DESIGN: Create a Virtual Alarm System
LOCATION: Brazil
PAGE: 36

DIAGNOSTICS: The CAN Protocol & OBD-II
LOCATION: United States
PAGE: 60

INSIGHT: Heat Management Essentials
LOCATION: France
PAGE: 68

CIRCUIT CELLAR

THE WORLD'S SOURCE FOR EMBEDDED ELECTRONICS ENGINEERING INFORMATION

JUNE 2011
ISSUE 251

COMMUNICATIONS

Build a Custom RFID Checkout System

Hardware Development with a USB AVR & CPLD

Electronics System Safety

Thermal Analysis Tips

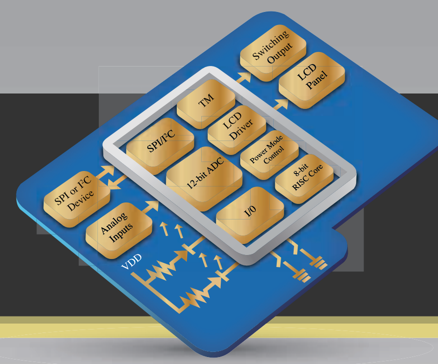
PLUS

An MCU-Based, DIY Network Controller

// Monitor & Control Networked Devices
// Acquire & Display Internet-Based Data
// Access a DNS Activity Log
// Wi-Fi Network Management

0 6>
7 25274 75349 9
$7.50 U.S. ($8.50 Canada)
www.circuitcellar.com

HOLTEK
www.holtek.com



All In One

Ultra Low Power Consumption - Highly Functionally Integrated Flash MCU Solutions

The Holtek Flash MCU range of devices encompass an extensive range of peripheral functions making them suitable for use in a diverse application area such as health care products, instrumentation products, household appliances, industrial control, consumer products, automotive peripheral products to name but a few. With several range types also including Holtek's unique TinyPower™ technology, the ability is provided to meet the demands of today's environmentally conscious products.

HT68Fxx	Internal Multi-function Timer Modules. IRC with four frequency selections. Multi-function I/O pins with re-mapping function. 4 LCD SCOM outputs for direct driving of LCD panels. Multiple Communication Interfaces.
HT66Fxx	Includes all the HT68Fxx functions with an additional 12-bit ADC and internal reference voltage source.
HT69Fxx	Uses Holtek's TinyPower™ ultra low power technology. Includes internal R-type/C-type LCD driver.
HT67Fxx	Includes all the HT69Fxx functions with an additional Internal High Speed 12-bit ADC and internal reference voltage source. Includes dual SPI or I²C interfaces and internal Data EEPROM.

STD Flash MCU HT66/67/68/69Fxx Series

- Wide Operating Voltage Range: 2.2V~5.5V
- Multiple Low-power Operating Modes
- Fast Wake-up Function
- Industrial Specifications: -40°C ~ +85°C
- Multiple External Communication Interfaces
- Internal Accurate High/Low Speed Oscillators
- High Noise Immunity and Excellent ESD protection

Touch Flash MCU | STD Flash MCU | STD 8051 Flash MCU | USB STD Flash MCU | 32bit STD Flash MCU | Enhanced OTP MCU
TinyPower™ MCU | Power Management | UART MCU | Phone MCU | EEPROM | WLED Backlight

HOLTEK SEMICONDUCTOR INC.

Holtek Semiconductor (USA), Inc.
46729 Fremont Blvd., Fremont, CA 94538 Tel: (510)252-9880 Fax: (510)252-9885
E-mail: sales@holtek.com http://www.holtek.com

TASK MANAGER

Silicon Valley Review

At the 2011 Embedded Systems Conference in Silicon Valley, I was pleased to chat with two of our most well-known authors, Larry Cicchinelli and Richard Wotiz. As you know, Larry wrote our most recent book, *Assembly Language Essentials*, which was on display at our booth. The busy conference floor was not the best place to sell books—although people bought copies anyhow—but that wasn't our goal. Our aim was to introduce the book to engineers and get their thoughts on Larry's interesting approach to writing about Assembly. The most common question about the book was "Which processor?" Our reply was that the book isn't processor-specific. Larry refers to the architecture of a fictional processor with its own hardware and instruction set, which many people found quite intriguing.

Richard Wotiz stopped at the booth soon after Larry. You're familiar with Richard because we published seven of his articles between July 2003 and January 2011. His eighth article, "Construct a Multifunctional Network Controller," appears on page 20. Like our columnist Robert Lacoste (see his primer on heat management, p. 68), Richard placed well in numerous international design challenges. And, very soon, Richard will follow Robert's lead and become a *Circuit Cellar* columnist. Keep an eye out for his first column, which will likely appear in Issue 254.

When I wasn't discussing Larry's book or chatting with readers, I was editing articles for this issue. Like the groups of people who stopped by our booth, this issue's authors are a varied bunch: professional engineers, innovative university students, and seasoned columnists.

George Novacek starts the issue with a discussion of test equipment and safety-critical systems. These are important topics every engineer should take seriously.

Starting on page 28, Jigar Shah and Kevin Yang describe the RFID checkout system they planned and designed at Cornell University. The article covers RFID technology, MCU-based circuit design, and custom antenna platform construction.

Interested in virtual systems? Check out Custodio Gomes Barcellos's article, "Virtual Sensing," in which he describes a virtual alarm system design (p. 36).

If you're intrigued by low-cost hardware development, consider the Chameleon platform Doug Commons describes on page 44. He covers development with a USB AVR and a CPLD.

On page 60, Jeff Bachiochi begins an article series on vehicle diagnostics and the interesting topic of the CAN protocol and OBD-II. In this first article he covers everything from data formatting to physical interfacing.

As the aforementioned authors understand, thermal measurement falls into the "need-know-know" category. That's why pragmatist Ed Nisley weighs in on the topic. He covers plastic extruder thermal analysis and applies it to a design (p. 52).

You have plenty of content to get through until next issue. Be sure to let us know what you think.

cj@circuitcellar.com

C. Abate

CIRCUIT CELLAR®

THE WORLD'S SOURCE FOR EMBEDDED ELECTRONICS ENGINEERING INFORMATION

FOUNDER/EDITORIAL DIRECTOR
Steve Ciarcia

EDITOR-IN-CHIEF
C. J. Abate

ASSOCIATE EDITOR
Nan Price

CONTRIBUTING EDITORS
Jeff Bachiochi
Robert Lacoste
George Martin
Ed Nisley

NEW PRODUCTS EDITOR
John Gorsky

PROJECT EDITORS
Ken Davidson
David Tweed

STAFF ENGINEER
John Gorsky

PUBLISHER
Hugo Van haecke

ASSOCIATE PUBLISHER
Shannon Barraclough

CUSTOMER SERVICE
Debbie Lavoie

CONTROLLER
Jeff Yanco

ADMINISTRATIVE COORDINATOR
Valerie Luster

ART DIRECTOR
KC Prescott

GRAPHIC DESIGNERS
Grace Chen
Carey Penney

Circuit Cellar's mission is to collect, select, and disseminate need-to-know information around the world in the fields of embedded hardware, embedded software, and computer applications. Circuit Cellar uses an assortment of print and electronic content-delivery platforms to reach a diverse international readership of professionals, academics, and electronics specialists who work with embedded, MCU-related technologies on a regular basis. Our aim is to help each reader become a well-rounded, multidisciplinary practitioner who can confidently bring innovative, cutting-edge engineering ideas to bear on any number of relevant tasks, problems, and technologies.

ADVERTISING

800.454.3741 • 978.281.7708 • www.circuitcellar.com/advertise

ADVERTISING REPRESENTATIVE
Peter Wostrel
Strategic Media Marketing, Inc.
1187 Washington St., Gloucester, MA 01930 USA
800.454.3741 • 978.281.7708
peter@smmarketing.us • www.smmarketing.us
Fax: 978.281.7706

ADVERTISING COORDINATOR
Kim Hopkins
E-mail: khopkins@circuitcellar.com

Cover photography by Chris Rakoczy—Rakoczy Photography
www.rakoczyphoto.com

PRINTED IN THE UNITED STATES

CONTACTS

SUBSCRIPTIONS
Information: www.cc-access.com, E-mail: subscribe@circuitcellar.com
Subscribe: 800.269.6301, www.cc-access.com, Circuit Cellar Subscriptions, P.O. Box 5650, Hanover, NH 03755-5650
Address Changes/Problems: E-mail: subscribe@circuitcellar.com

GENERAL INFORMATION
860.875.2199, Fax: 860.871.0411, E-mail: info@circuitcellar.com
Editorial Office: Editor, Circuit Cellar, 4 Park St., Vernon, CT 06066, E-mail: editor@circuitcellar.com
New Products: New Products, Circuit Cellar, 4 Park St., Vernon, CT 06066, E-mail: newproducts@circuitcellar.com

AUTHORIZED REPRINTS INFORMATION
860.875.2199, E-mail: reprints@circuitcellar.com

AUTHORS
Authors' e-mail addresses (when available) are included at the end of each article.

CIRCUIT CELLAR®, THE MAGAZINE FOR COMPUTER APPLICATIONS (ISSN 1528-0608) is published monthly by Circuit Cellar Incorporated, 4 Park Street, Vernon, CT 06066. Periodical rates paid at Vernon, CT and additional offices. **One-year (12 issues) subscription rate USA and possessions $45, Canada/Mexico $60, all other countries $63. Two-year (24 issues) subscription rate USA and possessions $80, Canada/Mexico $110, all other countries $116.** 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.

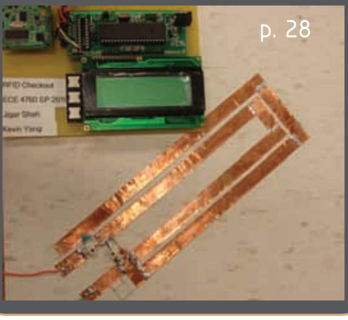
INSIDE ISSUE 251

June 2011 • Communications

DIY Network Controller

p. 20

Build an RFID Checkout System



p. 28

Virtual Alarm

p. 36

Thermal Analysis

p. 52

ENHANCED RJ45-ECS ETHERNET CONNECTOR SYSTEM

The **RJ45-ECS** Ethernet connector system is a field-replaceable outdoor-rated and shielded Ethernet connection system for high-speed 10BaseT, 100BaseT, and 1000BaseT networks. The patented connector system incorporates a molded strain relief into a cable design, which enables equipment to quickly and easily disconnect and reconnect. This feature reduces and can even eliminate damage caused to the connector components from the continuous coiling, curving, or straining of the cable.

