part of the exhibit.

Moving west across JP2, you find yourself in the IR and microcontroller halls. The Test push button is conveniently positioned in the southernmost quadrant of the microcontroller domain along with the IR transceiver. Both the push button switch and the IR transceiver are wired into the I/O subsystem of the eight-pin ZF8042A microcontroller, which is in the northwestern quadrant of the development board.

The ZF8042A I/O jumpers are in a debug configuration. This enables the microcontroller's I/O pins to interface with the D3 and D4 LEDs. The PA0 line that would normally drive D2 is connected to the DEBUG connector pin at P3. The ZF8042A is running in a totally self-contained state because the Y1 crystal/resonator position isn't populated. It doesn't get much more basic than this.

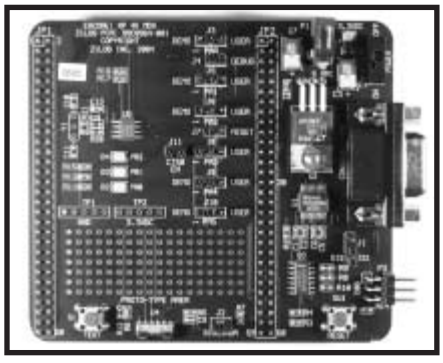


Photo 1—*This is like the old Where's Waldo? game. Can you find the ZF8042A? There isn't much here, and that's the point.*

Confident that I would get this all up and going quickly, I loaded the Zilog ZDSII IDE and the integrated Zilog C compiler package onto the workhorse Florida room PC. There seemed to be no problems. Carefully following the quick start notes that came with the package, I attached the USB Smart Cable you see in Photo 2 to the debug connector of the Zilog ZF8042A development board.

After connecting the USB Smart Cable to an open USB port on the PC, I received the expected "Found New Hardware" balloon and proceeded to install the USB driver for the USB Smart Cable per the quick start instructions. Everything seemed to be fine at this point. I successfully installed the ZDSII IDE and attached the program/debug dongle. After installing the USB Smart Cable, I fired up the ZF8042A in Demo mode. Sure enough, the LEDs danced as advertised.

Next, using the I/O pin jumpers, I put the ZF8042A development board into debug configuration. I loaded up the ZDSII IDE and pulled in the sample code for the demo application. I tried for hours to get a good compile and link without success. A compiler warning stating that an object file was missing kept appearing. The bad news was that this assembler file (zsldevinit.asm) with the missing

object file is crucial to operation because it configures all of the ZF8042A's I/O. OK. Now it was time to read more carefully.

I started with the README.TXT file and finally ended up with what I thought was an answer to my problem. The text file mentioned that I must make sure Generate Object was checked in the Assembler pane of the Project Settings window. I had checked that several times, so I already knew before I got there that it was checked off. But I went there anyway.

A few hours later, I was still searching for answers. My hands and feet were moving toward the telephone to get some technical support because I wasn't about to wait 24 h for an answer via e-mail. In the meantime, I decided to run the demo program and monitor the ZF8042A development board's serial port to make sure it was working as designed. I opened up a TeraTerm Pro emulator window and set the data rate for 57,600 bps on COM1. I then attached a one-to-one serial cable between my PC's serial port and the ZF8042A. Following this, I fired up the ZF8042A in Demo mode and got garbage in the TeraTerm Pro emulator window. How could this be? Zilog surely wouldn't have put out something that doesn't work.

After fiddling around with the TeraTerm Pro serial port speeds, I eventually got some readable messages at 38,400 bps. Whoa! I'd seen this kind of thing before. Rebooting had normally set things straight. Following the reboot, I still got garbage at 57,600 bps and good text at 38,400 bps, so I checked the demo source code, which states that 57,600 bps is the correct speed. I then recalculated the demo

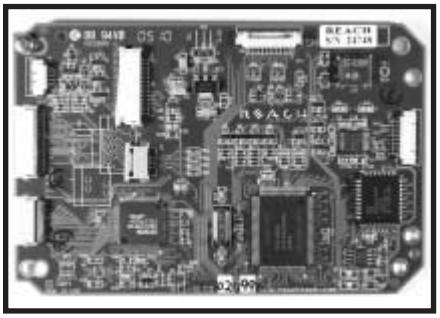


Photo 3—*It's all here. All you have to do is know how to talk to it to get what you want.*



Photo 2—*Yes. I opened it up. There's a Xilinx part and a Zilog part in there. I thought I had blown this baby up a couple of times but it hung in there.*

code's data rate generator values and found them to be set for 57,600 bps.

I was pretty fed up at that point, so I decided to try one more thing before calling Zilog for help. With that, I attached the ZF8042A's serial port output to my PC that runs Frontline's Serialtest Asynch RS-232 monitor program. I dialed in 57,600 bps on the Frontline control panel and once again fired up the ZF8042A development board. The laws of physics hadn't been altered. The Frontline Serialtest Asynch capture window displayed readable messages at 57,600 bps just like the Zilog documentation and source code said it should.

OK. Something was wrong with the Win2K Professional PC onto which I had loaded the ZDSII IDE and Zilog C compiler. So, I moved the installation lock, stock, and barrel to a laptop running Microsoft XP Professional. Whamo! The demo project compiled without error, and the USB Smart Cable was able to communicate with the ZF8042A development board and perform download and debug operations.

Zilog's ZF8042A tools are easy to use if your PC works correctly. Now I was able move on and open up a couple of more boxes that were perched on the Florida room's bench. Best of all, I didn't embarrass myself by calling Zilog's technical support hotline. The next step was stripping and reloading my troublesome PC.

LOOKING GLASS

Let's say you want to use the little ZF8042A in an application that requires a GUI. Don't laugh. The ZF8042A is a capable little eight-pin device with 4 KB of program flash memory, 1 KB of SRAM, a UART (yes, a full-duplex UART), an ADC, a couple of 16-bit timers with PWM capability, a comparator, an internal oscillator, and 20-MHz external clock capability. Rest assured that 4 KB of program flash memory won't let you load and run a native ZF8042A application based on Linux, UNIX, Windows CE, or headless Windows XP. You're also going to come to the conclusion that you can't stuff enough GUI driver code into that 4 KB of flash memory and still have enough space for your little I/O application. (You'll have four I/O pins left if you use the UART.) If you're skeptical or just have deep pockets, you can stop reading now and convert to one of the aforementioned platforms. Otherwise, let me help you put a professional GUI in front of the ZF8042A.

I started with a Hitachi SX14Q001-ZZA color STN 5.7″ display with an integrated touch screen and a Hitachi TX14D11VM1CAA-1 touch-enabled color TFT 5.7″ display. A Reach Technology SLCD controller, a PowerCom adapter, a backlight inverter, and a wall wart support both displays. The SLCD controller is shown in Photo 3.

The key to successfully using a GUI with an MCU as small as the ZF8042A is the SLCD controller's ability to offload the entire GUI support task and communicate with the ZF8042A via a high-speed serial connection. The SLCD controller consists of a flash memory-based microcontroller, an external flash memory area that is separate from the MCU's flash memory, an RS-232 serial port, and all of the power and logic necessary to drive either a passive-matrix color super-twist Nematic (CSTN) or an active-matrix TFT thin film transistor (TFT) display panel. The contrasting attributes of each display technology are shown in Photo 4.



Photo 4—*The CSTN display is the top display panel. You can really see the difference in the display panel technologies.*

The SLCD controller is controlled via commands issued from a serially connected host (the ZF8042A). Bitmaps and command macro structures, which can be downloaded via either a Windows application or the host, are held in the controller's external flash memory area. The Windows bitmap/macro load application (BMPload.exe) is part of Reach's support package. Reach's application notes describe how to reload the external flash memory image from a host microcontroller.

Custom bitmaps are easily realized if you use Photoshop or the Windows Paint program. SLCD macros are just as easily created as they are written in the form of standard C defines. The SLCD controller comes preloaded with some pretty nifty macros and bitmaps.

Now I'll describe how I pulled the SCLD controller's demo jumper and hooked up the ZF8042A. As you'll see, it's extremely easy to add the professional GUI to the ZF8042A using the TX14D11VM1CAA-1 display and the SLCD Controller.

REACH FOR THE DISPLAY

With the ZF8042A development board running at 57,600 bps, the SLCD controller wanted to see a serial port speed of 115,200 bps, so I modified the ZF8042A BRG divisor value to get the ZF8042A UART up to speed. In my initial ZF8042A daze, I calculated the ZF8042A's BRG divisor value to be six for 57,600. A quick check of the data rate tables in the ZF8042A's datasheet told me that three is the magic BRG divisor value for 115,200 bps with a 5-MHz clock. That was an easy modification. All I had to do was change the default data rate number in the SIO.H file. I also could have changed the SLCD controller's data rate with a macro (see Listing 1).

The C comment operators in the SLCD macro enable the SLCD macro to coexist within a C program as an include file. After I executed the macro in this fashion (`*PONMAC 1<return>`), an SLCD controller power cycle brought the SLCD's data rate down to match the ZF8042A development board's default data rate (57,600 bps). After modifying the ZF8042A development board's source code, I threw the ZF8042A development board onto the Frontline Serialtest Async PC and verified that I could read the LED messages emanating from the ZF8042A development board at the faster 115,200 bps.

I identified a Reach-created macro (`keypad_demo 6`) for putting a keypad on the Hitachi TFT display. The `keypad_demo 6` macro is called by send-

Listing 1—*Because the ZF8042A can fly with the eagles at 115,200 bps, I didn't want to degrade the speed of the SLCD controller's serial interface. Plus it's easier to change the ZF8042A's data rate.*

```
//Changing the SLCD data rate with a macro
#define pon_mac 1
//(Start comment out contents...)
//Set data rate
baud 57600
#end
**********************************************************************
#define number_pad 2 /*
//Bitmap that holds displayed number
xi 12 0 0
//Set font 24 for buttons
f 24
//Define buttons
bd 1 0 32 1 "1" 9 5 10 11
bd 2 32 32 1 "2" 9 5 10 11
bd 3 64 32 1 "3" 9 5 10 11
bd 4 0 64 1 "4" 9 5 10 11
bd 5 32 64 1 "5" 9 5 10 11
bd 6 64 64 1 "6" 9 5 10 11
bd 7 0 96 1 "7" 9 5 10 11
bd 8 32 96 1 "8" 9 5 10 11
bd 9 64 96 1 "9" 9 5 10 11
bd 10 0 128 1 "*" 9 5 10 11
bd 0 32 128 1 "0" 9 5 10 11
bd 11 64 128 1 "#" 9 5 10 11
//Tell user we're done
beep 10
//A comment here only works if C compiler handles nested comments
#end
...(end comment out contents) */
//Macro to set font and color for input box display. Foreground color
//is XOR of background color to match the buttons.
#define grey24 3 /*
S 333 CCC
f 24
#end  */
//These macros write text to the screen
#define button_text 4 /*
m3
t "`0`" 10 4
#end  */
//This macro enables the buttons specified in number_pad macro to
//write text to the screen inside a bitmap.
#define attach_buttons 5 /*
xa 0 p 4 0
xa 1 p 4 1
xa 2 p 4 2
xa 3 p 4 3
xa 4 p 4 4
xa 5 p 4 5
xa 6 p 4 6
xa 7 p 4 7
xa 8 p 4 8
xa 9 p 4 9
#end  */
//This macro creates a keypad and displays the key using previous
//macros.
#define keypad_demo 6 /*
s 0 1
m2
m5
#end  */
```

RoboNexus

IDG WORLD EXPO

roboticstrends

Founding Sponsor

iRobot

The Largest Robotics Event in the Western Hemisphere

October 6-9, 2005
San Jose Convention Center, San Jose, CA

EXPO Magazine's best new show 2005 Winner






The International Business Development, Educational and Consumer Event for Personal, Service and Mobile Robotics

- 50 Robotics Visionaries and Thought Leaders
- 50,000 Square Foot Exposition
- 5 Comprehensive Professional Development Conferences:
  - Business Development and Opportunities Conference
  - Emerging Robotics Technologies and Applications Conference
  - Robotics Design, Development and Standards Conference
  - Robotics Education and Instruction Conference
  - Consumer Robotics and Entertainment Event

New for RoboNexus 2005!

- Business-to-Business and Consumer Entertainment Expo Areas
- Service Robotics Summit
- Service Robotics Pavilion
- "Robotics at Home" Demo Stage
- Robotics Innovators Awards –the 'Robi'

www.robonexus.com

Premier Sponsor

Premier Media Sponsor

POPULAR science

Gold Sponsors

HiTEC

Corporate Sponsors

evolution robotics

ESS

LEGO education

mBus machineBus.com

OLogic

Openware Robotics

ShopBot

ROGUE robotics

SYSTRONIX
Embedded Java Spoken Here

Media Sponsors

BATTERY POWER PRODUCTS & TECHNOLOGY

CIRCUIT CELLAR

E-DRIVE

Entertainment Engineering

VEHICLE

Nuts & Volts

GLOBAL FUTURES

PC WORLD

Popular Mechanics

ROBOTBOOKS.com

SERVO

Association Sponsors

GETI

ing m6 followed by a carriage return over the ZF8042A-to-SLCD controller serial communications pipe.

SLCD macros are powerful, but they're relatively easy to understand and use. Let's tear apart the macro called keypad_demo 6 included in Listing 1.

The first line of macro code (s 0 1) sets the foreground (black-0) and background (white-1) colors. The second macro statement (m2) invokes the number_pad 2 macro, which draws the keypad number holding area in bitmap 12 at the x = 0 and y = 0 coordinates (xi 12 0 0) just above the key buttons, which are rendered with a font of 24 (f 24). The bd command within macro 2 also commands the SLCD controller to return x1 when the 1 key is pressed on the keypad. x2 returns when the 2 key is pressed.

Each keypad button is drawn separately with the bd command. The first argument of the bd command is the button number. The next two arguments define the upper left corner of the button. All of the buttons are defined as standard with the next argument (1) and in the unpressed state display as bitmap 10. With the buttons defined as standard, the host is notified after every button depression; it isn't notified when the button is released. When depressed, all of the buttons display as bitmap 11. The numbers and symbols contained with the quotation marks are what are written on each button. The two arguments that follow the button text define the location of the text within the button relative to the upper left-hand corner of the button.

When the m2 macro finishes, the m5 is called. The commands contained within the m5 macro attach each button depression (p) to macro 4 and pass the last argument to macro 4 to ultimately be displayed in the keypad number holding area by macro 3. After I knew what the m6 macro would do when invoked, I wrote some ZF8042A code to wake it up.

Listing 2 is a true code snippet. All of the underlying functionality isn't shown. The idea was simply to communicate with the SLCD controller in a useful manner. The SLCD code really begins at the do loop. A carriage return character was sent to the SLCD, which was displaying its banner at this point in time. After the carriage return was issued, the SLCD responded with a prompt character (>). The z command was then issued by the ZF8042A and the display is cleared.

The execute m6 macro command was issued followed by another carriage return. At that point, the keypad number holding area and the keypad appeared on the display. The idea here was to switch the red and yellow LEDs on the ZF8042A development board on and off by touching the 1 and 2 buttons on the keypad.