The RJ45-ECS features a shielded RJ-45 receptacle that is mounted in a UV-protected housing along with a mating cable gland that passes an RJ-45 connector and seals the cable to IP67 requirements. The IP67 sealing ability provides a reliable connection, even in harsh environments.

The connector system includes a shielded Cat5e cable with connectors that provides shielding against external EMI for consistent data communications. It also includes a quick disconnect feature that enables equipment to be easily replaced and minimizes customer and field downtime spent servicing the equipment.

Contact a Laird Technologies distributor for pricing information.

Laird Technologies, Inc.
www.lairdtech.com

STAMP-SIZE LCD/TFT-READY ARM9 BOARD

The **Stamp9G45** is the latest CPU module to be added to taskit's Stamp range of products. The energy-efficient embedded board utilizes less than 100 mW and has a circuit board measuring only 53.6 mm × 38 mm × 6 mm. The ARM9 board is based on Atmel's AT91SAM9G45 32-bit ARM controller.

The Stamp9G45 runs at 400 MHz and comes equipped with 128-MB DDRAM, 128-MB NAND flash, and an integrated microSD card slot. There are 200 processor pins leading out of two 100-pin industrial Hirose FX-8 connectors. Some of the supported interfaces and peripheral devices include an LCD/TFT controller, Ethernet, USB 2.0 host and USB On-The-Go (OTG), USART, SPI, I²C-compatible TWI, JTAG, up to 100 digital I/O ports, and a 16-bit parallel bus.

The Stamp9G45 comes pre-installed with the Linux operating system. A module configured with 128-MB NAND flash and 128-MB DDRAM costs approximately **$145**.

taskit GmbH
www.taskit.de

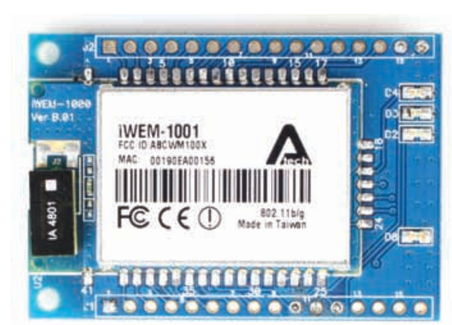
ULTRA-LOW-POWER CONSUMPTION Wi-Fi MODULE

The **iWEM-1001 series** is an ultra-low-power consumption Wi-Fi module with a fast boot-up/connection feature designed for industrial applications. It is the key product to complete last mile M2M wireless connectivity.

iWEM-1001 series Wi-Fi modules are fully integrated Wi-Fi solutions in a compact IC-style SMT packaging. The modules enable engineers to produce products incorporating Wi-Fi connectivity without having RF experience and without months of hardware and software development time. The only hardware requirement for the host system is a simple three-wire UART interface that is the common interface for most microcontrollers.

The series has been designed from inception to maximize battery life and achieve ultra-low-power consumption. The power conservation architecture operates in two modes: Active mode and Sleep mode. Active mode consumes 30 to 200 mA depending on transmit data density. The sleep mode consumes less 0.5 mA. You can maximize the battery life by keeping the module in sleep mode most of time and only waking it up when needed.

iWEM-1001 modules include flexible antenna choices of embedded chip, UFL, and SMA; a general sensor interface for monitoring push buttons, temperature, motion, and acceleration; 11 GPIOs; and pre-installed Atech Wi-Fi Thin-Client application firmware to provide complete networking functionality, infrastructure, and ad hoc mode support. They also offer a simple ASCII command set to configure the module and over-the-air TFTP firmware upgrade capability.



Contact Lemos International for pricing.

Lemos International Co. Inc.
www.lemosint.com

NEW PRODUCT NEWS

Edited by John Gorsky


1st in the Industry
PCB ASSEMBLY PCB ASSEMBLY
PCB ASSEMBLY PCB ASSEMBLY
Full Turnkey Online Quote!
Why suffer from Procurement Headaches!
Go to www.PCBnet.com and click on 'Online Turnkey Quote'.
Just upload your BOM
Define Part Number and Quantity
Enter few specifications for Bare Boards
...and you are DONE.
You will get Full Turnkey Price (Components, Labor & Bare Boards) within MINUTES!
imagineering inc.
SINCE 1985
www.PCBnet.com
847-806-0003 Fax: 847-806-0004
email: sales@PCBnet.com

THREE-TO-SIX-CHANNEL RF POWER SPLITTER

The **PS12000** is a three-to-six-channel DC-to-2-GHz power splitter built on Plessey's 20-GHz complementary bipolar process. It provides a highly linear power splitter for broadband RF or video signals.

The PS12000 provides either three differential output signals or six single-ended outputs. The input signal is initially boosted through an ultra-high linearity 6-dB gain buffer before being split into the three differential output drivers. Two of these output drivers include a 0- to –40-dB AGC stage enabling signal levels to be adjusted as required or effectively switched off. The datasheet describes in detail how the various input and output signals can be matched to meet individual system needs including 75 or 50 Ω. The device's key performance parameters include broadband operation up to 2 GHz, high linearity of 20-dBm IIP3 and 50-dBm IIP2, an 8-dB noise figure and a greater than 40-dB AGC range, along with a gain tracking error of less than 1 dB.

The PS12000 is offered in a 28-lead MLP package and the device is supported by a comprehensive datasheet. An evaluation board is available on request. The PS12000 costs **$5.45** in 1,000-piece quantities.

Plessey Semiconductors Ltd.
www.plesseysemi.com

COMPACT SSD DISK ON CHIP

The **OC177 DOC** is a DiskOnChip solid-state drive that is smaller than a U.S. quarter. It is a NAND drive that features a tiny dimension of only 22 mm × 22 mm × 1.8 mm and capacities up to 32 GB or 64 GB with a standard IDE or SATA host interface.

Marketed as "DOC," the thin, light, robust drive supports a standard integrated development environment (IDE) or serial advanced technology attachment (SATA) host interface and complies with ATA-7 specifications. Current shipping capacity is up to 32 GB with a read/write speed up to 70/40 MBps.

The DOC is designed for high reliability and rugged embedded computing applications where ultra-compact, ultra-low-profile, ultra-high antishock and ultra-high anti-vibration capabilities are required. The DOC is well suited for embedded computers and industrial PCs, various applications (e.g., military and aerospace), POS terminals, and mobile devices.

Contact Foremay for pricing.

Foremay, Inc.
www.foremay.net

PROGRAMMABLE USB WIRELESS MODULES

The **Wixel** is a general-purpose programmable module that features an integrated full-speed USB and a 2.4-GHz radio. Users can load their own custom programs or use precompiled, open-source applications to enable wireless control of a robot, create a wireless microcontroller programmer, establish a wireless sensor network, and more.

To get started with the Wixel, users simply download an application from the 'Net and upload it to the module using its built-in USB bootloader along with Pololu's free configuration software. (No external programming is required.) For example, the wireless serial app can be used to turn a pair of Wixels into a wireless USB/TTL serial link for communication between two microcontrollers or between a PC and a microcontroller.

The Wixel is based on Texas Instruments's versatile CC2511F32 microcontroller, which has an integrated radio transceiver, 32 KB of program memory, 4 KB of RAM, and a USB interface. The Wixel makes 15 general-purpose I/O lines available, including six analog inputs. Its 0.1" pin spacing makes it easy to use with breadboards and perfboards. For those who want to write custom applications in C, the Wixel SDK provides open-source development tools and libraries. Custom programs can also be built based on Pololu's selection of applications.

Wixels (item #1337) cost **$19.95** each.

Pololu Corp.
www.pololu.com

ICs FOR MICROINVERTER, POWER OPTIMIZER, & CHARGE CONTROLLER SYSTEMS

National Semiconductor has introduced 10 new **SolarMagic ICs**, the first in a series developed to reduce cost, improve reliability, and simplify design of photovoltaic systems. Ranging from the industry's first full-bridge gate driver to a micropower voltage regulator, the new ICs are well suited for a variety of photovoltaic electronic applications, including those found in microinverters, power optimizers, charge controllers, and panel safety systems.

The new SolarMagic ICs are the first to meet photovoltaic renewable energy-grade qualification requirements. Each IC is engineered specifically for demanding rooftop environments that range from extreme cold to severe heat, and each passes rigorous testing with enhanced reliability specific to solar requirements. In addition, the ICs ensure long-term operation, and are developed to meet and exceed the 25-year life expectancy of photovoltaic modules.

Collectively provided as a complete design, SolarMagic ICs increase energy harvesting, reduce cost per kilowatt hour, and improve safety in junction boxes and other types of enclosures. Used independently, the ICs provide high-voltage and high-current gate drive for microinverter or power optimizer designs.

The 10 new devices include the SM72441 and SM72442 MPPT controllers, the SM72295 and SM72482 gate drivers, the SM72485 and SM72238 voltage regulators, the SM72240 supervisory reset circuit, the SM72375 comparator, the SM72480 temperature switch, and the SM72501 precision amplifier.

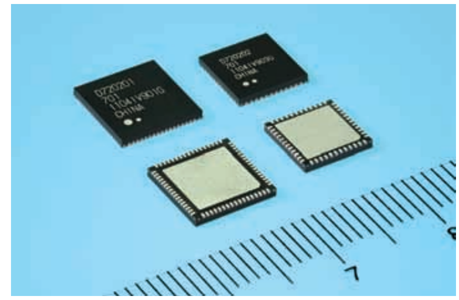
A full set of reference designs and application notes helps the designer quickly develop a complete photovoltaic system. Each reference design includes an evaluation board, a bill of materials, and a schematic.

National Semiconductor Corp.
www.national.com

USB 3.0 HOST CONTROLLERS WITH HIGH TRANSFER SPEED

The **µPD720201** and **µPD720202b** are SuperSpeed USB 3.0 xHCI host controllers that feature fast, effective data transfer speeds and reduced power consumption in low-power mode. The controllers enable you to easily build electronic devices combining high-speed data transfer, compact size, and extended battery life. In addition, Renesas offers device drivers for Microsoft Windows and device drivers with Linux support for the µPD720201 and µPD720202 host controllers at no additional cost.

The controllers feature technology that reduces power consumption when the peripheral device is in the unconnected state. Circuit improvements were made to suppress current leakage when in low-power mode, which decreases power consumption to 4.5 mW, a reduction of 90% compared to Renesas's other products.



The µPD720201 and the µPD720202 are equipped in an 8-mm square QFN package and a 7-mm square QFN package, respectively. This is a reduced package size of approximately 50% compared with the company's other products.

The µPD720201 host controller with four downstream ports and the µPD720202 host controller with two downstream ports cost **$20** and **$10**, respectively.

Renesas Electronics Corp.
www.renesas.com

ARM9 CAN BUS COMPUTER

The **Matrix-522** is an industrial Linux-ready ARM9 CAN bus computer. Its CAN bus interface supports CANopen protocols for target markets such as industrial automation, transportation, and building automation.

The computer is powered by an Atmel AT91SAM9G20 400-MHz ARM9 CPU. It is equipped with 64-MB SDRAM and 128-MB NAND flash memory. In addition, the Matrix-522 integrates two isolated CAN interfaces, two 10/100-Mbps Ethernet ports, and two RS-232/422/485 serial ports. The computer is equipped with two USB hosts, 21 GPIOs, and one SD socket inside its compact metal box. It also features DIN-rail and wall-mounting capability for flexible onsite installation.

The Matrix-522 is preinstalled with a Linux 2.6.29 OS and a BusyBox utility collection. The OS includes Socket-CAN, the open-source CAN driver and network stack that uses a network modeled on the CAN interface to enable multiple applications to simultaneously access one CAN device. In addition, the CANopen library and CANFestival are available on the Artila CD to ease user programming in the CANopen application. The UBI file system provides improved performance and longer lifetime for NAND flash compared with JFFS2.

Software utilities—including webmin and the GNU tool chain, which includes a C/C++ cross compiler and Glibc—are provided for easy management and development of the Matrix-522.

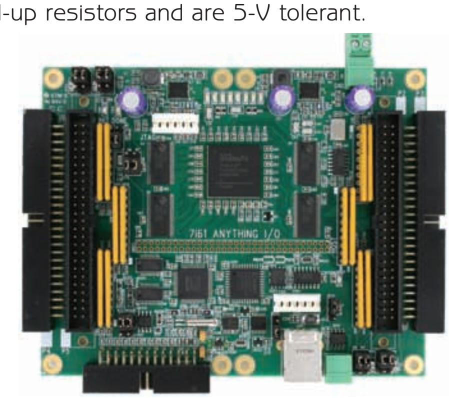
Contact an Artila Electronics distributor for pricing.

Artila Electronics
www.artila.com

USB INTERFACED FPGA I/O CARD

The **7I61** is a USB interfaced version of MESA's FPGA-based Anything I/O card series. It provides 96 I/O bits of easy-to-access programmable user I/O and uses a Spartan6 FPGA for high performance and low cost.

The USB port can be used to download initial configurations either directly to the FPGA or to the on-card flash EEPROM for stand-alone FPGA configuration. Once configured, the high-speed USB port can be used for 7I61 host communication. The 96 I/O bits are available on four 50-pin connectors, at 24 bits per connector. The 50-pin connectors feature I/O module rack-compatible pinouts. All I/O pins have pull-up resistors and are 5-V tolerant.



Several I/O interface daughtercards are available for the 7I61, including four-, six-, and eight-channel analog servo amplifier interfaces, a six-channel resolver interface, and RS-422/485 interfaces. Two daughtercards can plug directly onto the 7I61. A maximum of four daughtercards per 7I61 are supported.

Many I/O configuration files are provided with the 7I61, including simple remote I/O, smart remote I/O, four- to 16-axis servo motion control, four-to-48-axis stepper motor control, multiple channel PWM generator, arbitrary waveform generation, quadrature counters, resolver interface, and multiple timers. VHDL source is provided for all configurations.

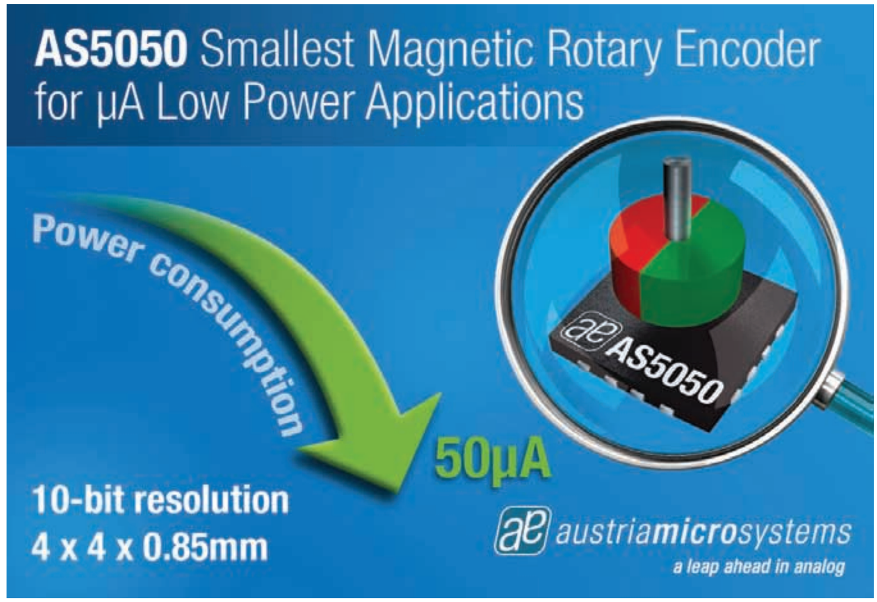
The XC6SLX16 version of the 7I61 costs **$165** for 100-piece quantities.

MESA Electronics
www.mesanet.com

10-BIT MAGNETIC ROTARY ENCODER

The **AS5050** is a 10-bit magnetic rotary encoder IC for contactless position sensing. It is designed for low-power applications, such as robotics, and is also well suited for servo motor control and as an input device for low-cost battery-operated devices.

The single-chip AS5050 integrates four Hall sensor elements, a 10-bit angle encoder, a smart power management controller, and an easy-to-use three- or four-wire SPI. Housed in a small 4 mm × 4 mm QFN-16 package, the low-power encoder is the smallest device in its class. Based on the readout rate, the current consumption is reduced to microamp levels. For example, 50 µA at 10 measurements per second is typical current drain.



To assemble a contactless position measurement encoder system requires only a few passive components and a magnet rotating above the chip. The AS5050 performs all angle calculations on chip and features on-chip automatic wake-up and power-down modes. The encoder only draws current when a new measurement is requested by the host controller and automatically powers down as soon as a measurement is completed.

The AS5050 10-bit rotary encoder costs **$2.84** in 1,000-piece quantities.

Austriamicrosystems
www.austriamicrosystems.com

NPN

ROBUST AND SECURE PORTABLE MEMORY LINE

The **RUGGEDrive** line of portable memory solutions offers the data storage and transfer capabilities of consumer-grade USB flash drives and SD cards in a reliable and secure package.

RUGGEDrive memory tokens are constructed by a solid over-molding process. They feature a rugged composite plastic that protects internal components and enables the devices to be used in harsh environments. Industry-standard protocols including USB 2.0, SDHC, and SPI enable the RUGGEDrive memory tokens to be easily integrated into embedded controllers, single-board computers, and industrial PCs.

The tokens interface with Datakey mating receptacles to provide a base level of security and help overcome the drawbacks of USB flash drive and SD card connectors. The receptacles are rated for 50,000 cycles.

The RUGGEDrive line features two families of memory devices. The UFX memory token provides USB flash drive functionality and the DFX memory token provides SD card functionality. Both tokens come standard with 4 GB of non-volatile memory. Larger capacities, up to 32 GB, are also available. RUGGEDrive receptacles feature multiple mounting options including through-hole, surface-mount, and panel-mount versions. Panel-mount receptacles are available in IP65 ("splash-proof"), IP67 (immersion), and EMI reduction versions to meet the needs of harsh environment applications.

Contact Datakey Electronics for pricing.

Datakey Electronics
www.datakeyelectronics.com

NPN

CIRCUIT CELLAR Test Your EQ

Edited by David Tweed

ANSWERS for Issue 250

Answer 1—This function counts the number of ones in the binary representation of the input argument. The expression n & -n has, at most, one bit set—in the position of the least-significant "one" in n— which is the only bit that a number and its two's-complement negation have in common.

Each iteration of the loop identifies one "one" bit in n, and then the expression n &= ~t clears that bit for the next iteration. The variable k counts how many times this occurs.

Answer 2—On average, this function is faster. It only iterates as many times as there are ones in the argument. Conventional algorithms usually iterate over all of the bits in the argument.

Answer 3—Handheld cameras typically have aspect ratios between 1.2 and 1.5; let's assume this one has a ratio of 1.33 (i.e., equivalent to 640 × 480). That means that the sensor has about 4,470 pixels horizontally and 3,360 pixels vertically.

22° divided by 4,470 is 0.00492°, or 85.9 μradians. Since sin(x) = x for small angles, we can just multiply the height by the angle in radians to get the resolution: 0.859′ or 10.3″.

Answer 4—I'm not aware of any solution using four 4s, but here's one using just three:

$$\frac{4! - \sqrt{4}}{.4} = 55$$

Contributed by David Tweed

What's your EQ?—The answers are posted at www.circuitcellar.com/eq/

You may contact the quizmasters at eq@circuitcellar.com

by George Novacek (Canada)

BITs and BITEs

Tests to Ensure System Safety

Built-in test equipment is an important component in safety-critical systems. Design engineers must always consider what's needed for safe operation and easy maintenance, and then take measures to adequately balance safety and over-testing.

Safety-critical systems can't exist without extensive diagnostics, but every well-designed embedded controller should have them to some degree, too. The watchdog timer (WDT) is the most rudimentary and easily achievable monitor. When coupled with some data typing and continuity checking, even the simplest applications will benefit.

Internal diagnostics are often called built-in tests (BITs). The hardware part of the BIT is built-in-test-equipment (BITE). In general, three types of BITs are used, although not necessarily all of them all of the time. Power-up BIT (P-BIT), also known as "power-up diagnostics," runs, as the name implies, upon the system power-up. It tests the power supply and the loads, exercises I/Os, checks the RAM, and verifies the firmware by computing its cyclic redundancy check (CRC) or the checksum.

Once the P-BIT has been successfully completed, the controller starts the continuous BIT (C-BIT). Its tests are run continuously in the background, verifying the validity of data and the health of the peripheral devices (in the aerospace industry, these are called line replaceable units, or LRUs), wiring continuity and additional tests, as long as they don't interfere with the controller's normal operation. If it detects anything amiss, an error handler is invoked and a predefined sequence of activities follows (typically): switching to a second processing channel, disconnecting loads, logging the event into a nonvolatile memory, attempting to reset, displaying the cause of the problem, and so on.