The turn_off_led function darkened both of the LEDs under the ZF8042A's control. I knew that either x1 or x2 would be returned to the ZF8042A when either button 1 or 2 was depressed.

Let's assume for now that the 1 button was depressed and x1 was returned via the serial connection. The switch statement flowed into the 0x31 case and both LEDs were extinguished. The red LED was then illuminated, and two strings were sent to the SLCD.

Recall that the xi command displays a bitmap. In this case, bitmap 27 happened to be an illuminated LED and bitmap 26 was an extinguished LED. The 0,0 coordinates defined the absolute upper left of the display. Therefore, the red LED bitmap, illuminated by depressing button 1, was to the left of the yellow LED bitmap, illuminated by button 2, on the display.

Listing 2—*I haven't shown all of the preliminary port bashing here. The focus is on the code that communicates with the SLCD.*

```
char incoming;
main ()
{
   init_systemclock();            // Initialize system clock source
   init_led_gpio();               // Initializes LED ports (ports A and C)
   init_test_button_gpio();       // Initialize Test button (port C)
   DI();                          // Disable interrupts
   init_timer0();                 // Intialize Timer0
   init_uart0();                  // Intialize UART
   init_test_button();            // Initialize Test button
   EI();                          // Enable interrupts
   do{
      puts("\r");
      incoming = getch();
      }while(incoming != '>');    // Wait for > prompt from SLCD
   puts("z\r");                   // Clear the screen
   do{
      incoming = getch();
      }while(incoming != '>');    // Wait for > prompt from SLCD
   puts("m6\r");                  // Display keypad
   turn_off_led();                // Turns off all LEDs
   while(1){
   incoming = getch();
   if(incoming == 'x')            // Look for incoming ASCII "x"
      {
      incoming = getch();         // Get button number (x1 or x2)
      switch(incoming)
         {
         case 0x31:               // Button 1 on the keypad
         turn_off_led();          // Turns off all LEDs
         PAOUT &= 0xFB;           // Turn on red LED
         puts("xi 27 128 128\r"); // Show 1 on LED bitmap
         puts("xi 26 250 128\r"); // Show 2 off LED bitmap
         break;
         case 0x32:               // Button 2 on the keypad
         turn_off_led();          // Turns off all LEDs
         PAOUT &= 0xFD;           // Turn on yellow LED
         puts("xi 27 250 128\r"); // Show 2 on LED bitmap
         puts("xi 26 128 128");   // Show 1 off LED bitmap
         break;
            }
         }
      }
}                                 // End of main program
```

If the code were to drift into the 0x32 case, the LED bitmap situation that occurred in the 0x31 case would be inverted. I obtained the results shown in Photo 5 by touching button 2.

THAT'S IT?

Yep. That's all there is to it. With a minimal amount of effort you can attach a touch-enabled GUI to a tiny eight-pin Zilog microcontroller and perform some tangible operations with LEDs on the display and on the ZF8042A development board.

I'm sure your mind is going crazy with ideas about how you can implement both the ZF8042A and the Reach SLCD controller in one of your projects. I manipulated some LEDs, but I'm sure you can substitute your own ZF8042A I/O endpoints into the scheme.

The message here is a simple one: With the Reach SLCD controller and a display panel, you can easily fabricate a professional touch-enabled, GUI-based product—irrespective of the size of your MCU—without having to resort to PC-bound languages and

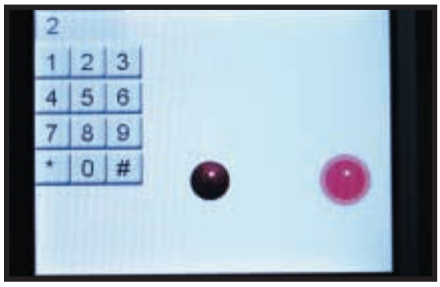


Photo 5—*No muss. No fuss. Just GUI. Sorry, I didn't have a preloaded yellow LED. Imagine a tiny ZF8042A behind all of this.*

expensive hardware platforms.

If price is an issue, good news is on the way. I have it from credible sources that the price point for the more expensive Reach TFT SLCD development kit will be reduced. That's great news for those of you who may need the higher contrast and quicker response times offered by TFT display technology.

Reach's SLCD controller reaffirms that in the embedded world, microcontrollers don't have to be hefty to be represented by a TFT display panel. Furthermore, programming the display panel doesn't have to be complicated with expensive hardware and complex operating systems.

Fred Eady has more than 20 years of 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.

PROJECT FILES

To download the code, go to ftp.circuitcellar.com/pub/Circuit_Cellar/2005/181.

SOURCES

SX14Q001-ZZA LCD and TX14D11VM1CAA-1 LCD
Hitachi
www.hitachi.com

SLCD Controller
REACH Technology
www.reachtech.com

ZF8042A Development kit
Zilog
www.zilog.com



Embedded Systems Conference: September 12 – 15, 2005
Embedded Security Seminar: September 12 – 14, 2005
Exhbition: September 13 – 14, 2005
www.embedded.com/boston
Hynes Convention Center • Boston, MA

Embedded Systems Conference Boston

Co-located with: Embedded Security seminar

We know it's hard to leave your desk. In today's competitive environment, you can't afford not to.

Take the time and we'll deliver what you need:

> The embedded design experts you need to hear from: **Ganssle, Saks, Labrosse,** and **41 other industry experts**

> The vital topics you need to stay ahead of competitors: **real-time software, board design, wireless security,** and other design topics in **92 classes**

> The industry leaders who provide the solutions you need: **Atmel, Green Hills Software, IBM Rational Software, Intel, Mentor Graphics/Accelerated Technology, STMicroelectronics, Xilinx** and other leading embedded vendors

> The chance to exchange ideas with your industry peers from leading OEM's such as **Boeing, Hewlett-Packard, Motorola, Raytheon,** and many more

Exhibiting opportunities available.
Please contact Steve Corrick at
scorrick@cmp.com or 415-947-6651

Register by August 16th to save up to $645!
embedded.com/boston
Priority Code U113

CMP

Media Sponsors Embedded Systems Programming EETIMES



Analog Tips and Tricks

Are you looking for ways to improve your analog and RF circuitry? Take a few tips from Ed to get started. In this column he shows you how easy it is to take your PCB wiring skills to the next level. Who knows, your digital projects just may improve too.

Circuit Cellar has always attracted readers who enjoy building gizmos, both at work and for their own use. My December 2004 column, "Building Boxes," prompted enough comments and suggestions regarding additional techniques that I decided a follow-up was in order.

Although these tricks are designed to improve your analog and RF circuitry, even your digital projects will benefit, because digital is just analog with the gain cranked way up. You're sure to find at least one technique that will make your next project work better.

BUILDING BOARDS

I wire most of my projects on PCBs built in my basement shop, using a process that produces both circuit documentation and reasonably high-quality hardware without too much effort. I've come up with some tricks that should help you get good results too.

I use CadSoft's EAGLE schematic capture and board layout software, which runs on Windows, Linux, and Mac OS X (www.cadsoftusa.com). The free version can handle most of the circuits in this column, and the Standard version is reasonably priced. EAGLE is perfectly stable on my SuSE Linux 9.2 desktop system.

The board layout program can produce output files in nearly any format, including the Gerber files used in board production shops. I save the output for each layer as a Postscript file, and then import the files into the GNU Image Manipulation Program (GIMP) image-editing program at 600 dpi.

The top image in Photo 1 shows the copper plane pattern for the charge-pump LED power supply I described in my April 2005 column. I panelize them with the GIMP to produce a single image with multiple patterns in a rectangular grid. Because all this happens digitally, there's no loss of resolution and no smudges. I then print the image through an HP LaserJet 1200 on a sheet of toner-transfer film from either Pulsar (www.pulsar.gs) or Techniks (www.techniks.com). It turns out that toner contains a thermoplastic that both adheres to bare copper and resists the etching chemical solution.

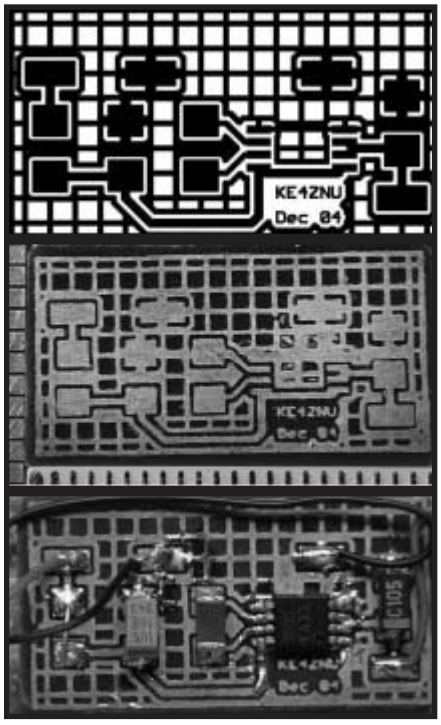


Photo 1—*The top image is the top copper layer from an EAGLE board design. The bare board shows several flaws, but the one on the bottom came out fine. The ruler scales are 0.050″ vertically and 1 mm horizontally. The board has extremely small features!*

Because most of my boards are extremely small, they don't fill a complete sheet of the toner-transfer film even after I panelize them. I print a sheet of paper, tape a square of film that's approximately 1″ larger than the patterns atop them, and then run the paper through the printer again. The adhesive on cheaper tapes tends to melt at laser printer temperatures, so use good tape and monitor your results. Put a single strip on the leading edge of the toner-transfer film to allow the paper and film to shift slightly as they pass through the fuser rollers.

Although I've tried applying the film to the circuit board using a clothes iron, I eventually bought a modified laminator that provides uniform pressure and regulated heat across the width of the film. The results aren't always perfect, as evidenced by the bare board in Photo 1. However, the finest lines on that board measure 10 mils, with approximately 10-mil spacing between IC leads, and the 3-mil strokes in the letters fall well below the recommended 6-mil width: these dimensions are aggressive for the toner-transfer technique. Most of my larger boards came out perfect. You'll have no trouble with through-hole components and 50-mil traces.

I have produced double-sided boards using toner-transfer films, but registering the two patterns presents a challenge. I accurately align the films face-to-face on a light table, tape them together along one edge, align the blank board between the films, feed the taped edge into the laminator, and hold everything together until the input rollers grab the board. This is good to 10 mils and requires pads and vias that are somewhat larger than normal .

The ink in ordinary Sharpie fine-point permanent markers is surprisingly resistant to copper etchant, so I use it to touch up small flaws. I preserve broad areas of copper around the edges of the panelized images with package-sealing tape.

Ferric chloride warmed to approximately 160°F on a hotplate etches 1-oz. (1.4 mil) copper in about 15 min. I hand-agitate the boards in a shallow glass tray, wearing my shop apron, latex gloves, and splash goggles.

I plate a 50-microinch layer of silver atop the copper, which both improves solderability and makes board photographs look better. A 4-oz. bottle of silver plating powder from Cool-Amp (www.cool-amp.com) could become a family heirloom passing from generation to generation.

Until recently, I had been drilling holes using carbide bits in a Dremel tool drill press, but now I'm using a Sherline CNC milling machine. I wrote an EAGLE user language program (ULP) that converts the hole information in a board layout to a G-code file for the milling machine. The ULP sorts the holes by drill diameter and then visits each hole in nearest-neighbor order. This crude solution to the traveling salesman problem works well enough for my small boards.

Laser printers are digital devices, but the paper and film media carrying the circuit patterns remain analog. I don't have hard numbers, but I have found that the CNC mill puts holes where they should be, which may not quite match up with where the copper patterns actually are after they go through the toner-transfer process.

The problem arises because the sheets travel through the printer with slight speed variations, they expand and contract at temperature extremes, and they squish under laminator pressure. Boards less than a few inches in each dimension, which constitute most of my projects, have fewer troubles. Run several test pieces before you commit to fine-line, large-area, double-sided boards!

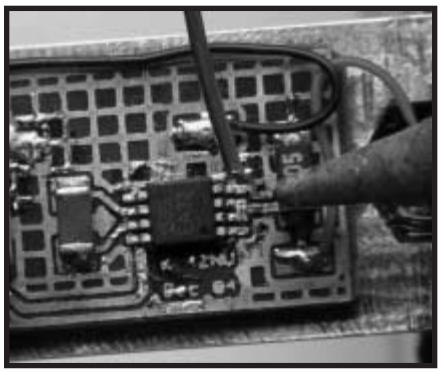


Photo 2—*Hand-soldering SMD parts is possible if you learn some new techniques. First of all, the board must be horizontal so the parts don't slide off!*

SOLDERING PARTS

Circuit board assembly lines use solder paste and heated ovens to mass-solder surface-mount parts over the entire board at once. The process requires considerable calibration to achieve solid joints while not overheating the semiconductors, so it isn't amenable to the production of one-off prototypes.

Photo 2 shows the relative sizes of 30-mil solder and a fine-point soldering iron compared to the board in Photo 1. I wear a magnifying headband and place the board flat on my bench, using needle-pointed forceps to maneuver the parts.

After I apply liquid flux to the board pads, I use a piece of Soderwick (that's right, no "l") as a sponge to put a thin layer of solder on them. Then I tack diagonally opposite leads of each IC to the board and solder the remaining leads, all without adding more solder. This doesn't produce a classic solder fillet, but I avoid having to join all the leads in a solder blob.

I tack one side of larger SMD components (such as the capacitors shown in Photo 2) to the tinned board, solder the other lead, and then put a fillet on the first lead.

Analog and high-speed digital circuits benefit from a low-impedance ground plane. The grid in Photos 1 and 2 is a compromise between an ideal solid-copper plane and something solderable.

Many of my SMD boards have numerous small vias between the top-surface ground areas and the unetched bottom plane, with C- or Z-shaped pieces of hookup wire substituting for plated-through vias. I solder brass shim stock along the edges for a low-impedance connection between the layers.

I've been using 4%/96% silver-tin solder rather than the familiar 60%/40% tin-lead alloy for my electronic projects. The silver alloy melts at 440°F, about 70° higher than tin-lead alloy, and a temperature-controlled iron is pretty much required for good joints.

A tip for stabilizing your soldering iron: drape the cord around your arm so your forearm supports the weight of the cord. I reach through the cord loop, flip the cord over my arm, and then pick up the iron as usual. Try it. The iron will float in your hand.

Glue a pair of mouse pads face-to-face to get a convenient work surface that won't slide around as you install the components. The black background also helps reduce glare.

BUILDING BOXES

I use JB Weld epoxy for both bond-



Photo 3—*This section of PS/2 keyboard cable shows the insulated conductor molded outside the shielding foil. A keyboard wouldn't notice, but a microphone might misbehave.*

ing small parts and filling unwanted holes on boxes. Jon Titus recommends Bondo's Glazing & Spot Putty to eliminate minute cracks, bubbles, and pits like the ones remaining after I bond a plate in a hole (www.bondo-online.com). It's available in hardware and auto body supply stores.

Rick Mainhart builds boxes with a fillet of solder joining the copper foil on the inside edges. I used Gorilla Glue and Loctite Epoxy Putty for my headset box because of the tight clearances around the connectors, plus a certain aversion to cooking my fingertips. Hard silver solder would work better than soft lead solder in this application.