Initiated BIT (I-BIT) is mainly used for maintenance. Additional tests, usually not permitted during normal operation, are performed while the controller may be connected to a PC to display the results. Diagnostics performed by car repair garages are a good example.

For safety purposes alone, redundancy might suffice. Outputs of the redundant processing channels are compared, and if they're different, a fault is declared. Most systems, however, require that the fault is also isolated to the LRU level for the faulty peripheral to be identified and replaced without lengthy troubleshooting. Let's see how this can be accomplished.

MONITORING

Economical systems, for example, use potentiometers to issue position commands or detect a position, as opposed to precision devices, such

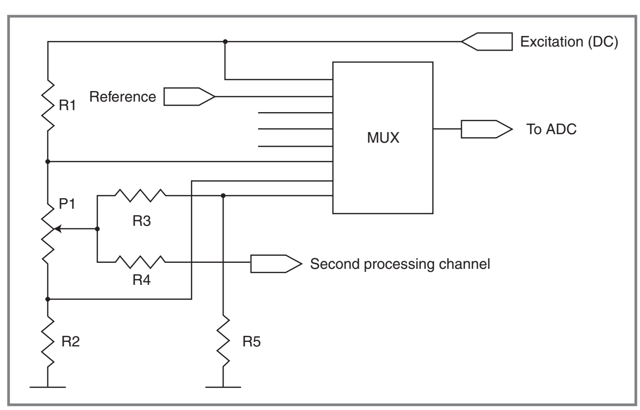


Figure 1—Failure monitoring of a potentiometer

as rotary variable differential transformers (i.e., RVDTs). Comparing the two redundant potentiometers' outputs will quickly indicate a problem, but will not identify the faulty pot. This is rarely necessary, as the entire subassembly is usually replaced, but dual pots with guaranteed tracking accuracy are still costly, thus a single pot is often employed. Monitoring of a single pot in a dual redundant system is shown in Figure 1.

Resistors R1 and R2 ensure that the output of a correctly working P1 is never 0 V or the DC excitation level. If it is, a fault must be present. The excitation level is monitored and if the connection between the pot wiper and the multiplexer (MUX) is broken, the MUX pin is pulled low. Resistors R3 and R4 isolate the two processing channels. A short in one channel affects the voltage in the other channel, but doesn't invalidate it. This, once detected, can be compensated for by the software, and the controller could continue to operate in a reduced authority mode.

If the multiplexer has a spare input, it can be tied to a reference voltage. It then monitors the system references and also corrects operation of the MUX and the ADC by verifying that expected reference values are acquired. To prevent fault propagation between channels, using parts from a multipart package, such as amplifiers from a quad op-amp in both channels, is not permitted. Sometimes a mechanical barrier between the processing channels or channels located in separate enclosures is required to prevent loss of control due to, for instance, a catastrophic event. Years ago an aircraft crashed, having lost control when a triple redundant rudder controller in a single enclosure was destroyed by an explosion.

DC loads are usually driven by totem-pole drivers Q1 and Q2 to ensure that the load can always be de-energized (see Figure 2). During the P-BIT, Q1 and Q2 are alternatively turned on while the voltages at the designated points and the load current through R1 are monitored. This establishes that both drivers work properly, power for the load is as required, and the load draws its rated current.

Overcurrent, including a short, demands immediate disconnect. Because software cannot guarantee fast enough disconnect, short circuit protection by hardware, such as Q3 and Q4, must be implemented. Monitoring electromechanical loads during P-BIT, typically solenoid valves, is challenging. Due to the solenoid's inductance, the voltage across R1 builds up slowly, but the solenoid is rarely permitted to be energized longer than about 30 ms to prevent opening of the valve and subsequent actuator movement. Other loads, such as incandescent lights, present an opposite challenge. Their in-rush current appears to the BIT as a momentary overcurrent. Clever designs solve such challenges, but, as you can imagine from these two simplified examples, the BIT can quickly snowball and cause the design complexity of a controller to grow beyond imagination.

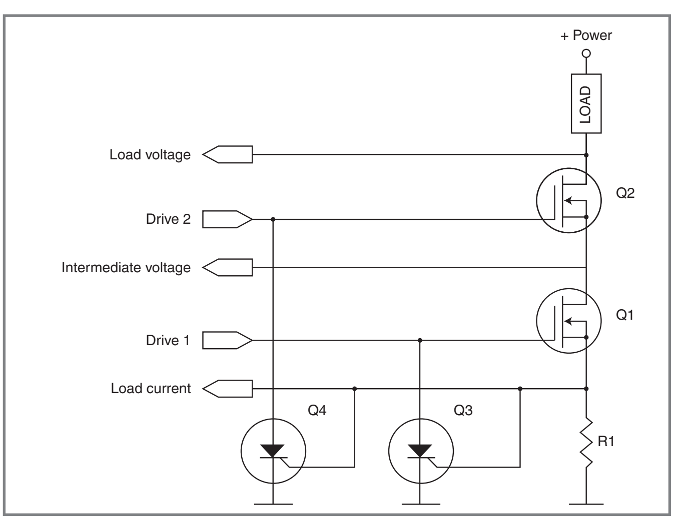


Figure 2—The output driver monitor

KEEP IT SIMPLE

It is not uncommon to see the BIT use more hardware and more of the microcontroller's resources than the fundamental functionality of the system. Design engineers need to retain common sense and not go overboard with fault-detection capability, as it is easy to create problems instead of solutions. The more encompassing and deeper the BIT gets, the more opportunity there is for false alarms due to timing, tolerances, interference, and so on. Keep in mind what the system needs for safe operation and easy maintenance. Once satisfied, stop! You want to detect whenever something goes wrong and identify the culprit in terms of an LRU, including the controller itself. The user rarely cares what the internal fault is when the entire LRU must be replaced.

I once saw a BIT designed to detect just about every internal and external fault of a system. It had professed fault isolation down to functional blocks and even some individual components because the engineer who designed it believed it could be done. It was a disaster. The problem is that faults are rarely solitary. Often a fault is followed by an avalanche of secondary faults. Since the processor executes the BIT firmware one instruction at a time, without a major increase in hardware complexity, it is impossible to tell whether a secondary effect or the root cause is the first fault captured.

I once worked on a controller driving a 200-V, three-phase load with the BIT designed to detect and identify every imagined fault. A momentary glitch on one power phase caused the BIT to declare a fault, but the isolation of the cause was unpredictable. It could have been identified as a low phase voltage, a low load current, a change in the load resistance, or a driver problem, all depending on which test the BIT was performing the instant when the glitch happened.

The bottom line is that to design an effective BIT and BITE, the engineer must have a good understanding of the system, creativity, imagination, and, above all, common sense to not get carried away by perceived technological capabilities. Keep it simple!

George Novacek (gnovacek@nexicom.net) is a professional engineer with a degree in Cybernetics and Closed-Loop Control. Now retired, he was most recently president of a multinational manufacturer of embedded control systems for aerospace applications. George wrote 26 feature articles for Circuit Cellar *between 1999 and 2004.*

by Richard Wotiz (USA)

Construct a Multifunctional Network Controller

The innovative Net Butler is a multifunctional design used to control, monitor, and automatically maintain a home network. Built around an iMCU7100EVB, the design has several functions, such as reporting on connected network devices and downloading Internet-based content.

About three years ago, my local phone company announced that they would be installing a fiber optic network in my neighborhood. Once completed, they would be able to offer high-speed Internet service for the first time. I couldn't wait for that day, since I had been surviving with a slow-but-reliable dial-up connection for as long as I could remember. The service came with a residential gateway (RG), which combines a DSL modem, hardware firewall, and wireless router in one package. I had never set up a home network before, and this seemed like the perfect time to start. I began by connecting my laptop and desktop systems to two of the RG's four Ethernet ports. Later on, I tried out the laptop's Wi-Fi port, and I was able to surf the Internet from anywhere in the house. The next step was to get my old boat-anchor laser printer on the network. I found an older HP JetDirect print server with a parallel output to feed the printer. After some gentle coaxing, I was able to print from both machines. Success!

The printer uses a lot of power even when it's idle, so I only turn it on when I'm using it. It takes a few minutes to warm up. If I'm not in the same room when I want to print something, I have to go turn it on, wait a while, return to the computer, print a document, then go back to the printer when it's done. Sometimes I'll get distracted and forget to turn it off. I thought it would be much more convenient if I could remotely control the printer's power.

I recently purchased a color inkjet printer with a USB port and I needed a way to connect it to the network as well. In addition, the printer has an annoying issue that plagues other inkjet printers, where the ink can dry up and clog the print nozzles if it isn't turned on at least once every few weeks. I don't use it very much, so I wanted some way to automatically turn it on for a few minutes every so often.

The RG has the habit of checking up on every device it's ever seen on the local network by broadcasting a request every 20 s. If there's no response, it waits 1 s before moving on to the next device. Since most of the devices I use aren't powered on much of the time, this means that the network activity light on the computer I'm using will blink once a second almost all the time. This makes it difficult to know if there's any real network activity, especially when waiting for a slow-loading webpage. It's particularly annoying on my laptop, since its network LED blinks at a fixed 2-Hz rate whether it's receiving a large file or just a series of ping requests. I wished there was some way to eliminate this constant blinking, or at least make it less frequent.

I created the "Net Butler" as a multifunctional network appliance to handle these tasks, and more (see Photo 1 and Figure 1). Like a real butler, it's adept at performing the same task over and over without needing to be reminded, and it's easily upgradeable to handle new tasks as they come along. As I was designing, I kept coming up with new features I wanted to include. Table 1 shows the design's current functionality. The system also includes a

Photo 1—Take a look at the finished Net Butler design. You can see the various cables leading to the network and the printers.