Rick solders a layer of copper foil, burnished and fluxed, to strengthen and smooth outside edges. He gets rolls of copper foil tape from craft stores that normally sell it to stained glass artists. The stores around here don't carry it, but a quick Internet search turned up many suppliers. Some foil tapes have adhesive backings, which you definitely don't want.

I have used brass shim stock that's available from machine tool suppliers in thicknesses starting at 1 mil. Brass isn't as conductive as copper, but that's not critical in this application because of the large surface area. I've bought shim stock from suppliers including Enco (www.use-enco.com) and Victor Machinery Exchange (www.victornet.com).

Home repair stores like Lowe's and The Home Depot carry 20- to 30-mil copper roof flashing. That's far too thick for circuit board boxes, but it can be useful for high-current or low-impedance conductors. You saw a piece of it as a ground strap inside the attenuators I described in "Robot Mechanics" (*Circuit Cellar*, 167, June 2004). Some flashing has asphalt-saturated paper backing that's intended to galvanically isolate the copper from aluminum or steel structural elements, which makes it useless for our purposes.

Micro-Mark carries precision brass angle stock in miniature sizes that work well for interior corners where you need high strength: solder a right-angle bracket to the copper foil inside the joint. This requires careful fluxing and far more heat than a standard pencil iron can produce (www.micromark.com).

An abrasive cut-off wheel in a Dremel tool slices thin angle brackets that are too delicate for a hacksaw. An ordinary scissors works well for brass shim stock up to approximately 10 mil in thickness.

POWER ON!

The ubiquitous wall wart power supply has become the scourge of modern life. Every device requires a slightly different combination of voltage, current, and charger intelligence. Worse, no two warts can share adjoining outlets. If you don't already know about power strip Liberators, go to www.cyberguys.com and search for "Liberator."

Matthew Karas suggests using PC cases to hold larger projects. They have plenty of internal room, good structural qualities, great EMI shielding, and a "free" power supply. However, remember that most PC-class switching supplies have a minimum load requirement for at least one output. If your circuitry doesn't draw that much current, you must add ballast resistors that will get hot; heatsink them to the case.

Nearly everyone selects standard battery holders for smaller projects. Most of these holders use nickel-plated, spring-steel wire to contact the cells, which works well for low currents, but devices with high-current motors or solenoids will see a voltage drop at the holder's output terminals. Even if you run separate supply wires to the battery holder, pulses from high-current devices can feed back into your logic or analog circuitry.

Ken Boone recommends adding unfluxed Soderwick (again, without the "l") to the holder's contacts, folding it over the ends of the springs so that the copper braid contacts the cells and carries most of the current in parallel with the springs. This wouldn't work in production, but for prototypes and one-off projects, it could make the difference between success and failure. Ken should know. He routinely places at or near the top in the Trinity College Fire-Fighting Home Robot contest.

ODDS & ENDS

My enthusiasm for soldering connectors burned out long ago, so I now adapt standard cables whenever possible. Even if you have a standard cable designed for something completely different, it's probably better than anything you could build yourself.

For example, the microphone preamp I described four years ago connects to my bike helmet through a PS/2-style keyboard cable ("Mic Check: A Communication System for Cyclists," *Circuit Cellar*, 133, August 2001). These work extremely well for about a year's worth of riding, then the wiring fails at the helmet despite my best efforts at stress relief. Keyboard cables simply aren't designed with continual flexing in mind, but I haven't found a more durable cable that's both lightweight and inexpensive.

The last time I cut a new cable, I found the situation in Photo 3. The foil shield should have enclosed all six conductors and the drain wire to help reduce EMI. The orange wire somehow made its way out of the foil and was molded into the outer insulation along the entire length of the cable. (Color photos are available on the *Circuit Cellar* FTP site.)

The lesson? Use a destructive test to verify some of the specs! This cable would pass any low-speed electrical test and might even pass an EMI test, but it isn't what you'd want for a critical application.

CONTACT RELEASE

This is the golden age of RF and analog hardware hacking, with more easy-to-use stuff available than ever before. Fire up those irons, keep building, and tell me how it works out!

Ed's an E.E., P.E., and author in Poughkeepsie, NY. You may contact him at ed.nisley@ieee.org. Type "Circuit Cellar" in the subject line to clear the spam filters.

PROJECT FILES

To download color photos and the EAGLE ULP to drill board holes, go to ftp.circuitcellar.com/pub/Circuit_Cellar/2005/181.



I Left My Chip in San Francisco

The chips and accessories featured at the March 2005 Embedded Systems Conference will be the technologies driving the automotive, industrial, and real-time applications of tomorrow. Read on for a look at what the industry leaders, from Atmel to ZMD, have in store for you.

Judging by the turnout at the March 2005 Embedded Systems Conference in San Francisco, the embedded business is shaking off the hangover of the past few years. There were plenty of attendees and exhibitors, and my schedule was completely filled with meetings with suppliers anxious to tout their wares.

Mark Twain is alleged to have said, "The coldest winter I ever spent was summer in San Francisco." Nevertheless, even the weather cooperated with a rare March heat wave, making for short-sleeve weather instead of the rain, fog, and blustery winds that are the norm. Enough of the travelogue. Let's get down to business and let the chips fall where they may, perhaps right into a socket on your next design.

UFO ALERT

Long-time readers will remember that UFO stands for Unidentified Fifty-One ("UFO Alert," *Circuit Cellar*, 54, January 1995). The fact that I can use the same lead-in more than 10 years later says it all. Yes, it's time once again to talk about the venerable 8051, the Energizer bunny of MCUs.

Traditionally, the most compelling feature of the decades-old '51 design wasn't found on the datasheet. Well, actually, it was—not in the specs, but rather in all the different supplier logos at the top of the page. The big advantage for the '51 isn't the chip; it's that you can get exactly the part you want by choosing from the catalogs of a dozen different suppliers. As a virtually open-source design, the '51 has become the de facto "people's MCU." In fact, as for the specs themselves, the '51 has long struggled with some skeletons rattling around in the closet. Chief among them has been performance, or more accurately, lack of same. By today's standards, the original part's 12-clocks-per-cycle dummy fetching design appears laughably quaint.

Competition among the many '51 suppliers has been a good thing. Innovative outfits like Dallas (now Maxim) and Cygnal (now Silicon Labs) led the way with faster parts, and now the heavy hitters are following suit. At the March ESC, both Atmel and Infineon announced hot-rod additions to their '51 lineups.

Just keep in mind that this doesn't mean you get 1 MIPS per 1 MHz. Thanks to the '51 CISC-like multi-cycle instruction set, branch penalties, and slow flash memory access time (wait states), these engines will stall from time to time. Nevertheless, it's clear that the performance skeleton in the '51 closet can move on to more permanent digs (i.e., under six feet of dirt).



Photo 1—*Silicon Labs showed off an extremely effective demonstration of their Direct Sensor Interface Technology in the form of a digital compass taking advantage of their hyper-51 '350 MCU's built-in 24-bit A/D converter.*

The Atmel AT89LP family includes parts with 2 to 64 KB of flash memory in 14- to 44-pin packages. The first two parts to hit the street are in 20-pin packages (DIP, TSSOP, and SOIC) and come with 2 or 4 KB of flash memory. Running at up to 20 MIPS (the fastest instructions execute in a single cycle at up to 20 MHz), they deliver a lot of bang for your buck, literally, because they cost less than $1 in volume.

Meanwhile, the Infineon XC866 delivers double-digit MIPS (e.g., 13 MIPS peak at 26 MHz) to blue-collar applications along with an extended temperature range (–40° to 125°C), a LIN, motor control PWM, a built-in voltage regulator, and up to 20-year flash memory data retention.

The Silicon Labs C8051F350 microcontroller isn't only the '51 MIPS master (100 MIPS peak), it also leads in the integration

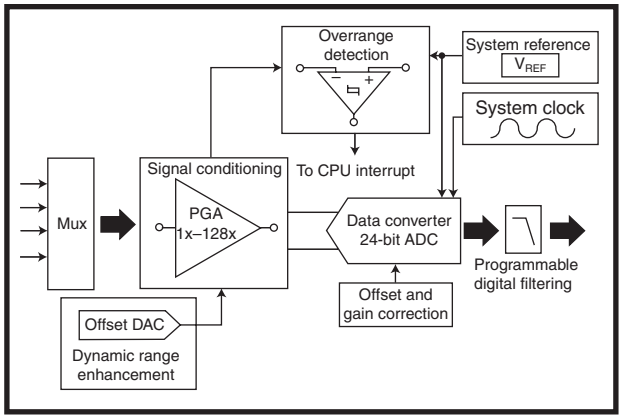


Figure 1—*The Silicon Labs '350 highlights the trend toward mixed-signal integration, supplementing the A/D converter with an entire signal conditioning analog front-end.*

of evermore complete and precise analog subsystems (see Figure 1). With amazing 24-bit resolution and a full complement of signal conditioning, the C8051F350 enables what Silicon Labs calls direct sensor interface technology (DsiT). As the name implies, the idea is to eliminate the need for the external analog front-end add-ons (e.g., op-amp and filter) typically required by low-gain sensors that are often too noisy.

At the ESC, Cygnal demonstrated a great example of the concept in action in the form of a digital compass (see Photo 1). The C8051F350 is directly connected to the various low-level sensors (e.g., magnetometer, accelerometer, and inclinometer), and all those MIPS are put to work crunching the nontrivial navigation algorithms.

After chewing on all of these fancy-pants parts, if you've still got a hankering for an old-school slow '51 with no analog features, check out the PB8051 microcontroller from Roman-Jones. It's a '51 with a difference all right: a virtual '51 emulated by the mini-me PicoBlaze soft-core processor running on a Xilinx FPGA! Think of it as a '51 in drag. Running the PicoBlaze core at 40 MHz, the PB8051 emulator manages to crunch through '51 instructions (even existing binaries in an external EPROM) at 1 MIPS or so.

FPGA SHRINK-WRAP

While I'm on the subject of FPGAs, check out the nifty modules from Memec and Pico Computing (see Photos 2a and 2b). These little gadgets



Photo 2—*FPGAs are slowly, but surely, migrating into mainstream embedded applications. As they do, third-party suppliers are responding with modular solutions ideal for prototyping, experimentation, and low-volume production. The handy gadgets shown here include Xilinx-based modules from Memec (a) and Pico Computing (b) along with the Altera-based FastPack from Parallax (c).*



are perfect for prototyping high-end applications, especially those using a soft-core CPU (e.g., MicroBlaze), because in addition to the FPGA itself, they include a bunch of memory (flash and RAM) and an Ethernet interface along with must-have glue logic (e.g., clock). They also give you a handle, literally, on the high-pin-count, fine-pitch, surface-mount FPGA packages that are otherwise a pain to play with. Both Memec and Pico offer lineups of Xilinx Virtex- and Spartan-based modules to choose from.

Meanwhile, Altera FPGA aficionados should head on over to Parallax. In addition to their well-known BASIC Stamps, Parallax offers a line of Altera Stratix and Cyclone-based prototyping boards. With its built-in plug board areas and handy I/O (switches, lights, and buzzer), the FastPack is handy for quick and easy prototyping and experimentation (see Photo 2c).

Neat stuff, but be prepared for some sticker shock. Like all things FPGA, these little modules command a relatively high price. To be fair, much of the premium is driven by the price of the FPGA itself, but even the most inexpensive modules have triple-digit price tags (e.g., $150 for the Memec Spartan-3-based Mini-Module and $195 for the Parallax Cyclone FastPack). Of course, you'll pay more if you have your heart set on a rocket science Xilinx Virtex-4 FPGA with a built-in PowerPC processor (or two!).

ARM FLEXES

It's been apparent to me for some time that ARM is poised to dominate the deeply embedded 32-bit market. Everything I saw at the ESC reinforced my conviction. Here's the convergence of factors that make it so.

The ARM architecture's roots are as old as the hills. Ironically, that's an advantage now because most embedded applications neither need nor are willing to pay for bloaty performance-at-any-price features.

Meanwhile, the reduced transistor count dovetails nicely with the march of Moore's law to deliver ARM-based MCUs with single-digit prices that are closer to those of an 8-bit part than the 32-bit computer chips of yore.



Photo 3—*Do you think that a 32-bit, ARM-based MCU is just too big for deeply embedded applications? Maybe the Oki '406x, packaged in a Wafer Chip Size Package (WCSP), will prompt you to think again.*

That's all well and good, but it's only part of the story. The compelling advantage for ARM is the number and diversity of the suppliers who are climbing onboard. It's the '51 story all over again, this time with a cast of 32-bits.

I've previously covered the groundbreaking ARM-based MCUs from Philips and Atmel. These are two major players in the MCU business and their blessing alone would be a pretty big deal. But in fact, Philips and

AmpFlow's high performance two-channel motor controller is ideal for robotic and industrial applications that require high power in an economical and compact package.



- Two channels operate independently or mixed together for "tank style" steering
- Supplies up to 180 Amps per channel at 12 to 40 Volts
- Full forward and reverse control with regenerative braking and no relays
- Adjustable current limiting and over-heat protection
- Accepts inputs from radio control, joystick, computer, or microcontroller
- Operates in open or closed-loop speed mode
- Motors may also be operated as heavy-duty position servos
- Accepts feedback from optical encoders or low-cost position sensors
- Single-channel version available with twice the rated output
- Compact controller measures just 7" × 5.5" × 1.5"

Several inputs are available for connecting switches or sensors via AmpFlow's 15-pin connector. The controller may also be connected to a PC to enable data capture, storage, and configuration of parameters using the included serial-port cable and the convenient graphic computer interface.

AmpFlow
AmpFlow.com

Call 650-593-6906 www.AmpFlow.com

Engineers Delivering Tomorrow's Designs

DESIGNCON 2006

Conference February 6–9, 2006
Exhibition February 7–8, 2006

Santa Clara Convention Center
Santa Clara, California

www.designcon.com

The premier conference and technology exhibition for semiconductor and electronic-design engineers!

OFFICIAL SPONSOR
Agilent Technologies

International Engineering Consortium
www.iec.org

Atmel are just the leading edge of the ARM MCU wave.

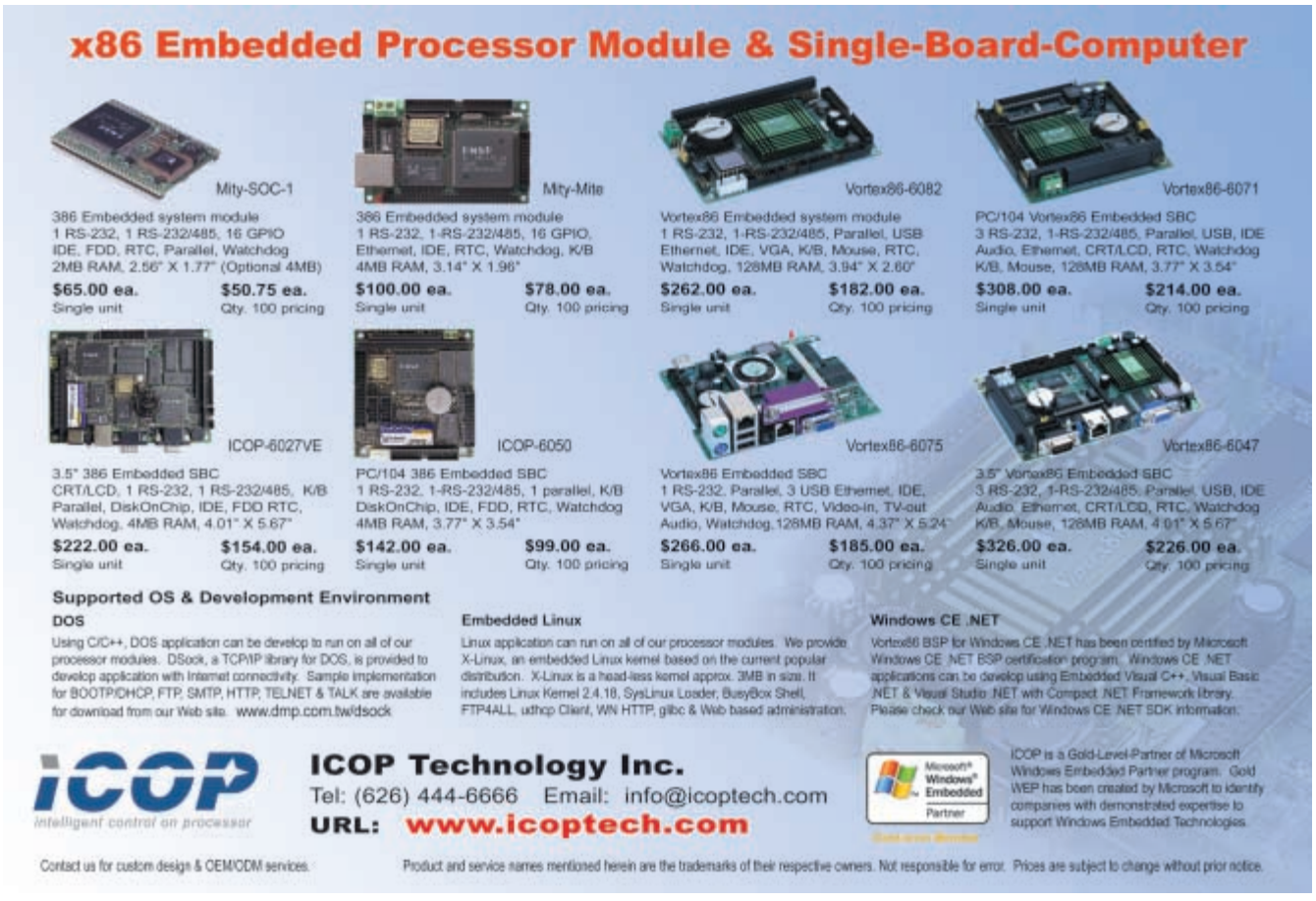
For example, at the show, Oki announced their flash memory-based ML67Q406*x* ARM7 single-chip MCU. It's a big deal by virtue of being an extremely small chip. I don't mean to say their chip is anorexic in terms of being starved for features. On the contrary, it's everything you've come to expect on a high-integration MCU with up to 128 KB of flash memory, a roomy 16 KB of SRAM, a 10-bit A/D converter, and plenty of timers and serial ports. It's even got some high-priced spread features like a two-channel DMAC and direct connection to codec chips for audio applications.

What makes the Oki part a big story is that the chip is tiny. As you can see in Photo 3, by exploiting their wafer-level chip size package (WCSP) technology, Oki manages to pack 32-bit horsepower and 64 I/O connections into a sliver of computation a mere 5 mm (less than 0.25″) on a side. But a small footprint isn't all that matters when it comes to downsizing. What isn't apparent in the photo is that unlike many surface-mount packages, WCSP parts are also extremely low profile (i.e., thin) and lightweight. It may not seem like a big deal, but every extra millimeter and milligram counts when you're trying to design a gadget that you can drop in your shirt pocket without looking like the Hunchback of Notre Dame.

The list of ARM7 microcontroller headliners seems to grow daily. STMicroelectronics has their STR7 lineup, with versions including USB, CAN, and both, which can do double duty on the desktop, in automotive applications, or on the factory floor. Analog Devices has an ARM version of their MicroConverter, which, like the existing '51-based version, targets the high-performance analog niche with a built-in high-speed (1 MSPS) analog signal chain including an ADC and DAC. Oh, mustn't forget Texas Instruments, which brings formidable

	Atmel AT91SAM	Philips LPC21	OKI ML67Q406	ST STR7	Analog Devices	Texas Instruments
Flash memory (KB)	32–256	32–512	64–128	64–256	8–62	64–1,024
SRAM (KB)	8–64	8–32	16	16–64	8	4–64
Speed (MHz)	55, 60	60	33.33	50	45	48, 64
Power	3.3 V	1.8-V core, 3.3-V I/O	2.5-V core, 2.5–3.3-V I/O	3.3 V	3.3 V	1.8-V core, 3.3-V I/O
Package	48–100 LQFP	48–64 LQFP	64 WCSP, TQFP 84 LFBGA	64–144 TQFP	40–80 LFCSP, LQFP	80–144 LQFP

Table 1—*Attention shoppers, there's a special on ARM7s in the 32-bit flash memory single-chip MCU aisle. With dozens of parts from top-tier suppliers to choose from, ARM7 MCU buyers couldn't have it better.*



ARM expertise (their ARM-based chips are in millions of cell phones) to bear on the merchant market with its TMS470 family of ARM7 single chips.

Let's see, we've got Philips, Atmel, Oki, STMicroelectronics, Analog Devices, and Texas Instruments. A rather formidable lineup, wouldn't you agree?

Table 1 (page 81) gives you a view from 50,000', but it doesn't get into the I/O features. Because each supplier offers a range of parts with different memory, packaging, and I/O, there is just no easy way to cover the spectrum of I/O represented by dozens of parts in a simple table.

Most chips include the usual suspects (the basic serial ports, counter/timers, and glue logic you'll find on a typical MCU) when it comes to I/O. Beyond that, the different parts fall into groups that target major market segments with application-specific I/O add-ons. Industrial, automotive, and real-time applications are served by chips with specialized high-speed, multi-channel timer subsystems and

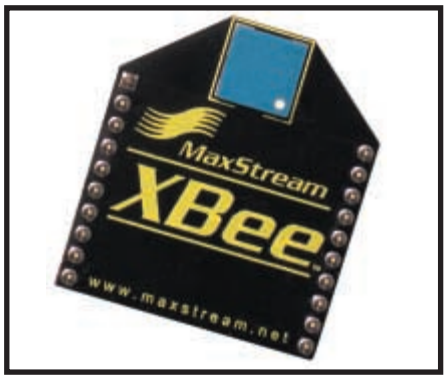


Photo 4—*Now that the first 802.15.4/ZigBee chips are hitting the street, look for the first wave of products (e.g., the XBee module from MaxStream) to hit distributors' shelves.*

CAN interfaces. For example, members of the TMS470 lineup come with a high-end timer (HET) subsystem. Built around a completely independent 96-bit VLIW microengine and handling up to 32 timing channels, the HET certainly lives up to its name (see Figure 2).

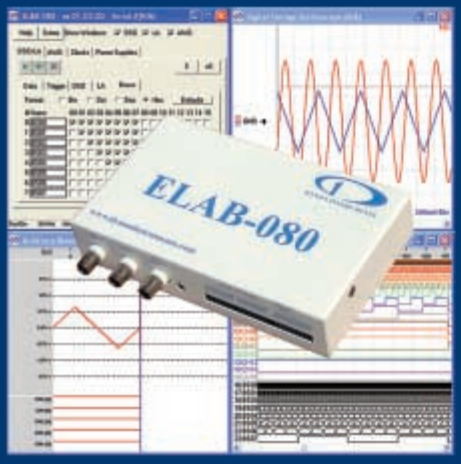
Meanwhile, office-oriented parts typically come with Ethernet or USB ports. Although Atmel's ARM9-based AT91RM9200 isn't a single-chip MCU (no flash memory), it stands out with built-in Ethernet and USB host and USB device ports. Have it your way.

One of the biggest ARM MCU developments at the ESC wasn't an MCU at all. Rather, it was the announcement by Actel that they secured a license to deliver an ARM7 soft-core for their ProASIC flash memory FPGAs. I was high on ARM, and I was high on flash memory FPGAs; needless to say, the Actel announcement had me seeing stars. Details to follow. But even at the press release level, it's easy to see this is a pretty big deal. Why? Consider that there are essentially three possible chip implementations to choose from for a particular application: a standard chip, an ASIC, or an FGPA. Deciding which solution is best depends on the application's specific characteristics (e.g., production volume) and the march of technology (e.g., lower-cost silicon and tools).

Thus, only a controller architecture that's available across all implementations can be truly "future proof." And

DYNON INSTRUMENTS

A Complete Electronics Lab for $495



ELAB-080

Includes 5 Instruments
- 2 Channel 80 MHz Digital Storage Oscilloscope
- 16 Channel Logic Analyzer, Synchronous with DSO
- Arbitrary Waveform Generator
- 2 Programmable Power Supplies
- 2 Programmable Clocks

Plus....
- Powerful PC User Interface Program
- Probes
- Cables

Now includes FFT!

www.dynoninstruments.com
Woodinville, WA 98072 (425) 402-0433

guess which 32-bit controller architecture can make that claim?

KILLER BEE

Judging by all the action at ESC, the ZigBee folks have been as busy as, well, bees. Promulgating a new standard is never quick or easy, so let's give the ZigBee Alliance credit for herding a lot of cats (150 members at last count) along in a relatively expeditious and orderly fashion.

In addition to all the technical and marketing activities, there have been a lot of developments in the boardroom as well. Chipcon has leveraged their early leadership in ZigBee silicon with the acquisition of ZigBee stack software supplier Figure 8 Wireless. At the same time, Ember (an OEM customer of Chipcon) will take future silicon matters into their own hands via their acquisition of ZigBee know-how from chip design boutique Cambridge Consultants.

I see the process of getting a new standard going as kind of like building a fire. The first step is to find some wood (i.e., hash out the standard, get the details down on paper, and perhaps dabble with experimental silicon). The next step is starting the fire (i.e., introduce tested, verified, and certified commercial silicon). Finally, presuming you get past a puff a smoke, the fire is fed with emerging applications, each of which contributes to the blaze.

At this point, the ZigBee fire is going, and all indications are that it's only going to get bigger and hotter. The standard has been ratified. Chipcon, Ember, CompXs, and Freescale all offer certified silicon. There are plenty more ZigBee-capable chips in the pipeline. At the conference, ZMD announced their new ZMD44101 900-MHz chip that integrates not only the entire IEEE 802.15.4 radio but also other thin MAC functions that offload an attached MCU of grunt work like packet framing and CRC. Meanwhile, Atmel takes a similar tack from the other side of the MCU-radio link. Their AT86ZL3201 Z-Link controller is an AVR with ZigBee-centric enhancements such as built-in AES encryption hardware, a random number generator, and send/receive buffers. Now early adopters have the chips they need to create high-integration, off-the-shelf solutions such as the XBee line of OEM modules from MaxStream (see Photo 4).

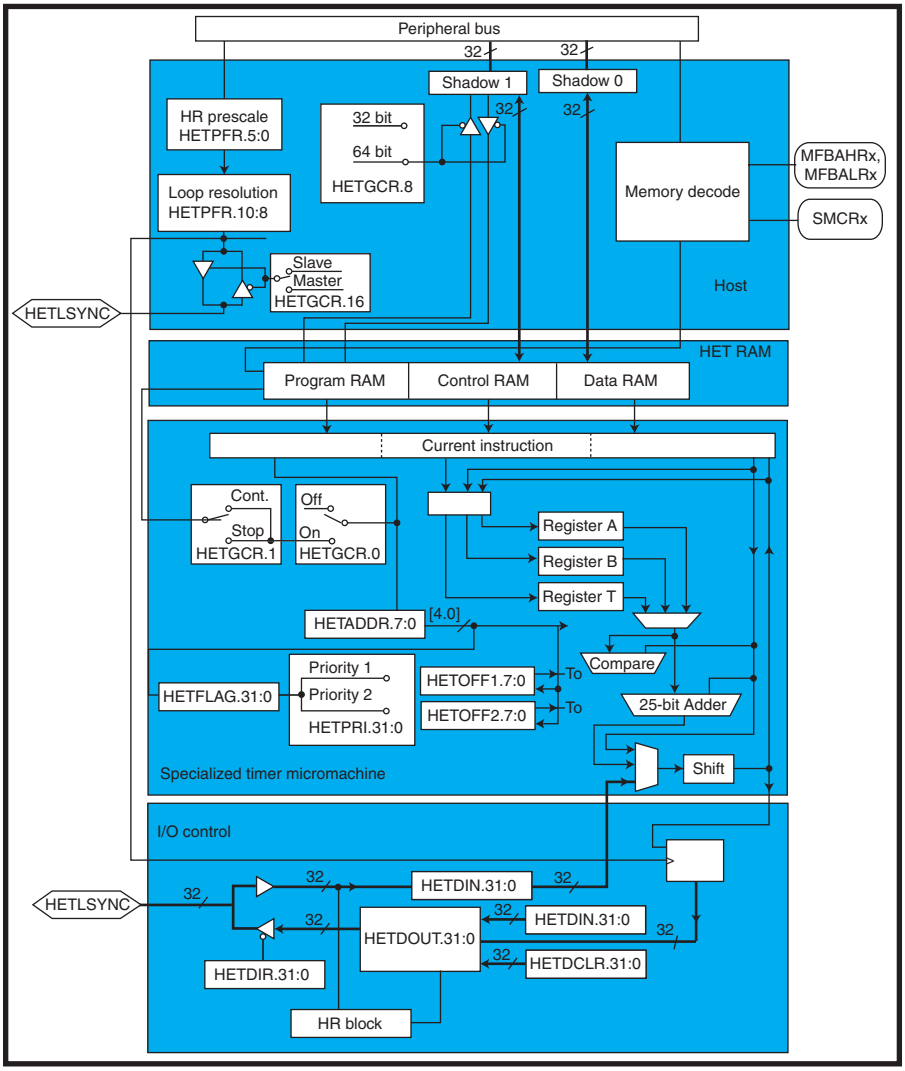


Figure 2—*With a 96-bit VLIW engine, not to mention a 180-page datasheet, the high-end timer (HET) subsystem on Texas Instruments ARM MCUs is a complex processor in its own right.*

ESCAPE FROM SAN FRAN

Do you know the way to San Jose? Better figure it out, because next year the spring ESC is moving back to the San Jose Convention Center.

Old-time attendees will recall how the ESC outgrew the San Jose facility. It got so crowded that some exhibitors were forced into outdoor tents and attendees had to hit the streets in order to find conference sessions held offsite in hotels somewhere. Speaking of hotels, more have been built in the years since the move, so the search for a place to stay should be less daunting. And now that the convention center has completed a multi-year expansion project, it will presumably accommodate exhibitors in style and give everyone a little more elbowroom.

Sure I'll miss the San Francisco nightlife and cable car rides. On the other hand, now I can drown my sorrows by downing some chips and DIPs at nearby Fry's Electronics. And moving to San Jose will make it easy for you Silicon Valley locals to drop in for a visit with the *Circuit Cellar* gang at our booth. Hope to see you there!