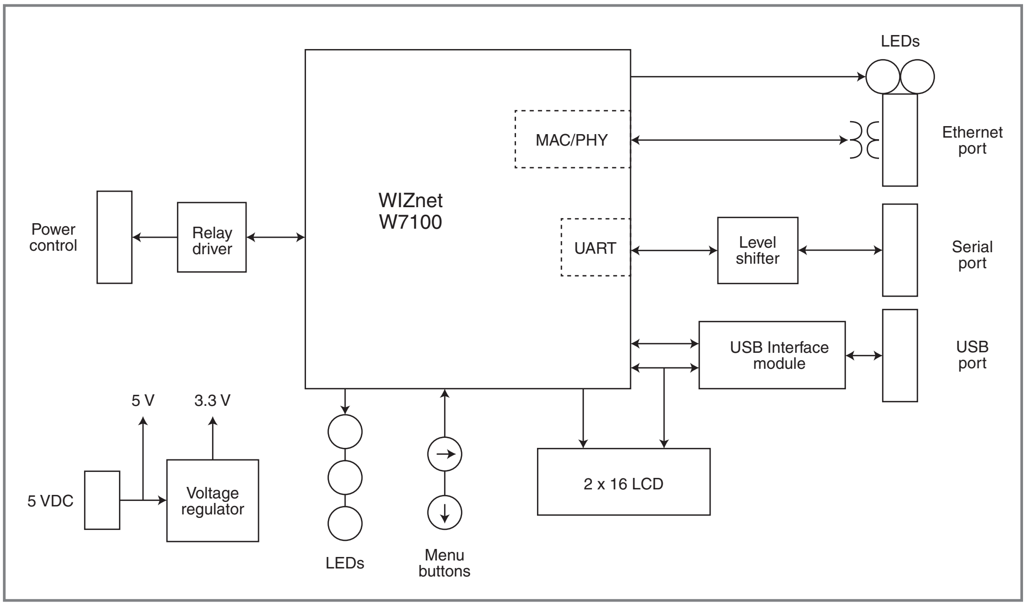


Figure 1—The system is built around the WIZnet W7100 Internet microcontroller. It's surrounded by I/O ports for the network and printers.

is connected, since the output will float high with no load present. I mounted the relay inside a plug-in surge suppressor module, rather than worrying about having line voltage inside the Net Butler's enclosure.

bootloader to download new code over the network, making it easy for me to add new functions as needed.

BUILDING THE BUTLER

I started out with a WIZnet iMCU7100EVB evaluation board, which contains a W7100 Internet MCU and support circuitry. It also has Ethernet and serial ports, and an LCD. I made a few modifications to the board, then added a USB interface module, a pair of push buttons, and a relay driver. A 5-V regulated wall adapter powers the system (see Figure 2). I installed the board in a small plastic box, which you can see in Photo 2 with its cover removed. I used a Future Technology Devices International Vinculum VDIP1 USB module for the USB print server port. It's a completely self-contained USB interface that communicates with the W7100 using an 8-bit parallel bus.

I wanted to remotely control the USB inkjet printer's power, but its instructions advise against shutting off the power externally (e.g., with a power strip). That's so it can properly park the print head on power-down to keep it from drying out. I found that the printer's on/off switch simply outputs a logic-level signal that controls its power supply. I was able to easily modify the printer so that the USB bus power feeds an optoisolator with its output connected across the power switch. Then I made a minor modification to the VDIP1 module to enable the W7100 to control the USB bus power. The Net Butler can then control the printer's power through the modified VDIP1 module. (Refer to the *Circuit Cellar* FTP site for a schematic of the modified module.)

The laser printer doesn't have any special power-sequencing requirements. I added a transistor to the evaluation board, which is fed by an output port. The evaluation board controls an external relay that switches power to the printer and its print server. An input port connected to the transistor's output enables me to detect whether the relay

WRITING THE CODE

I wrote the code in C using IAR Embedded Workbench for 8051 version 7.51A. The code uses almost all of the 64 KB of flash memory, all of the 64 KB of RAM, and all eight of the network sockets along with all of the socket memory. I set the compiler options to store constant data in flash memory rather than RAM. This saved more than 15 KB of RAM space. But this broke the C library's string handling routines, since they could only access data in RAM. I ended up writing my own I/O and string functions for flash-memory data, as well as some RAM-based ones using the W7100's dual DPTR registers for faster operation.

I decided to build debugging into the code rather than use an external debugger. I included `PRINTF` functions in every task and user-settable flags to control the level of detail shown for each task. The messages are sent to the serial port, and optionally through a TELNET connection for debugging over the network. This made it easy to track down bugs, and will help when adding new features later on. I left the debugging code in the final version of the software, so the messages can be enabled at any time. This will help me to analyze unexpected changes in system behavior, such as when external devices are changed or upgraded. I also found a network protocol analyzer to be invaluable. That's because I had to write most of the protocol code from scratch, and documentation on the W7100's hardware TCP/IP core wasn't as complete as I would have liked. I used Microsoft's Network Monitor 3.2, but any similar packet sniffer would have worked as well.

Some of the Net Butler's features can be accessed

The Net Butler's Features
Track currently and previously connected network devices
Remote power control for two printers
Remote enable/disable of Wi-Fi network
USB print server
DNS proxy with domain name block list and activity log display
Daily and weekly weather forecast display
Web server for viewing system activity and configuration settings
LCD menus for local system control and monitoring

Table 1—The Net Butler's main features. This list will evolve over time.

through the LCD's menus. Two push buttons enable you to step through the menus, and pressing both at the same time performs a menu-specific action. The main menu shows the status of the overall system. It displays "Ready" when the network is connected and an IP address is assigned. There's a flashing dot that acts as a

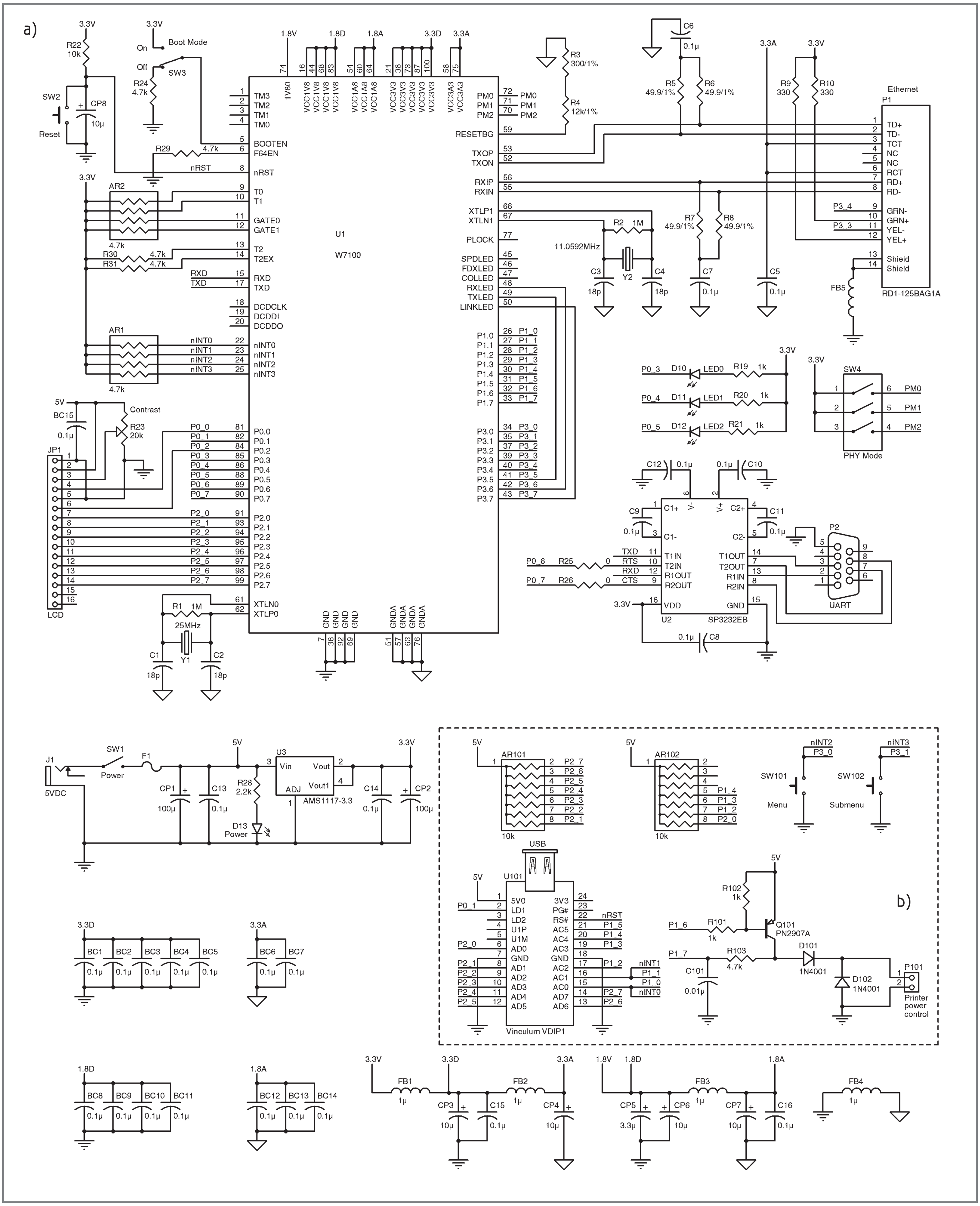


Figure 2a—The evaluation board, which I left largely intact. b—I made a few minor changes and added this circuitry.

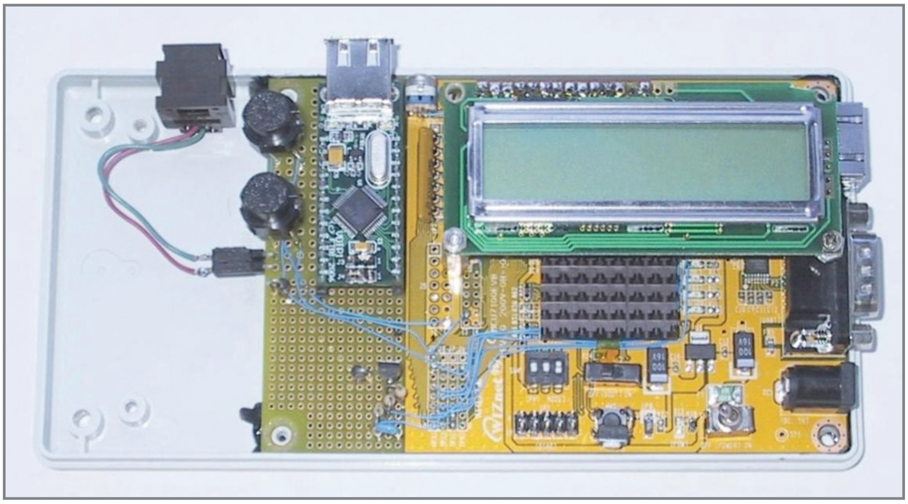


Photo 2—The modified evaluation board. The USB module is to the left of the LCD. The printer power relay plugs into the connector at the far left. I was concerned about the W7100's increasing temperature so I stuck a small heatsink on top. It's partially hidden by the LCD.

heartbeat indicator. The second row of the LCD shows a series of eight two-character activity indicators, one for each socket. They illuminate with a code, such as "Dh" for DHCP or "HC" for HTTP client, each time that particular protocol has some activity. Other menus show whether the Wi-Fi network or printers are turned on, indicate the current IP address settings, or display the weather forecast.