Tom Cantrell has been working on chip, board, and systems design and marketing for several years. You may reach him by e-mail at tom.cantrell@circuitcellar.com.

IDEA BOX

THE DIRECTORY OF PRODUCTS AND SERVICES

AD FORMAT: Advertisers must furnish digital submission sheet and digital files that meet the specifications on the digital submission sheet. **ALL TEXT AND OTHER ELEMENTS MUST FIT WITHIN A 2″× 3″ FORMAT**. Call for current rate and deadline information. Send your disk and digital submission sheet to: IDEA BOX, Circuit Cellar, 4 Park Street, Vernon, CT 06066 or e-mail kc@circuitcellar.com. For more information call Sean Donnelly at (860) 872-3064.

The Suppliers Directory at www.circuitcellar.com/suppliers_dir/
is your guide to a variety of engineering products and services.

PHYTEC
phyCORE® OEMable Single Board Computers
8051 CAN
C166/XC16x/ST10
PowerPC
x86:
ColdFire:
XScale:
ARM7:
■ sub-miniature SBC subassemblies (as small as 47x55 mm)
■ low EMI, multi-layer design plugs like "big chip" into target circuitry
■ 32 KB to 128 MB external SRAM and Flash (controller-dependent)
■ CAN, Ethernet, RS-232/485; USB, 3rDA; ADC, DAC (controller-dependent); free I/Os and I/O
■ available in widemos and compilers
■ insert-ready COTs subassemblies accompany you from evaluation through prototyping to end production... accelerating your design cycle and reducing time-to-market
(800) 278-9913

video|text overlay
Turn serial data into video text:
BOB-3 · XBOB · COG-4
www.decadenet.com
DECADE ENGINEERING
503-743-3194 Turner, OR, USA

I2C, MDIO, CAN, SPI, XA Kits, & ISP FLASH
PC-ISP-FLASH programs 80C51, XA and AVR ISP/IAP FLASH micros from Philips and Atmel with user upgradeable Windows software, ISP interface cable, software and documentation included for **$199.00**.
PC-I2C-KIT allows bi-directional communications between any PC and I2C, SPI or MDIO based peripherals. Windows software, board, interface cable and documentation included for **$249.00**.
PC-I2C-DEV lets the user write programs to control the I2C/SPI/MDIO interface board directly using Microsoft Visual C/C++, Borland C/C++, Microsoft Visual Studio, etc. Includes a software DLL for custom user applications and a complete PC-I2C-KIT for **$499.00**.
XTEND Kits allow the user to debug or prototype any Philips XA based design. Kits include XA Dev Board with Monitor and Debugger, Software, Documentation, Schematics, Power Supply and Serial Cable.
XTEND-Gxx (G3 & G49) or **XTEND-Sxx** (S3) for **$249.00**
XTEND-C3 or **XTEND-H4** for **$299.00**
FDI
www.teamfdi.com
VISA/MC/Amex
Future Designs, Inc.
2702 Triana Blvd
Huntsville, AL 35805
(256) 883-1240
Fax (256) 883-1241

USB
Add USB to your next project—it's easier than you might think!
USB-FIFO up to 8 mbps
USB-UART up to 3 mbps
USB/Microcontroller boards
pre-programmed with firmware
Absolutely NO driver software development required!
www.dlpdesign.com
Design services available
DLP Design

Epsilon
Surface Mount Assembly
Need 5...10...100 ...500 SMT Boards?
Fine Pitch
Prototypes
BGA
Cost Calculator on Web
14666 N 74th St. C. Scottsdale, AZ 85260
480-998-3503
www.EPSILONMFG.com

New Development Kits!
$614
Wireless ZMD Edition
Includes two Wireless ZMD Boards to develop spread spectrum wireless devices.
$474
Embedded Serial Busses
Includes an Embedded Serial Busses Board to develop robust applications with serial busses such as I²C and SPI.
Both Kits Include:
PCW Compiler • Exercise Booklet • Carrying Case
DC Adapter and Cables • ICD
CCS
Call Sales: 262-522-6500 x35
Email: sales@ccsinfo.com
www.ccsinfo.com/picc
Same Day Shipping With Most Orders

PC KEYBOARD EMULATION

Interface Keypads, Switches, or RS-232 to your PC Keyboard Input

MODEL KE24 ONLY $99.95



• Up to 12 x 12 matrix • Programmable • RS-232 Port • Macro Capability

The KE24 is the ultimate in flexibility. Inputs or serial data can emulate any of the 101 keys from a standard keyboard.

MODEL KE18 ONLY $44.95



• Up to 9 x 9 matrix • 2.5" x 3.0" Size • PC Keyboard Port • PCXT, AT Compatible

The KE18 combines a multitude of features with small size at an economical price. Custom units available.

HAGSTROM ELECTRONICS
11 Fiddlers Green
Lansing, NY 14882

Toll Free: 888-690-9080
Phone: 607-533-4441 Fax: 607-533-4443
www.hagstromelectronics.com

ValueCAN

The High Value Tool For Controller Area Network

- USB to CAN
- Simple software analyzer included
- DLL with examples for custom applications
- PC isolated from CAN
- 100% bandwidth at 500Kb

Only

$295

ICS RADDAQ

www.intrepidcs.com/vcan

It writes your USB code!

NO Need to be a USB expert!

HIDmaker ($399) – creates ready to compile PC & PIC programs that talk to each other over USB.

Choose your favorite languages!
PIC: Pic Basic Pro, CCS C, Hi-Tech C, MPASM. **PC:** VB6, Delphi, C++ Builder.
Single chip solution: PIC with built-in USB

HIDmaker Test Suite ($149)

USBWatch – shows your device's USB traffic, even during 'enumeration', without expensive equipment.

AnyHID – Test any USB HID device. See what data it sends, even what the data is used for.

Trace SYSTEMS, Inc.

301-262-0300
WWW.TraceSystemsInc.com

Put the Power of USB in Your Designs

Learn how to select a controller chip for your project, write device firmware & host applications, use USB classes for quicker programming, develop USB On-The-Go devices, and much more!



NEW! THIRD EDITION

USB Complete
Everything You Need to Develop USB Peripherals, Third Edition
by Jan Axelson
ISBN 1-931448-02-7 $49.95
Lakeview Research www.Lvr.com

From the author of Embedded Ethernet and Internet Complete

HIGH PERFORMANCE SOCKETS AND ADAPTERS



For Test, Prototype, SMT package emulation, Socketing.

We have solutions for all packages from SOT23 to BGA.

---Quick Turn Customs---

Ironwood Electronics
1-800-404-0204
www.ironwoodelectronics.com

USB Data Acquisition

ADU208 -USB Relay I/O Interface

Complete SDK Online at: www.ontrak.net



FEATURES
-8, 5-AMP relay outputs
-8, ISOLATED digital inputs
-Port Powered, Aux 5VDC output $189.00 QTY 1

Other Models:
ADU200- 4 Channel Version with RS232 $139.00
ADR218- Solid-State Version 8 Channel $225.00
ADU100- 3 CH, 16-Bit ISOLATED Analog Inputs, PGA, 4 digital I/O, RS232 and 5 AMP Relay Output $199.00

ONTRAK CONTROL SYSTEMS INC.
PH: (705) 671-2652 Fax: (705) 671-6127

www.ontrak.net

NOW YOU CAN HAVE IT ALL!

- *Easy to Use*
- *Low Cost*
- *Low Power*
- *Tiny Size*
- *High Speed*
- *DOS Environment*

with the



DOS Stamp™

Easy Software Development: Use your C/C++ or Basic compiler to produce DOS EXE. Download EXE to flash disk via serial port. EXE runs on power up.

Standard Features: BIOS & DOS-ROM, 128K flash disk, 512K SRAM, 40 MHz CPU, 16 digital I/O (opto rack interface), 2 RS-232, 2 timer/counters, simple bus interface, real-time clock

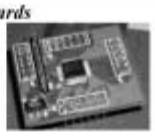
Options: 8-ch 12-bit ADC, flash disks up to 288 MB, 1 RS-485, TCP/IP

Tiny Size, Low Power: 2"x2.6", 5V @ 200 mA at full speed, 40 mA at lowest speed.

Visit http://www.bagotronix.com for info, prices, and FAQs

BAGOTRONIX
2900-1 Crescent Drive
Tallahassee, FL 32301
850-942-7905 phone & fax

ARM/AVR Proto Boards



FROM AN IDEA TO THE FINAL PRODUCT

- Controllers
- Embedded Controls
- Analog/Digital/Software
- Test Program Development
- Printed Circuit Boards Layout
- FPGA

ADM provides 30 years of experience and will examine your needs. We will establish parameters and generate a realistic timetable and seek out the optimum path to successful implementation.

ADM DESIGNS, LLC.
Website: www.adm-designs.com
E-Mail: info@adm-designs.com
Contact us: (602) 283-4983

PROTOTYPE CIRCUIT BOARDS

$85 US for two 5" x 6" 2-layer boards

Shipped NEXT BUSINESS DAY if data is received by 1:00 pm EASTERN

www.apcircuits.com

AP Circuits

(403) 250-3406

VISA staff@apcircuits.com

PLCs for OEMs
Program like a PLC (Ladder + BASIC)
Low Cost & Flexible like a Micro-p

Ethernet XServer / Web HMI



$199

8 Analog i/os
24V Digital i/os
MODBUS
LCD Port, PWMs
RTC, Stepper
Interrupts,
Hi-Speed Counter
Modem / Internet

New Option:
Ethernet / Wi-Fi
w/ Free Web HMI

Incredibly Easy to Program!
Our software is used by many colleges for teaching PLCs!

Get Free Ladder Logic Simulator:
www.tri-plc.com/cci.htm

TRi Triangle Research

Tel: 1-877-874-7527 - PLC specialist since 1993

Unique Keyboard Emulation IC allows your circuit to place characters and control directly into almost any P.C. application (USB to RS-232 chips cannot do that). Very easy to use. Parallel or Synchronus Serial interface.

USB

- No Software Needed
- No Drivers Needed
- No "Wedge" Needed
- 24 pin DIP or SOIC
- Only one external component (resistor)
- Works with almost any PC application

RADOVAN ROBOTICS 805-375-7059
http://www.radovan.org/EasyInput/index.htm

Flashlite 186



- 186 processor @ 33 MHz
- DOS w/ Flash File system
- 44 Digital I/O lines w/ CPLD
- Console / Debug Serial Port
- 7-34V DC or 5V DC power
- Accepts 8MB DiskOnChip
- 512K DRAM & 512K Flash
- Expansion options with Peripheral Boards
- 2 Serial Ports
- 2 16-bit Timers
- Watchdog Timer

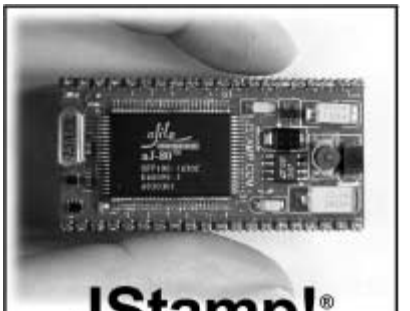
$69 QTY 1

$99 Development System

Development kit includes:
- Flashlite 186 controller
- Borland C/C++ ver 4.52
- FREE Email Tech Support
- Serial Driver library
- AC Adapter and cable

Call 530-297-6073 Email sales@jkmicro.com
On the web at www.jkmicro.com

JK microsystems



JStamp!®

FAST 32-bit native-execution Java
Executes 3 million Java byte codes/sec
compact 1 x 2 inch DIP40 package
Real J2ME CLDC Java (threads, floats)
2048 KB Flash & 512 KB SRAM

www.jstamp.com

SYSTRONIX®
Salt Lake City, Utah, USA
Java is a TM of Sun Microsystems, Inc. JStamp is a TM of Systronix, Inc

Web Controlled Relay

Control devices over any IP network



- no programming required
- simple browser-based setup
- password protected
- on/off and pulse modes
- 10 Amp contacts (SPDT)
- optically isolated control input
- only $95 (requires 5VDC power and relay socket)
- power over ethernet (802.3af) option available

WebRelay™ can be controlled over an IP network using a web browser, or by sending text commands from a computer or industrial controller. In addition, two WebRelay™ units can be configured to extend the output of a PLC to another building, or allow a switch to control a device in another city. Limitless possibilities!!

For more information, see
www.ControlByWeb.com

(435) 750-5999

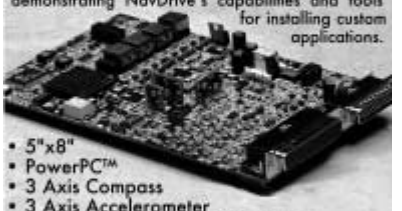
Serial, Ethernet, Web, Email
Simple, DTE, DCE, No SW Changes
$99.95 Qty 1



Grid Connect Inc.
+1 800 975-GRID
www.serial-to-ethernet.com

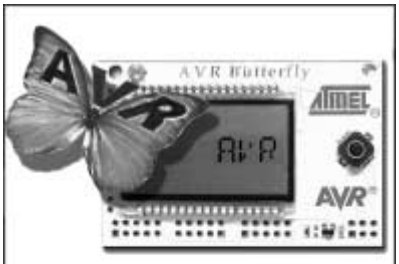
NavDrive™ II
Autonomous Robot Controller

NavDrive is a combination of software and hardware for building custom mobile robot applications. Source code is provided for the extendable C++ framework that integrates sensor data, and controls a robot's path while avoiding objects. On-board flash contains a complete application demonstrating NavDrive's capabilities and tools for installing custom applications.



- 5"x8"
- PowerPC™
- 3 Axis Compass
- 3 Axis Accelerometer
- 3 Axis Gyroscope • GPS Interface
- 4 Channel Ultrasonic • 2 Encoder Inputs
- 2 Axis Motor Control - Tracked or Steered
- Many Digital/Analog Inputs and Outputs

Integrated Knowledge Systems
www.IKnowSystems.com
602-705-8978



AVR Butterfly $19.99
C Programming Text $39.95
Projects Kit $24.99
Free Unlimited C Compiler
www.SmileyMicros.com

DEVICE PROGRAMMER, USB ROM Emulator, EPROM ERASER

TopMax-II



$995

UniMax



USB 2.0 $345

TopMax-II : TopMaxII is one of our finest products to date and a product destined to become industry standard. If you are interested in cost-effectiveness and increased performance efficiency, TopMaxII is a must-have. TopMaxII programs 128-Mbit 16-bit word Flash Memory in 2 minutes via a USB port on your PC.

UniMax : UniMax is the latest universal programmer from EE Tools, Inc. It utilizes the same design as TopMaxII, but by eliminating some of TopMaxII's extra components, minimizes product cost.

EeRom-8U
$495
"USB Interface"



EeRom-8U : 8Mbit-45ns EPROM/Flash Emulator, High speedy USB port interface(Plug & Play)
- Fast download(8Mbit per second), Lower Voltage, 8bit, 16bit and 32bit data split
- High Speed access time(built-in 8M SRAM 15ns)
- No external power is needed, 9x/XP/NT/2000
- Accepts all standard Hex formats and binary files
- Jumper less Emulation size configuration



LA6T-110
$115



LA6T-110 : UV ERASER for Personal user, Fast and efficient, Molded Plastic Case, Hold 9 of 28-pin or 6 of 40-pin EPROM. Timer(15-40 min).
Programming Adapters(low price) : 8-SOIC($70), 32-PLCC($70), 44-PLCC($115),48-TSOP($135), 44-QFP($65), 52-PLCC($165), 68-PLCC($165).
EPROM Emulator Adapter: 32/28-pin($95) 44-pin($115) DIP-to-PLCC

Made in USA
EE Tools
www.eetools.com
E.E.Tools, Inc.
3350 Scott Blvd. B51-02
Santa Clara, CA 95054
TEL:(866) 496-6664(free)
FAX:(408) 496-0006

Custom Printed Circuit Boards
Nobody beats our PROTOTYPE PCBs
We are the 5 PCBs/5days specialist
AMAZING VALUE
2 layer: Get 5 PCBs for $13 each in 5 days
4 layer: Get 5 PCBs for $33 each in 5 days
(Upto 20 sq inch, FR-4, 0.062" thick, 7 mils trace/space,
15 mils finished hole min., unlimited drill sizes,
FREE Tooling, green LPI soldermask, white silkscreen)
(More sizes and offers available. See website for details)
Order Online
www.PCBFabExpress.com
Phone: (408) 857 0039
email: support@pcbfabexpress.com
PCBFABEXPRESS
High Quality PCBs @ Low Impact Prices

ElectronicKits.com



Usb/Serial Port PIC Programmer
Educational Electronic Kit $34.95
* Connect to computer via USB or Serial Port
* Free Software and Updates
* Windows 9x/Nt/2000 XP Compatible
* ICSP (In Circuit Serial Programming)
* Quality PCB and Silkscreen
See complete details at www.ElectronicKits.com
USB and USB Flash Pic Programmers Available
Carl's Electronics Inc. Offering hundreds of Educational Hobby Kits
Visit our website at www.ElectronicKits.com

Full-featured 9S12
with embedded Ethernet!



www.technologicalarts.com

IDE, Compilers, ICD, Simulators, Programmers...
Emulators



8051 PIC® AVR®
80196 XEMICS
MSP-430 SENSORY
Phyton 718.259.3191
www.phyton.com

"Your Complete PCB Design Services for a low $35/hour"

✓ Product Design, Schematic Capture
✓ PCB Design Engineering & Layout
✓ High-speed, DFM, EMI Controls
✓ Specctra Auto-Interactive Routing
✓ Component Library Development
✓ Contract Fabrication/Assembly

Mentor Graphics ◆ Cadence ◆ OrCAD ◆ PowerPCB ◆ Specctra ◆ CAMCAD

VNISOURCE DESIGN
www.vnisource.com info@vnisource.com
Tel: (972) 333-9573 Fax: (469) 675-0346
Contact us today for low cost high quality design!!!

Lifetime Warranty
SeaI/O™
Data Acquisition Solutions
- Seven Models Offer Choice of Optically Isolated Inputs, Reed Relay Outputs, Form C Relay Outputs, and TTL Interface for Industry Standard Solid-State Relay Racks
- Status Indicator LEDs for Communication, Fault, and Power
- Field Removable Terminal Block Connectors
- Power Input via Terminal Block or DC Jack
- Daisy Chain up to 247 Modules
- Rugged Metal Enclosure



Choice of Connectivity:
Ethernet
USB
RS-485
RS-232



SEALEVEL
Sealevel Systems, Inc.
155 Technology Place
Liberty, SC 29657 USA
864.843.4343 TEL
sales@sealevel.com
www.sealevel.com

okwelectronics.com
RF Remote Control Systems
AM 315 MHz
'Mix-n-match' Encoders & Decoders

110 SERIES KEYFOBS
- Transmitter encoders for OEM systems
- Secure KEELOQ protocol
- AM 315 MHz frequency
- 1,2 or 3 push button switches
- LED transmission indicator
- Range up to 165ft
- Auto shut-off
- ABS enclosure
- Custom overlays
- FCC compliant



AM-HiRK DECODERS
- Miniature RF Rx/decoder modules
- Secure KEELOQ protocol
- AM 315 MHz frequency
- Lazer trimmed hybrid
- Range up to 330ft
- FCC compliant



More info. @ www.okwelectronics.com
OKW Electronics, Inc.
Tel: (800) 965 9872 Fax: (412) 220 9247
300 Old Pond Road, #203, Bridgeville, PA 15017
Email: sales@okwelectronics.com

okwelectronics.com
In-Circuit Emulator
ICEPIC 3
Next Generation Emulation!
All new universal In-circuit emulator for the PIC Microcontroller. Exceptional performance and value.

FEATURES
- Operates within MPLAB
- Hardware support for single steps in 'C'
- Fast USB connection
- Microchip assembler and 'C' compatible
- Source level debug in assembler or 'C'
- Full speed 40MHz emulation

From only $900*



- 2.8/5V Emulation
- Plug-in daughterboards ordered separately to configure system to the desired PIC

*Daughterboard not included in price.

SEE OUR WEBSITE FOR PIC COMPATABILITY!
More info. @ www.okwelectronics.com
OKW Electronics, Inc.
Tel: (800) 965 9872 Fax: (412) 220 9247
300 Old Pond Road, #203, Bridgeville, PA 15017
Email: sales@okwelectronics.com

www.can232.com
The original (Controller Area Network) CAN to RS232 serial converter.

Powered with an Atmel® AVR® RISC microcontroller and a Philips SJA1000 CAN controller makes a unique combination of high speed transfers possible even at higher CAN bus speeds. CAN232 has also a built in receive FIFO of 32 messages which makes it possible to receive small bursts of messages without losing any frames.

Does it sound interesting?
Read more at www.can232.com

* Free sample programs in Delphi, VB, Linux, C & C++
* 8-15VDC supply via CAN connector (CiA DS102)
* Small size 68mm by 30mm (2.7" by 1.2")
* Timestamp in mS on CAN frames
* Operating system independent
* Both 11 & 29 bit ID support
* ASCII or Binary protocol
* Firmware upgradable
* Custom design
* CE approved



Philips is a registered trademark of Koninklijke Philips N.V.
Atmel® and AVR® are registered trademarks of Atmel Corporation

I²C™/SMBus
- Bus Monitors
- Protocol Analyzers
- Host Adapters
- Multiplexers
- Battery Applications
- Software Tools



MCC
Micro Computer Control
I²C is a trademark of Philips Corporation
www.mcc-us.com

Everything you need to build your own...
MOBILE ROBOT
- SONAR UNITS
- VISION SYSTEMS
- MOTOR DRIVERS
- OPTICS
- MICROPROCESSORS
- ARITIFICIAL INTELLIGENCE
- WEB CONTROLS



Zagros Robotics
PO Box 460342, St. Louis, MO 63146
(314) 768-1328 · info@zagrosrobotics.com
www.zagrosrobotics.com

2005 Advertising Calendar
October Issue #183
Analog Techniques
Space close: Aug. 10
November Issue #184
Graphics & Video
Space Close: Sept. 12
Call: Shannon Barraclough
(860) 872-3064
Shannon@circuitcellar.com
CIRCUIT CELLAR®

PROBLEM SOLVED!

Custom Hardware & Software Design

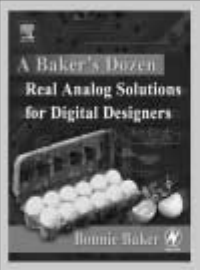


- Rapid Prototypes
- Cost-effective Design
- Free Consultation and Quotes

MJS CONSULTING INC.
Mark J. Samarin M.S.E.E.
http://www.mjsconsulting.com
253-858-6844

ANSI C software development tools for:
IMAGEcraft™
NEW: ARM7 compiler (Philips LPC, Atmel SAM, etc...)
Version 7 Tools
Full-featured 45-day demos on our website!
CypressM PSoC
AVR & tinyAVR
HC08,11,12,16
TI MSP430
ARM
high powered development, garage powered prices!
hardware kits:
Elektronikladen - HC08 & HCS12 / 12 ref. design & starter kits
CANDIP - AVR/CAN on a chip!
Ethernut - Complete AVR TCP/IP solution
www.imagecraft.com
info@imagecraft.com
706 Colorado Ave. Palo Alto, CA 94303
(650) 493-9326 • FAX (650) 493-9329
VISA / MC / AMEX
Check / Money Order
...say 'NO' to JURASSIC PRICES!

10% Off & Free Shipping



A hands-on designer's guide to the most important topics in analog electronics!

Includes discussion of A/D and D/A conversion, op amps, filters, and much more.

Save 10% and get Free Shipping at
www.books.elsevier.com/computereng
Use Offer Code 82459

EmbedDisk -40 to +85°C Extreme Temp



IDE Flash Industrial Grade Storage

MSTI Media Stream Technologies Inc.
www.embeddisk.com
info@embeddisk.com

ICOP ICOP Technology Inc.
www.icoptech.com
info@icoptech.com

PROTOCESSOR

Add Industrial Automation Protocols to your Device - Fast



Ensure connectivity using interchangeable, low cost OEM Modules with a common PCB footprint and software interface. Development kit with "C" source code available.

www.ProtoCessor.com
800-317-8319

FlashCore-B (FB)™ Add embedded mass data storage, 16-bit ADCs to your product

$79 Qty 1 $34 OEM

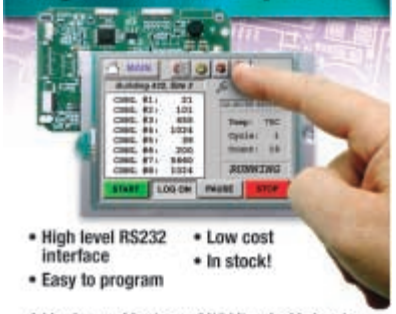


- 2.1" x 2.4", C/C++ programmable.
- CompactFlash interface with FAT file system.
- 16-bit ADCs, DACs, RS232 and TTL I/Os.
- Ultra-low quiescent current for battery power.

50+ Low Cost Controllers with ADC, DAC, 18 UARTs, 300 I/Os, solenoid, relays, CompactFlash, LCD, DSP motion control. Custom board design. Save time and money.

TERN INC. 1724 Picasso Ave., Suite A, Davis, CA 95616 USA
Tel: 530-758-0180 • Fax: 530-758-0181 www.tern.com • sales@tern.com

Add a color touch interface to your embedded product!



- High level RS232 interface
- Low cost
- Easy to program
- In stock!

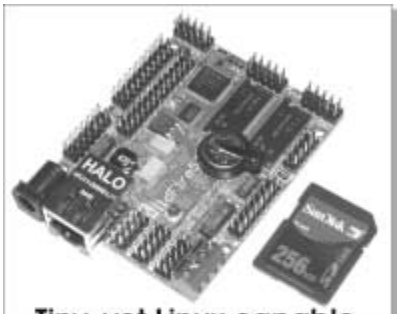
Add color graphics to any 8/16 bit embedded system. Easy, fast and flexible. Up and running in hours!

REACH TECHNOLOGY INC.
www.reachtech.com • 510-770-1417
842 Boggs Avenue • Fremont, CA 94539

Low Cost CAD Software For Windows 98, XP, NT, 2000

- Circuit design package with schematic entry, circuit board layout with autorouting, and simulation for only $599!
- Buy modules starting at $149 (**SuperCAD**, **SuperPCB**, **mentalSPICE** & **SuperSIM**)
- Encyclopedia of Electronic Circuits on CD with 1000 electronic circuits including schematics and design notes. Vol. I & II, each for only $99

Contact **Mental Automation** at
(253) 858-8104
http://www.mentala.com



Tiny, yet Linux capable

16MB fast SDRAM
64MHz Coldfire MCU
1 - 4 MB flash memory
SD card socket (to 1GB)

10/100 Ethernet
3 serial ports (RS232)
1 CAN port
LCD/KPD port

Eclipse/CDT dev. env.
serial debugger incl.

64 I/O pins & capabilities to burn...

www.steroidmicros.com

China Connection...

PCB Production,
Assembly,
Turnkey

Prototype ~ Production

D/S PCB ~ 12¢ per in^2
4-L PCB ~ 20¢ per in^2

10-Day Delivery w/solder Mask
NRE not included
UL Approved

MYLYDIA INC.
1-800-Mylydia
sales@mylydia.com
www.mylydia.com

Get Hi-Speed USB 2.0 Without Learning USB!



QUSB2: $149 qty 1

QuickUSB Applications
> Parallel/RS-232 port
> I²C/SPI chip controller
> DSP to PC Interface
> Micro to PC Interface
> FPGA to PC Interface
> And many more!

Easy, Low Cost Licensing!
QuickUSB
www.quickusb.com

SNAP CIRCUITS

Have Fun Learning All About Electronics

Build Over 750 Projects!



Includes Over 50 Computer Interfaced Experiments!

Use code CC08 for a FREE gift with your order!

www.BuilditPlayit.com
Toll Free 1-800-453-4214

Radiotronix

RF TX



RCT-AS

The RCT-XXX-AS series RF transmitter modules offer a cost effective, high-performance solution for remote control type applications. The transmitters are very small (.25" x .4"), have a wide operating voltage range (2-12V), and can deliver 0dBm output power into a 50 ohm antenna at 3V.

www.radiotronix.com 1.886.814.4446

Radiotronix



RCR-H2

- -109dBm Sensitivity @ 4800 baud
- Use with any OOK/ASK TX
- Superheterodyne design
- 5V, 4.5mA power
- $7/1K qty

RF RX

- -102dBm Sensitivity @ 4800 baud
- Use with any OOK/ASK TX
- Super-regen design
- 5V, 4.5mA power
- $2.4/1K qty



RCR-RP

www.radiotronix.com 1.886.814.4446

WIRELESS UART TO UART COMMUNICATIONS



Frequency band	902-928 MHz
DTS Mode	
Channels	32
Max RF Data Rate (effective)	94.6 kbps
TX Power	+11 dBm**
Max Sensitivity	-104 dBm***
LP Mode	
Channels	84
Max RF Data Rate (effective)	19.2 kbps* (bi-directional)
TX Power	+5 dBm
Max Sensitivity	-106 dBm***
Power	
V_{DD}	2.7-3.6 V
TX I_{DD} - LP Mode	28mA**
TX I_{DD} - DTS Mode	57mA @ +11 dBm**
RX I_{DD}	20mA**
Sleep/ Standby	100µA/ 850µA
Temperature	-20 to +70°C

WI.232DTSMODULE
Also available in 868MHz for Europe

Radiotronix

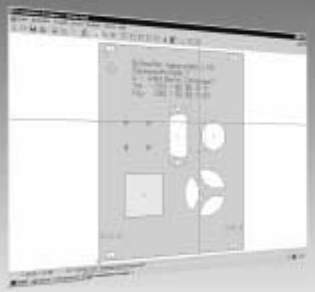
MOUSER ELECTRONICS
Mouser Order Code: 509-Wi.232DTS
www.mouser.com/radiotronix

www.radiotronix.com 405.794.7730 866.814.4446

Front Panels?