The Net Butler performs all of its functions simultaneously, using all of the W7100's network sockets. Table 2 shows the major features along with their file names and socket numbers. I designed most of the tasks with a separate application and protocol layer. The application takes care of the higher-level processing, and the protocol layer handles the details of packet transmission and reception. I used separate state machines to handle each layer. They're repeatedly called by the main routine to handle all of the processing for each task. (Refer to the file Flowcharts.pdf on the *Circuit Cellar* FTP site for a complete set of flowcharts and state diagrams. The file Details.pdf includes more information on the processing algorithms.)

PRINTER PROGRAMMING

The print server uses a raw data protocol on TCP port 9100, which passes the TCP data directly to the USB port unchanged. It's the same as on the HP JetDirect print server. The VDIP1 USB module takes care of all the USB-to-printer protocols. It provides a command set that enables USB port and printer status checking, as well as

Feature	Protocol	Socket	Application filename	Protocol filename
ARP automated reply server	ARP	0	app_arp.c	p_arp.c
ARP polling (used by DHCP client)	ARP	0		p_arp.c
Destination unreachable processing	ICMP	1		p_icmp.c
Debug port (if ICMP is disabled)	TELNET	1		serial.c
DHCP client	DHCP	2		p_dhcp.c
DNS proxy server and client	DNS	3 & 4	app_dns.c	
Web client	HTTP	5	app_httpc.c	p_httpc.c
DNS client (used by Web client)	DNS	5		p_dns.c
Web server	HTTP	6	app_https/*.c	p_https.c
Print server	RAW	7		p_print.c
Printer power control	–	–	app_print.c	

Table 2—The Net Butler supports all of these protocols using the specified socket numbers. Some sockets are shared by two different features, but they'll never both be active at the same time.

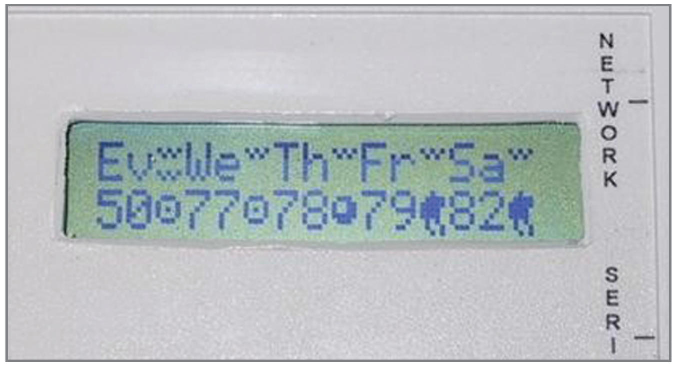


Photo 3—The Net Butler is displaying the current week's forecast. As you can see, the week will start sunny with moderate winds. Thunderstorms and light winds will begin by the weekend. (Data source: The Weather Channel, www.weather.com.)

other USB-specific functions. But I decided to keep it simple and treat it as a straight data pipe to the printer.

The server uses the state machine in Figure 3. It waits for an incoming connection, then writes any data it receives into a buffer. The VDIP1 module has an internal FIFO that accepts data in 64-byte blocks from the buffer. The FIFO has a significant delay between blocks, which slows down the data transfer. I've found that the printer can take more than twice as long to print a page as compared to plugging it directly into the laptop's USB port, but the extra convenience is worth the wait.

There is also a reverse data channel from the printer to the host that I'm currently not using. I originally put in the code just in case I buy a USB scanner someday. But I found I needed to read in any status messages sent by the VDIP1 for it to work correctly, even though I'm ignoring them for now.

The printer power control feature enables me to turn either printer on and off. Each one also has a timer to turn the power off after a configurable delay. Additionally, there's a timer to turn the USB inkjet printer's power on every few days so it can perform a head-cleaning cycle.

NETWORK DEVICES

The ARP reply server keeps track of which devices are currently active on the network and it sends out its own replies for any that don't respond. That prevents the RG from doing a string of 1-s timeouts while it waits for each unresponsive device during its 20-s polls. That means my laptop's network activity light will flicker once every 20 s, which I found to be less annoying than once a second.

The server keeps a table of all ARP requests that the RG sends out. It keeps track of which ones were repeated within the same 20-s burst, indicating that the requested device didn't reply the first time. If this happens enough times in a row, then the server assumes the device is not connected and sends out a reply in its place after a short delay. The delay is just long enough for the server to detect whether the device has been reconnected and has replied on its own. The server needs to know both the IP and MAC address for a device in order to send a reply. These can be specified in a configuration table, which holds data for all of the devices that are permanent members of my home network. It can also capture the address of a new device whenever it sees an ARP request from the device.

DOMAIN NAMES

One of the Net Butler's more useful features is the DNS proxy. It acts as an intermediary for all DNS requests coming from inside the network. Its main features are a domain name block list and an activity log. I can specify a list of site names that will be blocked when I try to access them. I've used this to enhance my privacy by blocking several advertising and tracking sites. The blocking function can take one of several actions when it's triggered. It can return either a DNS request error, the local loopback address (127.0.0.1), a webpage with an HTTP error, or a blank webpage. A blocked site name will match any subdomains of the site. For example, specifying "blocked.com" will also match “also.blocked.com,” but not “notblocked.com.”

The DNS activity log keeps track of all lookups for the past 30 days. It logs the domain name, time of first access, time of most recent access, and total number of accesses. The entries can be sorted by any of these fields. The log also tracks whether the site was blocked, and whether there was a delay of more than a few seconds since the immediately preceding DNS lookup. This last feature provides some rather intriguing data about website behavior. It can separate the log entries into groups, with each group containing the website the user selected as well as all sites accessed by the webpage itself. I’ve seen some webpages access dozens of sites at once, including some tracking sites that didn’t take me long to add to the block list. Log entries that have been accessed in the past few minutes can be highlighted in red to help them stand out.

ON THE WEB

The Net Butler gets some of its information from various webpages. The web client application requests a page and sifts through the results for specific fields, using the HTTP client protocol engine, which handles the details of the transaction. The application currently has two functions. It monitors and controls the wireless router by exchanging data with the RG’s built-in web server, and it displays the weather by downloading data from a weather website. It’s designed so new functions can be easily added.

I prefer to keep the wireless network disabled when I won’t be using it for a while. But I wanted to be able to enable it without having to carry my laptop to the hard-wired Ethernet cable in my home office. I can enable and disable the Net Butler with the LCD menus. Pressing a button sends a request to the RG to change the network status. The Butler also sends a status request to the RG every 10 minutes to keep the menu display updated. I can also set an option that will automatically disable the Wi-Fi

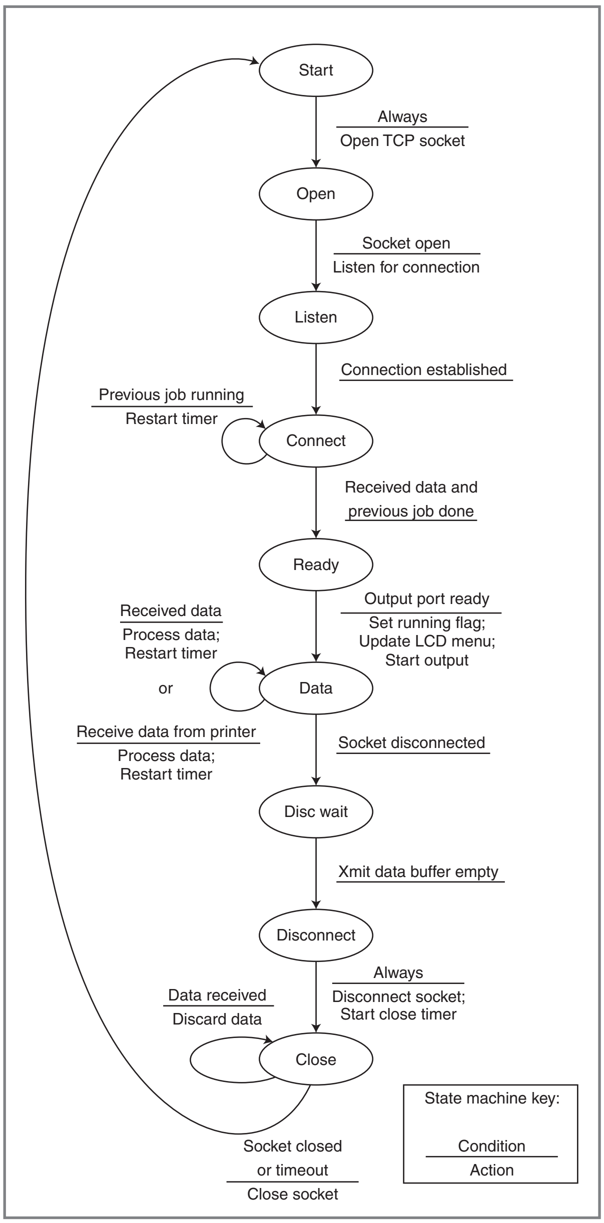


Figure 3—This state diagram shows the print server’s basic operation. It’s normally in the Listen state until it sees an incoming connection. It then proceeds to the Data state to process a print job. State transitions for various error conditions aren’t shown.

network after a specified elapsed time if there are no active devices.

The weather application periodically sends forecast requests for the current day and for the week ahead. It extracts the temperature, sky condition, probability of precipitation, and wind speed and stores them in a pair of arrays. I made a set of bitmap icons for the various sky conditions and wind speeds that are loaded into the LCD's character-generator memory. Up to five hours or days can be displayed at once (see Photo 3).

The Net Butler's web server provides access to all of its features and configuration settings in one place. Like the web client, it consists of two parts. There's an application that generates dynamic webpages and a protocol engine that manages the connection to the client. The protocol state machine has a basic structure that's similar to that of the print server in Figure 3.

Pages are generated from the raw data each time they are requested. Each page consists of dynamic data combined with string constants. Rather than build a complete page in a large buffer, I wrote the low-level routine `sendbuf()`, based on WIZnet's original TCP `send()` function, which buffers data directly in the TCP/IP core's transmit buffer. I also wrote a set of macros and functions (see app_https.h) to output various types of flash- and RAM-based strings. These are used to simplify the page-generating routines. (Refer to Screens.pdf on the *Circuit Cellar* FTP site for screenshots of the various webpages.)

HOUSEKEEPING

The W7100's TCP/IP core has its own ICMP processing function, which handles destination unreachable messages in response to misdirected UDP packets. But the host and port numbers it reports are incorrect, according to a WIZnet errata document. I wanted to have accurate data to avoid confusing some of the other protocol routines, so I wrote my own ICMP code to get the correct values. But opening an ICMP socket disables all of the core's automatic ICMP processing, including echo (ping) requests, so I had to include that in the code as well.

The Net Butler can be configured in either DHCP or static IP mode. The DHCP client attempts to locate a DHCP server on the network and request an IP address. If it fails, it can either fall back to static IP mode, or retry forever until it succeeds. The Link LED will flash before configuration has been completed and be on solid afterwards. I originally wrote the protocol to match the specifications in RFC 2131. But I had to make some minor tweaks after trying it with a couple of servers, one of which didn't follow all of the details of the specification.

BOOTLOADER

The bootloader is a separate software project that's downloaded to the W7100 using WIZnet's WizISP program. I wrote a simple TFTP protocol handler to download code from the network and write it to flash memory. The bootloader uses only 5 K of flash space. I used the TFTP client supplied with Microsoft Windows XP to send binary images to the bootloader. That made debugging easy, since downloading new code took only a few seconds and could be automated with a batch file.

The bootloader always starts up after a system reset. It first checks if the Menu button is pressed or if the first byte of the application's reset vector is invalid. In either case, the main part of the bootloader is started. Otherwise, it immediately jumps to the application. The bootloader downloads the entire code into RAM and does several integrity checks on it before writing it to flash.

Part of the bootloader resides in the flash memory's first 1-KB sector, which includes the interrupt vectors. That sector is never erased, but includes a set of jumps to the application's vectors, which are automatically relocated from their original location at 0x0 to the beginning of the second flash sector starting at 0x400. The application can't use the area between the end of its vector table and address 0x47F in order to give the bootloader enough space to do the relocation. Since the vectors are at their usual locations until the bootloader relocates them, the application will also run normally if the bootloader isn't installed.

SERVING UP RESULTS

I have been using the Net Butler for several months now, and I use every one of its features regularly. It has eliminated all of the annoyances I originally had when dealing with my home network. Someday I'd like to have it check for new e-mail too, but first I would need to implement SSL to talk to my e-mail provider. That's a project for another time. ■

Author's note: I would like to thank The Weather Channel (www.weather.com) for granting me permission to use some of its data for this article.

Richard Wotiz earned a B.S.E. in Electrical Engineering and Computer Science from Princeton University. He has run his own hardware consulting business for the past 20 years, specializing in consumer products and children's toys. You can reach him at dick601@mystics.org.

PROJECT FILES

To download the code, screenshots, and webpages, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2011/251.

SOURCES

Vinculum VDIP1 USB Module
Future Technology Devices International, Ltd. | ww.ftdichip.com

HP JetDirect print server
Hewlett-Packard | www.hp.com

IAR Embedded Workbench
IAR Systems AB | www.iar.com

Microsoft Network Monitor
Microsoft, Inc. | www.microsoft.com

iMCU7100EVB and W7100 Internet microcontroller
WIZnet | www.wiznet.co.kr

by Jigar Shah and Kevin Yang (USA)

Wireless Authentication

RFID Checkout System Design

You can design an RFID system to serve as a viable alternative to the decades-old UPC barcode. This article details the topics of RFID tag structure, antenna tuning, and more.

A trip to the store usually involves waiting in line to get your items scanned for checkout. This results in lost productivity for both consumers and retailers. Furthermore, the barcode standard dates back to the 1970s, before the age of affordable computing power. Today, computers power items ranging from airbags to headsets. Thus, we decided to see if we could envision the next revolution in retail technology for our senior design final project at Cornell University.

With the increasing prevalence and affordability of radio frequency identification (RFID) tags in everyday authentication systems, the technology holds great promise in the retail world for both customers and retailers in inventory control, convenience, and cost savings. Our design uses RFID tags to automate the checkout process by reading RFID signals on objects placed in proximity to an antenna platform. This eliminates the need for barcode scanning each individual item, making checkout a significantly faster experience. Furthermore, unlike barcodes, each item—even multiples of the same item—has a unique tag. This means much better inventory control, recall ability, and consumer behavior monitoring, which makes for a improved and safer overall customer experience and creates higher margins for manufacturers and retailers.

Our prototype system is organized around several central components (see Photo 1). The prototype board containing an Atmel ATmega644 microcontroller receives user input from three push buttons and provides user feedback via an LCD. This communicates with a SkyeTek SkyeModule M1 HF RFID reader, which transmits power to any RFID tags in the vicinity and receives the reflected signal back via the antenna. Multiple tags within the RFID field won't result in a problem because each tag is told to "stay quiet" after it is read. This is due to the anti-collision support of the reader we use.

HARDWARE COMPONENTS

When choosing an RFID reader for our project, we focused on minimizing cost while ensuring compliance with industry standards and requiring simultaneous,

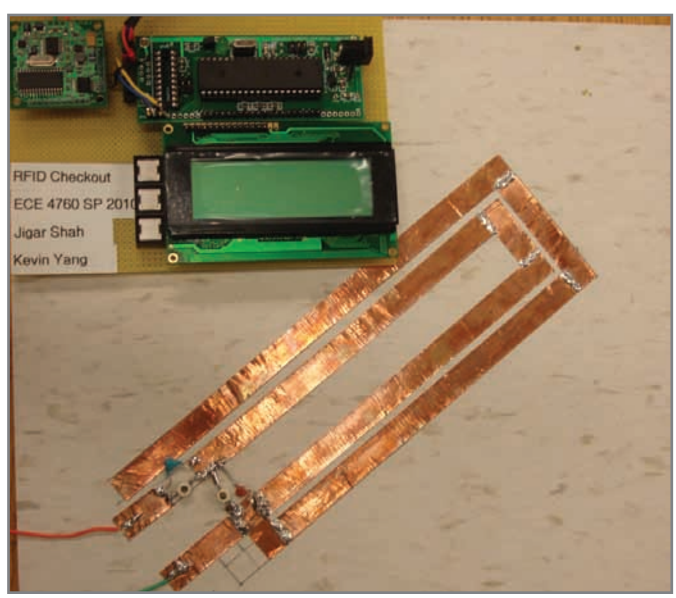


Photo 1—The finished RFID checkout system includes a custom-built antenna platform, a user interface, an Atmel ATmega644 microcontroller, and a SkyTek SkyeModule M1 HF RFID reader module.

multiple-read capability. A SkyeModule M1 HF RFID reader powers our prototype system (see Photo 2). This reader is capable of reading most of the industry-standard 13.56-MHz RFID tags. It supports SPI, TTL, I²C, and RS-232 communication with computers, microcontrollers, and other devices. It also transmits the least-significant bit first in its communication. In reading tags, the reader has anti-collision capabilities, which means it can read multiple RFID tags in the vicinity of its antenna. This makes it an ideal choice for our RFID checkout system. In addition, the SkyeModule M1 HF RFID reader uses its own proprietary SkyeTek communication protocol, which we will discuss later in this article.

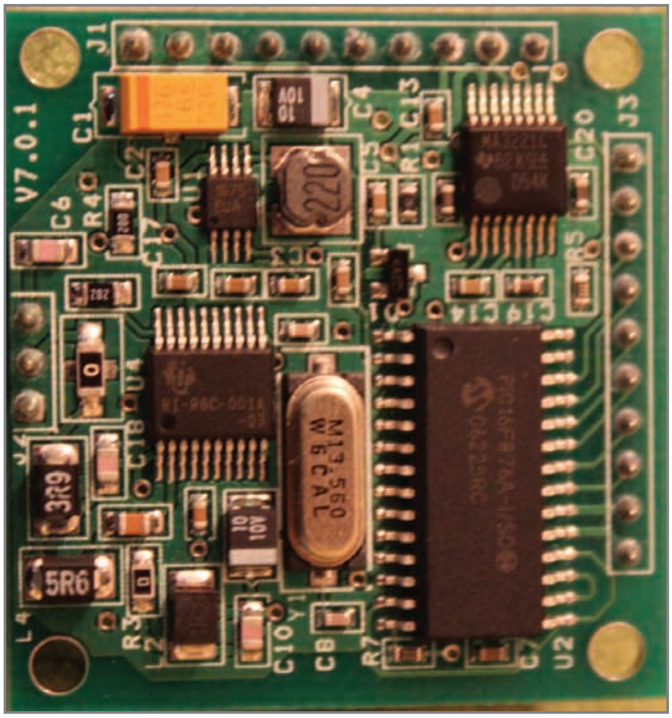


Photo 2—The SkyeTek SkyeModule M1 HF RFID reader is at the heart of our project, reflecting power on the passive RFID tags and interpreting the reflected signal. A 64-bit interpreted serial code is then transmitted to the microcontroller for our look-up table.

Our prototype system requires a wide read area to accommodate the scanning of an entire shopping basket or cart. Compared with RFID's most common application for security, this requires deviation from the typical coil antenna. We use copper foil tape, which has much lower overall impedance in comparison with a wire of similar area and can be readily scaled to the required size.

We designed the antenna with 1-cm wide copper foil tape with conductive adhesive (see Figure 1). Much of the design was based on the SkyeTek antenna design tutorial. Our design did not require an external power amplifier as the length was relatively short, specifically less than 10″, as suggested by the guide. Our design required a fair amount of tuning, switching out capacitors until resonance at 13.56 MHz was achieved, with the help of our professor. This antenna was able to cover a much larger surface area over the stock antenna, with a reasonable range of approximately 3″ above the surface area. In an industrial application, a power amplifier would be necessary to ensure proper performance to read over a larger area.

Our reader supports RFID tags so multiple tags can be simultaneously read. In summary, each tag type has a 64-bit factory-programmed permanent ID. These are the 64 bits we used for our look-up table. Each of these tags also has a user-programmable data field to store a variety of information (e.g., inventory data, expiration, etc.). However, we chose not to implement this data field for our prototype. Our reader supports Texas Instruments's Tag-It HF series, ICode, ISO14443, and ISO15693 standards, in addition to others. Refer to Photo 3 for an example of the backing of an RFID product sticker.