Unrivaled in price and quality for small orders

Download the free ›Front Panel Designer‹ to design your front panels in minutes





Order your front panels online and receive them just in time

www.frontpanelexpress.com

NEW

RS232 to TCP/IP

- *TCP/com*™ v2.0, RS232 to TCP/IP software. Plus TCP/IP to RS232.
- *WinWedge*™ RS232 or TCP/IP data direct into any Windows app. - Excel, Access, etc.

TALtech

Free 30 day evals at
www.taltech.com

CUSTOM MEMBRANE KEYBOARDS / SWITCHES



- 1 TO 2 WEEKS TURNAROUND
- VERY COMPETITIVE PRICING
- Ex.: (5) 4-switch keyboards for $395.00
- PCB backed switches
- Custom metal backplates/assemblies
- Electronic assemblies/graphic overlays
- Electronic file transfer capabilities

Picofab Inc.

4780B Blvd. Henri-Bourassa
Charlesbourg, Quebec, Canada G1H 3A7
Tel: (418) 622-5298 • Fax: (418) 622-9996
Email: sales@picofab.net

PRINTED CIRCUIT BOARDS

QUALITY PRODUCT
FAST DELIVERY
COMPETITIVE PRICING

PROTOTYPE through PRODUCTION

- UL approved
- Single & Double sided
- SMOBC
- LPI mask
- Through hole or SMT
- Nickel & Gold Plating
- Routing or scoring
- Electrical Testing
- Artwork or CAD data
- Fast Quotes
- Flex Circuits

SPECIAL OFFER:
10 pcs (3days) 1 or 2 layers $249
10 pcs (5days) 4 layers $695
(up to 30sq. in.ea.) includes tooling, artwork, L.P.I. mask & legend

PULSAR, INC

9901 W. Pacific Ave. Franklin Park, IL 60131 • Phone 847.233.0012
Fax 847.233.0013 • Modem 847.233.0014 • www.pulsar-inc.com • yogii@flash.net

Weeder Technologies

www.weedtech.com DATA SHEETS

RS-232 Stackable

1710-B Brighton Cove, Ft Walton Beach, FL 32547 850-863-5723



Digital I/O • Analog Input • Analog Output • Solid State Relays • Stepper Motor Controllers • Pulse Counters • Pulse Timers • Frequency Measurement • RPM Measurement • Duty Cycle Measurement • Period Measurement • Projectile Velocity Measurement • Third-Party Peripheral Interface • Data Logging

Download Windows software

Plug end-to-end
Stack 32 modules on the same RS-232 cable.

Data Acquisition & Control

...simple as a laptop.

The Mosaic Handheld Customizable Instrument



- Graphics Display & Keypad
- Digital, Analog & Serial I/O
- Up to 1MB Flash, 512K RAM
- Built-in Personality Wiring Board
- Holds up to 4 I/O Wildcards
- 6 AA NiMH batteries, Built-in Charger
- DB-25 Connector

The Mosaic Handheld integrates a C-programmable computer, user interface, charger, and expansion I/O bus in a rugged enclosure so you can rapidly create your custom instrument.

Mosaic Industries Inc.
tel: 510-790-1255 fax: 510-790-0925
www.mosaic-industries.com

WWW.CONITEC.NET

$333 COMPLETE KIT

GALEP-4 UNIVERSAL PROGRAMMER

Introducing a pocket programmer with true **Universal Output**



Latest generation Galep-4 uses ASIC universal pin technology for each pin of 40 pin ZIF-socket. **5200+ device library**/lifetime free updates. Programs 8/16-bit EPROM'S, EEPROMs, 0-Power RAM, FLASH, Serial EEPROM's, GAL, PALCE, microcontrollers such as 87/89xxx, PIC, AVR, ST62, etc. Low voltage devices down to 1.3V. No adapter required for DIL devices. 8 Hrs. operation on battery (AC charger included). Runs **WIN 98,NT,ME, 2000,XP** with Hex/Fuse Editor.

Remote control from other apps, (e.g. VisualBasic). Substitutes high priced universal programmers e.g. ALL-11 (HILO) or LAB-T00L-48 (ADVANTECH) Providing virtually matching performance at only 1/3-1/5 the price. Info, orders, softwr : **619-702-4420**

ISO 9001 CERTIFIED 9001 PROFESSIONAL TOOLS

Info, downloads: www.conitec.net

CONITEC DATASYSTEMS - 1951 4TH AVE, SUITE 301 - SAN DIEGO, CA 92101 - TEL: 619-702-4420 FAX: 619-702-4419

Low Cost Embedded OEM

Modem & Wireless Modules



Low Cost 2400 baud—56K V.92 MNP4 Modems
33K baud 902-926 MHz Wireless Modules

- 300 – 56K baud V.92
- V.42 - V.42bis - MNP4
- Global Compliance—3KV
- FCC, CE and TBR21 approvals
- AT command set
- SocketModem™ compatible
- Low Power—35mA
- Small Footprint 1.045" x 2.54"
- Wireless Modules—902-928MHz - 33Kb
- Auto Error Detection & Flow Control
- Modules are H/W & S/W compatible
- Custom antennas
- USB 2.0 Modules—Adapt legacy RS-232 DB9 to USB
- Other modules coming soon!

www.copelandcommunications.com

CCi COPELAND COMMUNICATIONS, INC. (614) 475-1690

TwinText™

STRAIGHTFORWARD SOURCE CODE DOCUMENTATION

DOCUMENT ALL YOUR EMBEDDED CODE WITH A SINGLE TOOL

- C/C++, Assembler, BASIC, all their variants, plus 25 additional programming languages
- Easy to use: **NO** tags, XML, or specific comment boxes
- Automatic and scriptable code documentation and application help generation

⬇Download your 30-day trial

WWW.PTLOGICA.COM

ALL ELECTRONICS CORPORATION

Electronic and Electro-mechanical Devices, Parts and Supplies. Wall Transformers, Alarms, Fuses, Relays, Opto Electronics, Knobs, Video Accessories, Sirens, Solder Accessories, Motors, Heat Sinks, Terminal Strips, L.E.D.S., Displays, Fans, Solar Cells, Buzzers, Batteries, Magnets, Cameras, Panel Meters, Switches, Speakers, Peltier Devices, and much more....

www.allelectronics.com

Free 96 page catalog

1-800-826-5432

Give the Customer More!

If your company has a physical storefront, you may qualify to become a *Circuit Cellar* distributor.

***Circuit Cellar* will:**

- Entice customers to stay in your store longer
- Attract repeat walk-in business
- Serve as a useful product demonstration tool

Contact: Sean Donnelly
Associate Publisher/Director of Sales
sean@circuitcellar.com

Order online at: www.melabs.com

microEngineering Labs, Inc.

Development Tools for PIC® Microcontrollers

Phone: (719) 520-5323
Fax: (719) 520-1867
Box 60039
Colorado Springs, CO 80960

Serial Programmer for PICmicro® $119.95



Includes:
Programmer
9-pin Serial Cable
AC Power Adapter
ZIF Adapter for 8 to 40-pin DIP
Software for Windows 98/Me/NT/2K/XP

Optional USB Adapter $39.95

LAB-X Experimenter Boards



Pre-Assembled Board
Available for 8, 14, 18, 28, and 40-pin PIC® MCUs
2-line, 20-char LCD Module
9-pin Serial Port
Sample Programs
Full Schematic Diagram

Pricing from $69.95 to $349.95

BASIC Compilers for PICmicro®



Easy-To-Use BASIC Commands
Windows 9x/Me/2K/XP Interface

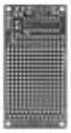
PICBASIC™ Compiler $99.95
BASIC Stamp 1 Compatible
Supports most 14-bit Core PICs
Built-In Serial Comm Commands

PICBASIC PRO™ Compiler $249.95
Supports All PICmicro® MCU Families
Direct Access to Internal Registers
Supports In-Line Assembly Language
Interrupts in PICBASIC and Assembly
Built-In USB, I2C, RS-232 and More
Source Level Debugging



EPIC™ Parallel Port Programmer starting at $59.95

PICPROTO™ Prototyping Boards



Double-Sided with Plate-Thru Holes
Circuitry for Power Supply and Clock
Large Prototype Area
Boards Available for Most PIC® MCUs
Documentation and Schematic

Pricing from $8.95 to $19.95

See our full range of products, including books, accessories, and components at:

www.melabs.com

CIRCUIT CELLAR

Test Your EQ

Edited by David Tweed

Problem 1—What sort of filter is this? What applications is it good for?

```
int filter (int sample)
{
  static int previous;
  if (abs(sample - previous) >1) previous = sample;
  return previous;
}
```

Problem 2—What sort of filter is this?

```
int filter (int sample)
{
  static int previous;
  previous = (previous >> 2) + (sample >> 2);
  return previous;
}
```

Problem 3—What is a median-value filter? For what sort of applications is it well suited?

Problem 4—Write a C preprocessor macro that produces a mask for a bit field, given the numbers of the leftmost and rightmost bits in the field. In other words, `FIELD_MASK(7,4)` should produce the value 0x00F0.

Contributed by David Tweed

What's your EQ?—The answers are posted at www.circuitcellar.com/eq.htm

You may contact the quizmasters at eq@circuitcellar.com

Circuit Cellar
back issues are available as
searchable archives
on
CD-ROM.
Get the ones you missed!
CD-ROM #9 2004 Issues 162-173
CD-ROM #8 2003 Issues 150-161
CD-ROM #7 2002 Issues 138-149
CD-ROM #6 2001 Issues 126-137
Order online
www.circuitcellar.com
or call (860) 875-2199

Micro-Server
Internet Appliance Engine
SOM-5282EM

• Coldfire 66/80 Mhz CPU
• 10/100BaseT Fast Ethernet
• 16 GP I/Os & 8 Channel A/D
• Programmable in Java™ or C
• Telnet, FTP, and HTTP Servers
• 3 Serial Ports, CAN 2.0B & SPI
• uClinux with Real Time support
• 4.5 MB Flash & up to 16 MB RAM
• Typical Power Consumption < 2 Watts
• Real Time Clock & Nonvolatile Memory
• Carrier/Socket Board & Power Supply Available
• Small, 144 pin SoDIMM form factor (2.66" x 1.5")
The SoM-5282EM is a System on a Module, based on the Freescale MCF5282 Processor. This 32-Bit processor runs uClinux making it extremely easy to create smart Network/Internet capable devices, with Data Acquisition and Control properties. If Real-Time processing is required we can optionally provide RTAI Real-Time extensions. Write sophisticated network applications in days instead of months using standard GNU tools. Single Unit pricing starts at $150. Optional Carrier/Socket board, Enclosure, and Power Supply also available. For additional information visit, www.emacinc.com/som/.
Since 1985 OVER 20 YEARS OF SINGLE BOARD SOLUTIONS
EMAC, inc.
EQUIPMENT MONITOR AND CONTROL
Phone: (618) 529-4525 • Fax: (618) 457-0110 • Web: www.emacinc.com

INDEX OF ADVERTISERS

The Index of Advertisers with links to their web sites is located at www.circuitcellar.com under the current issue.

Page		Page		Page		Page	
57	Acon, Inc.	95	Echelon Corporation	13	Lemos International	90	Radiotronix
2	Advanced Circuits	87	EE Tools (Electronic Engineering Tools)	33	Maxstream	86	Radovan Robotics
86	ADM Designs, LLC	89	Elsevier, Inc.	88	MCC (Micro Computer Control)	89	Reach Technology, Inc.
36	Advanced Embedded Technologies	93	EMAC, Inc.	90	Mental Automation	49	Renesas
92	All Electronics Corp.	84	Epsilon Electronic Mfg	37	Micro Digital	71	Robonexus Conf & Expo
80	AmpFlow	74	ESC Boston	92	microEngineering Labs, Inc.	50	Saelig Company Inc.
86	AP Circuits	25	ExpressPCB	19	Micromint, Inc.	3	Scott Edwards Electronics, Inc.
90	Aquila Creative LLC	84	FDI-Future Designs, Inc.	89	MJS Consulting	88	Sealevel Systems
79	Arcom Control Systems	89	FieldServer Technologies	91	Mosaic Industries, Inc.	5	Sierra Proto Express
7	Atmel	91	Front Panel Express	55	Mouser Electronics	9	Silicon Laboratiroes, Inc.
85	Bagotronix, Inc.	86	Grid Connect	90	Mylydia, Inc.	87	Smiley Micros
36	Bellin Dynamic Systems, Inc.	85	Hagstrom Electronics	C2	NetBurner	86	Systronix
32	Beta Layout Ltd.	47	HI-TECH Software, LLC	21	Noritake	91	TAL Technologies
41	Bitscope Designs	29	Holmate/Holtek Semiconductor, Inc.	37	Nurve Networks LLC	C3	Tech Tools
90	Bitwise Systems	81, 89	ICOP Technology, Inc.	88	OKW Electronics, Inc.	16, 17	Technologic Systems
42	CadSoft Computer, Inc.	19, 89	IMAGEcraft	85	Ontrak Control Systems	87	Technological Arts
87	Carl's Electronics	90	Intec Automation, Inc.	87	PCB Fab Express	89	Tern, Inc.
84	CCS, Inc.	86	Integrated Knowledge Systems	C4	Parallax, Inc.	32	Tianma Microelectronics
92	Conitec	85	Intrepid Control Systems	31	Philips ARM Design Contest	85	Trace Systems, Inc.
92	Copeland Communications, Inc.	33,85	Ironwood Electronics	84	Phytec America LLC	86	Triangle Research Int'l, Inc.
65	CWAV	73	Jameco	87	Phyton, Inc.	88	VNISource Design
1	Cypress MicroSystems	36	Jeffrey Kerr, LLC	91	Picofab, Inc.	91	Weeder Technologies
84	Decade Engineering	64, 86	JK microsystems, Inc.	92	PTLogica	86	Xytronix Research & Design, Inc.
80	DesignCon East	56	Keil Software	91	Pulsar, Inc.	88	Zagros Robotics
84	DLP Design	37	LabJack Corp.	63	R4 Systems, Inc.		
82	Dynon Instruments	85	Lakeview Research	66	R.E. Smith		
64	Earth Computer Technologies	88	Lawicel HB	39, 58	Rabbit Semiconductor		

Preview of September Issue 182
Theme: Signal Processing

Replacing a Legendary Controller: Build a Functional Equivalent to the CDP1861

Digital Decoding Simplified: Sequential Exact-Frequency Goertzel Algorithm

Digital Watermarking

High-Performance Square Wave Source: Build a Digitally Controlled Signal Generator

Hardware Synthesis with VHDL (Part 2): VHDL in Action

Signal Generation Solution: Build an Inexpensive RF Signal

Strong Encryption for PICs (Part 2): Triple-DES Implementation

HappyTerminal: A ToothPIC-Based Terminal Emulator

APPLIED PCs Throwing Ethernet Frames

FROM THE BENCH Capacitive Sensing Technology

SILICON UPDATE Batteries Not Included

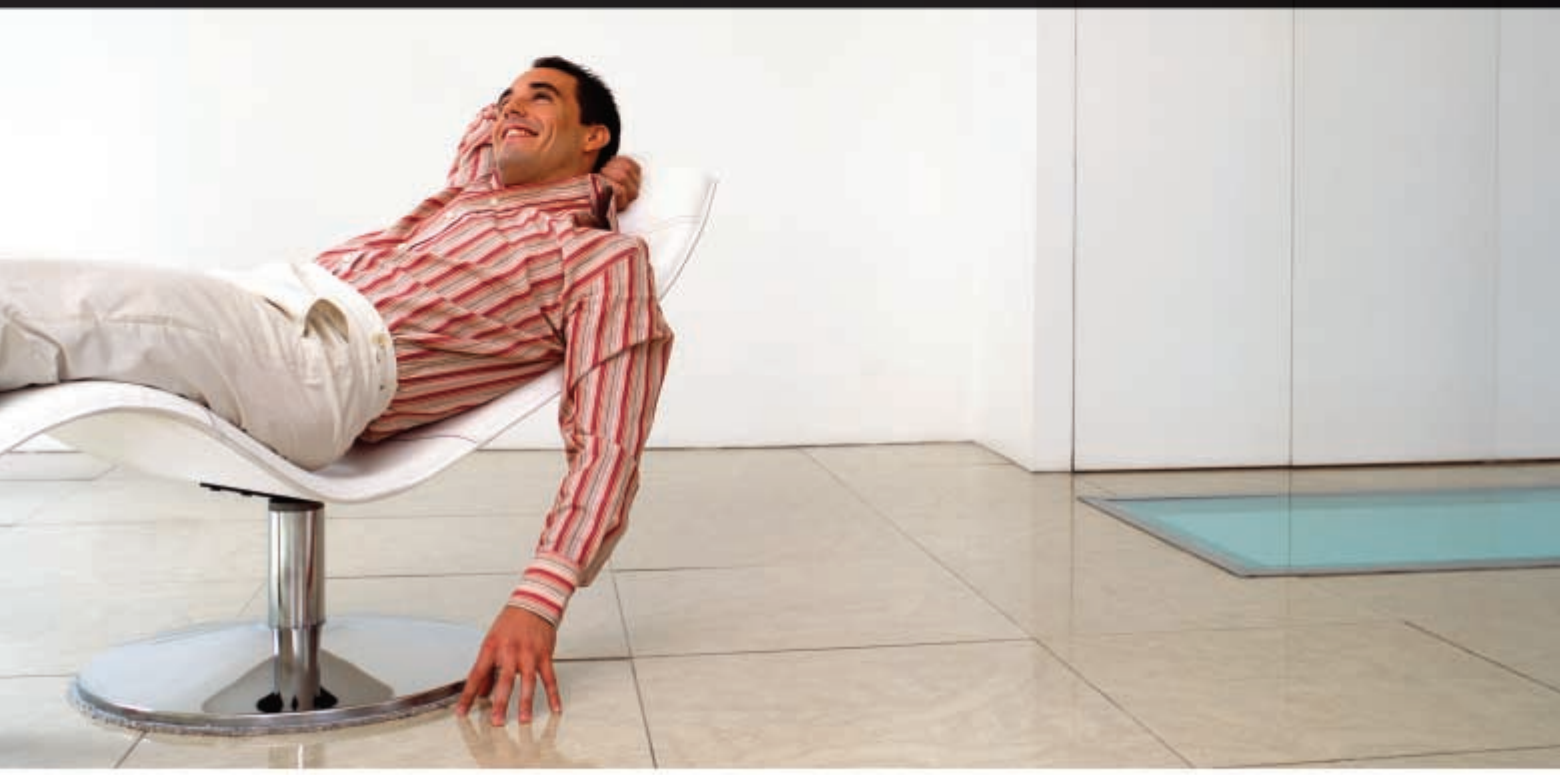
ATTENTION ADVERTISERS

October Issue 183 Deadlines

Space Close: August 10
Material Close: August 19
Theme:
Analog Techniques
Bonus Distributions:
ARM DevCon,
PCB East, and RoboNexus

Call Shannon Barraclough
now to reserve your space!
860.872.3064
e-mail: shannon@circuitcellar.com

GET YOUR COSTS
[TIME-TO-MARKET, PRODUCTS, LIFE]
IN CONTROL.





If you design control systems or devices, Echelon can help lower both your development and product costs.

Our LonWorks® platform is the worldwide leader for networking anything you want to control — lights, appliances, meters, sensors, actuators, meters, and more. The new Mini EVK lets you evaluate the LonWorks platform, develop new devices, write and load new applications — all for just $345. And, it's available with power line communications — so you can get wired without needing to pull any cables.

ECHELON®

www.echelon.com/mini

©2005 Echelon, Echelon, LonWorks, and the Echelon logo are registered trademarks of Echelon Corporation registered in the United States and other countries.



PRIORITY INTERRUPT

by Steve Ciarcia, Founder and Editorial Director

Risk and Opportunity

I frequently get e-mails from readers asking my opinion on career moves or entrepreneurial ventures. The correspondents aren't long-haul truck drivers in Alaska or machine operators working in Manhattan. Instead, the messages I get come from a project engineer at United Technologies or a software designer at Itty Bitty Controller Company, USA. The questions are typically something like should they go get an M.S.E.E. that would help them climb the corporate ladder, or should they mortgage their house and try to start a company selling the little embedded controller they designed in the basement.

Engineering, or more importantly an engineering mentality, can be a double-edged sword. A truck driver or a machine operator is trained to do a task, and accomplishing it satisfies their self-esteem. It seems to be the opposite for engineers. If my mail is any indication, I get the feeling that the more engineers accomplish on the job, the less happy they are about whether they should be doing it that way. The conflict arises because a corporate engineering career is primarily about creating solutions and designing inventions for the corporation. Our culture is all about rewarding people who are entrepreneurial and capitalize on invention for their own benefit. Can you see the dilemma?

I'm very willing to admit that there are lots of people smarter than me. Fortunately, having entrepreneurial success in an engineering career isn't just about being smart. In my opinion, it's mostly about risk and opportunity. I would never say to someone, "Quit your job. If Bill Gates can do it, so can you." That's insane. Before giving up the cushy day job that supports a wife and four kids, you might want to find a less risky way to assess whether the world is interested in your invention and talents. That's where opportunity becomes important.

I didn't wake up one morning and decide I was going to run a magazine or manufacture electronic products. I woke up one morning and said I'm interested in these new things called microcontrollers, even if the place where I work has no interest and considers them forbidden fruit. Before I published my first design project—an 8008-based vector graphics display generator—no one outside the bowels of Control Data Corporation had a clue who Steve Ciarcia was. It didn't take too many more articles before I had to start a manufacturing company just to satisfy the production demand for these projects.

While I suppose the book deals and interviews that come from bugging out of a wedding and having the whole world searching for you for a week might appeal to a few, it's a high-risk venture. A safer way to achieve your next engineering career move might simply be to publish something and put it in front of the world.

The testimonial evidence of *Circuit Cellar* as a career builder is overwhelming. I received this note just this week:

It seems that the face-to-face contact is still a highly ranked criterion in customers selecting their consulting partners. It certainly provided a lot of instant credibility during initial meetings to be able to pull out a copy of Circuit Cellar *and show the articles I had written (especially when the subject of the article matched that of the product to be designed). I found through this process that a surprising number of people read* Circuit Cellar. *Following the article release, I also received many phone calls and e-mails from colleagues and friends I hadn't heard from in a while. I am also now asked regularly by some when my next article is coming out!*

Eric Gagnon

Publishing doesn't only mean print articles. Contest winners and Distinctive Excellence designees can consider themselves "published" when they see their projects posted on www.circuitcellar.com. And since contest sponsors often put everything on their own sites, many people can have the added distinction of being able to say they have one of their designs published on www.cypressmicro.com, www.atmel.com, etc. You can't buy this advantage.

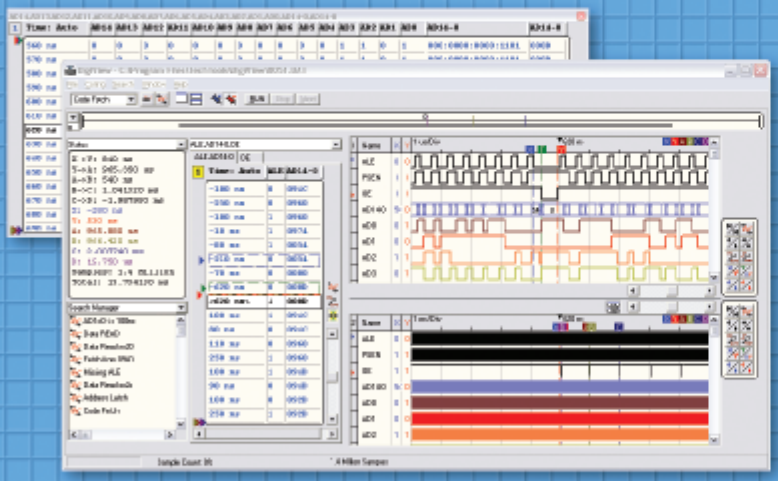
Through it all I'd like to think that I've stayed true to the brotherhood. When you participate in a *Circuit Cellar* contest or publish an article with us, our contractual fine print isn't just about what you give away. It's also about what you keep. Unlike many other contests and publications, I want designers and authors to know that when you do things with us, you retain the intellectual property ownership. If a company sees your posted project or reads your article and wants more, they have to deal with you. IP ownership along with a great publishing vehicle succeeded in creating a fine career for me. If you are willing to take some risk investigating new opportunities, perhaps it can also work for you too.



steve.ciarcia@circuitcellar.com

DigiView

100 MHz, 18 Channel, Portable Logic Analyzer!

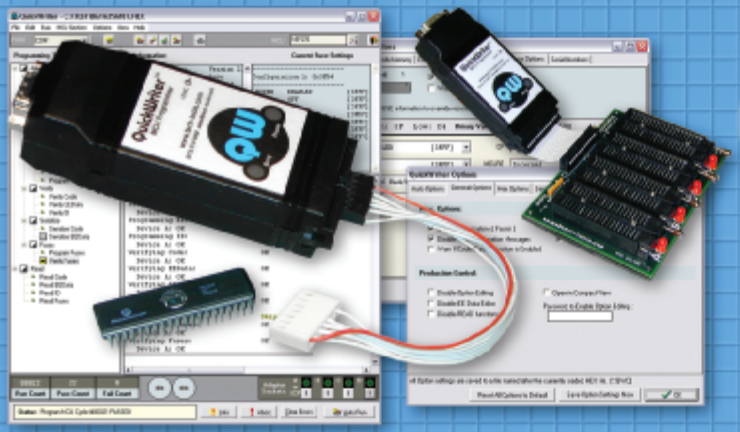


Auto Hardware Compression Captures from 128K to 30 Billion Samples @10ns resolution!

only $499.00

- Save up to 5 minutes of Data @ 10ns.
- Edge and Pattern Triggers.
- Real-time Hardware Compression.
- USB Powered for Portability.
- Includes Capture & Analysis Software.
- Pattern Searches with Match, Duration & Animation.
- Drag & Place and Drag & Snap Markers.
- Waveform Next/Previous/Nearest Edge snapping.
- Multi-Signal Data Tables with Link Groups.
- Split Waveform Views with Link Groups.

Download and try the Software!



QuickWriter

Multi-Function PICmicro® MCU Programmer In-Circuit & Gang Operation

only $199.00

- Supports 12, 14, 16 & 18 Series PICmicro MCUs.
- Edit Code, EE Data, IDs and Configuration Fuses.
- Auto or Manual Serialization in Code or EE Data.
- Firmware auto synchronizes to Software version.
- Control Files protect Option Settings.
- Accepts Command-Line Parameters.
- Tests and Monitors Voltages.
- Control Signal for Self-Powered Circuits.
- Single and 4 Gang Adapters available.
- Includes Cables, Software & U.S. Power Supply.

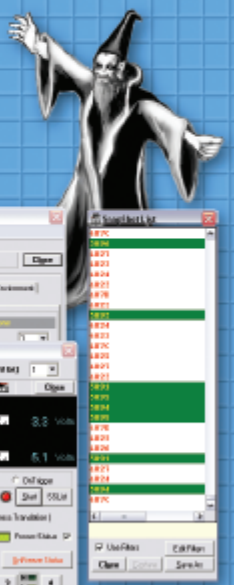
See Our Other Embedded Development Tools



EconoROM III & FlexROM III

EPROM & FLASH Emulation at its best!

from $179.00



TechTools

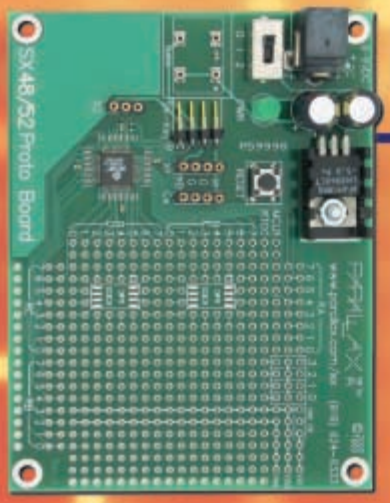
www.tech-tools.com

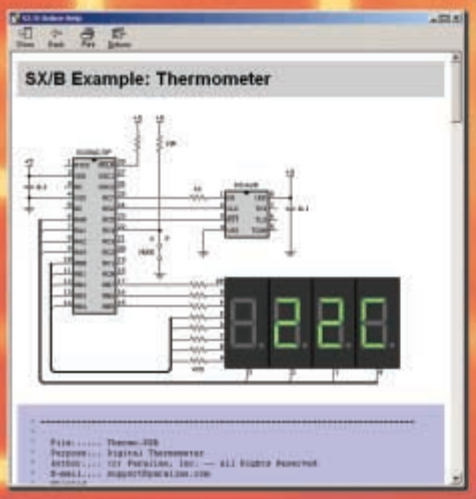
(972) 272-9392 • sales@tech-tools.com

Copyright © 2005 TechTools • DigiView, FlexROM, EconoROM and QuickWriter are trademarks of TechTools • PICmicro is a registered trademark of Microchip Technology Inc.

SX/B

and the SX48 Proto Board





Our new **SX48/52 Proto Boards** come complete with the SX microcontroller, voltage regulator, prototype area, and resonator socket. **Order part #45300 for the SX48 Proto Board or part #45304 for the SX52 Proto Board. Both are only $9.95 each!**

If assembly language isn't your favorite, use our free **SX/B BASIC compiler**. The **entirely free** SX/B compiler expedites the programming of the SX18/20/28/48/52 by providing a simple yet robust high-level language for the 50-MIPS 8-bit SX microcontroller. SX/B supports in-line assembly language and has BASIC commands for serial, I²C, 1-Wire, R/C circuits and more.
For more details on the SX/B compiler, or to get your free download of this software, visit www.parallax.com/sx.

To program SX chips, you'll need the **Parallax SX Tech Toolkit PLUS (#45180; $99.00)**, which includes the SX-Key® programmer/debugger tool, the SX Tech Board, manuals, and two SX chips.

Order online at www.parallax.com/sx or call the Parallax Sales Department toll-free at 888-512-1024 (Monday-Friday, 7am-5pm, PT).



PARALLAX inc

www.parallax.com

SX-Key® is a registered trademark of Parallax, Inc. SX/B, Parallax, and the Parallax logo are trademarks of Parallax, Inc.