CONTROL CIRCUITRY

An ATmega644 microcontroller is at the heart of the design's circuitry. It's ideal due to its sufficient memory capacity and TTL capability. Pins D0 and D1 are responsible for sending receive and transmit signals to

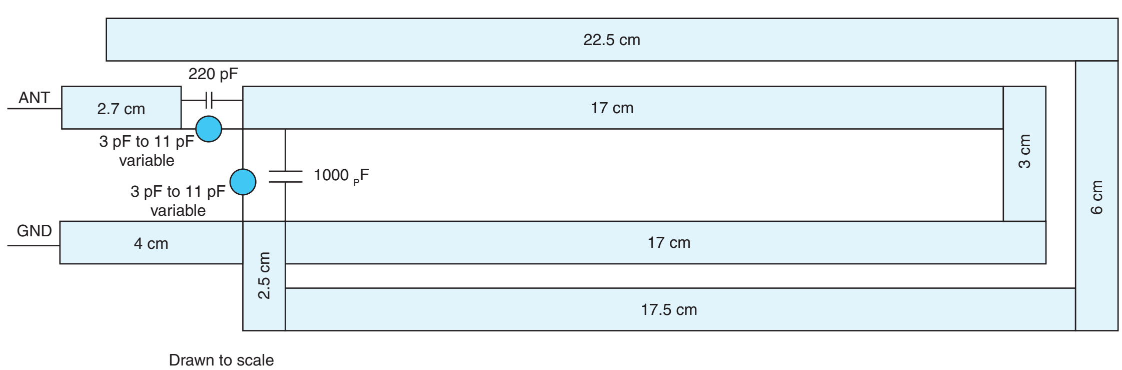


Figure 1—The custom-built antenna platform. We used a floor tile to stay within budget. Copper foil tape combined with specified resistors and variable capacitors enabled the necessary tuning to 13.56 MHz.

the SkyeTek M1 RFID module using UART. Pins A0, A1, and A2 receive inputs from the three push buttons in our circuit. Port C controls the LCD.

We built the protoboard with a board designed specifically for our class at Cornell. The board has a slot for a Maxim MAX233CPP driver/receiver for an RS-232 connection, which we don't need. Furthermore, the board features a programmable header, a power switch, an indicative LED status light, various jumpers, and a diode to protect the circuit, in addition to resistors and capacitors (see Figure 2).

Our program for the RFID checkout system comprises three main sections. In the beginning of our program, we declare multiple arrays that corresponded to known RFID tags and their associated information. In our main method, we implement a state machine that defines an intuitive user interface for a checkout lane. Finally, we also defined many helper methods for searching for particular RFIDs or recognizing certain RFIDs as a particular product.

A list of the RFID tag arrays and its associated information is declared at the beginning of the program. The associated information is in the same order as the RFID tag codes. This will enable later find functions to determine whether a particular RFID tag is valid and what information is associated with it (e.g., price, product name, etc.). This proved sufficient for a limited demonstration for our system. In a larger implementation, it would be ideal if the system could communicate with a larger and easily modifiable database. Therefore, future work would include an easier methodology to input the RFID information, perhaps by utilizing interaction between this system and a computer.

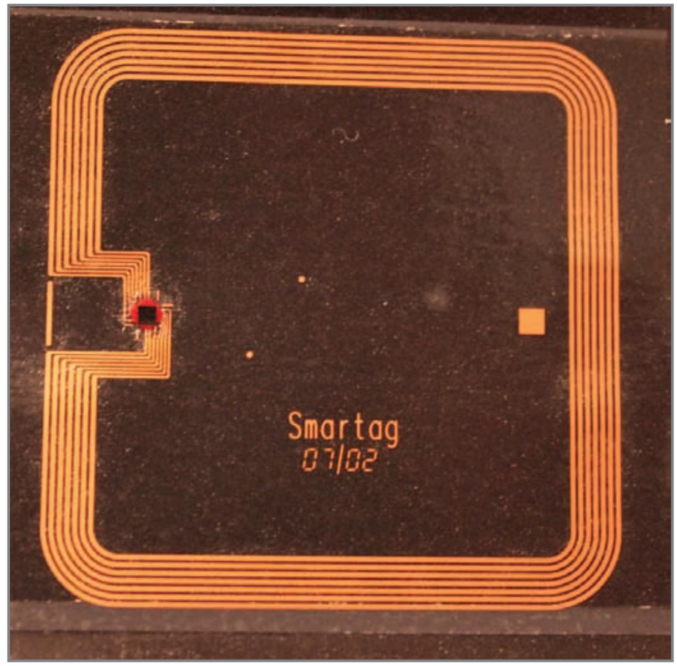


Photo 3—An example of the circuitry embedded in every unique passive RFID tag. The other side of the label you see is simply a white sticker enabling seamless integration of the technology once viable.

match our desired operation. For our particular application, we set options to enable an inventory of all of the tags within the read area, a "stay quiet" command for tags after identification, and a loop command to set a continuous read for RFID tags that had not yet been scanned. From the protocol builder, we surmised that the chip would send a response to this command indicating whether or not the reader was on. Thus, in our program,

SKYETEK PROTOCOL

The SkyeModule M1 HF RFID chip in this project uses the SkyeTek communication protocol when communicating with computers, microcontrollers, and other devices. This SkyeTek protocol describes how to use a microcontroller to configure a module to read RFID tags.

In order to interface the M1 module with the microcontroller, we had to learn the methodology used to send commands to the chip. Furthermore, we knew the microcontroller had to have a limited ability to interpret commands sent in response from the SkyeTek module, whether the command would be a signal indicating that the module is ready to read tags or a signal indicating a read RFID tag.

When experimenting with the SkyeTek protocol, we used the SkyeTek Protocol Builder software program included with the RFID kit. From this, we developed the proper commands to send to the M1 module to

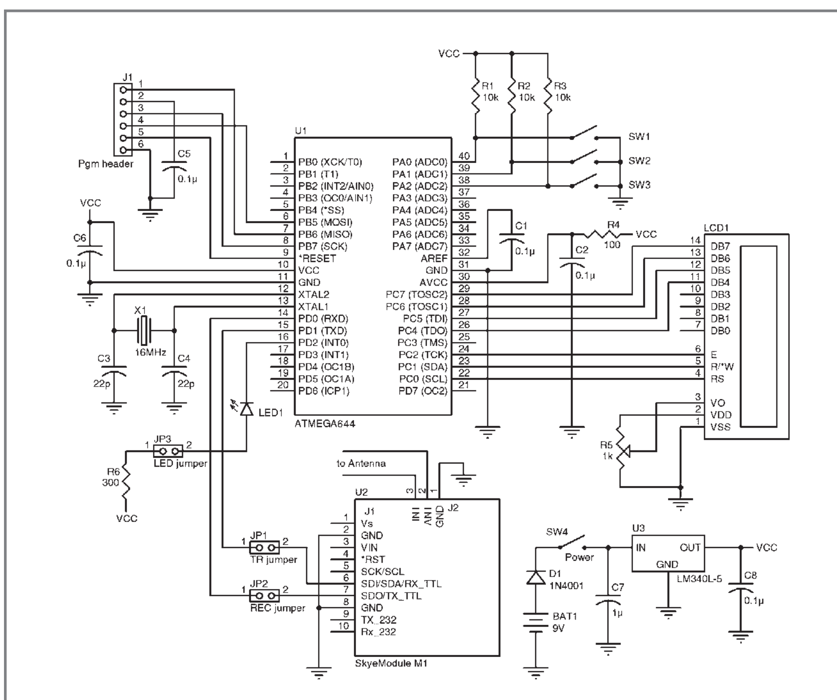


Figure 2—The ATmega644-based RFID checkout system's circuitry

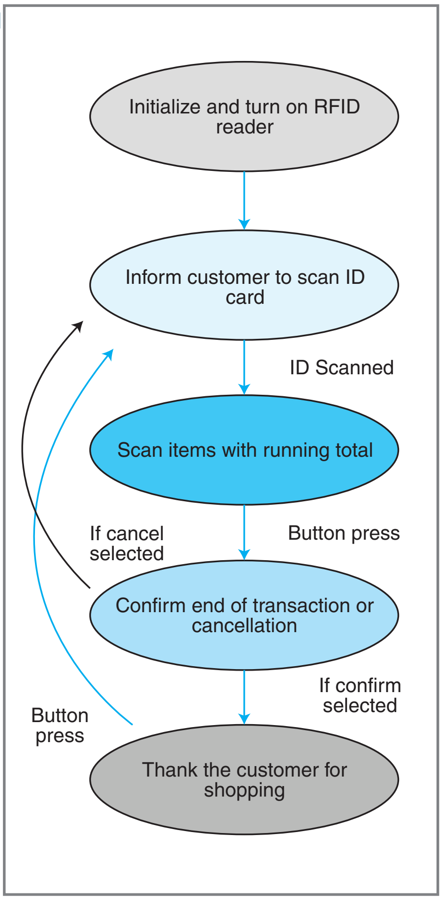


Figure 3—Our checkout system has a step-by-step, user-guided interface. After initialization, a shopper identification card is scanned. Next, items are scanned in parallel within the read area until the shopper ends the session.

we expected a message indicating that the reader was not actively reading or a message that the reader was actively reading, as required.

To send the commands to the chip, we used the UART interface of our microcontroller. This enabled us to easily send and receive serial data to and from the chip. Therefore, we connected the D0 and D1 pins on the ATmega644 microcontroller to the TX_TTL and RX_TTL pins on the SkyeModule RFID chip, respectively. We decided to use a TTL connection as opposed to a SPI or a RS-232 connection because of its relative simplicity given the time constraints of our project.

STATE MACHINE

We designed our state machine to be straightforward and simple, allowing for the demonstration of our prototype without much overhead or frills (see Figure 3). Upon initialization, the reader-on signal is transmitted to the SkyeTek M1 module until it is confirmed that the reader is operational and ready for the system. You can then scan your customer identification/shopper loyalty card. Once you are identified, all items placed within the read range are scanned. As each item is scanned, its RFID tags are sent to the "stay quiet" command as described earlier to ensure that all other tags within the field are properly read.

When you are done placing your items on the antenna platform and all items are recorded, press the Confirm button. This brings up a confirmation dialog that asks whether you want to proceed to payment or cancel the transaction. If you cancel the transaction, the system reinitializes to ask the next consumer to present a customer identification card. If the transaction is confirmed, a "thank you" message is displayed (as we have not implemented a payment system) followed by a reinitialization once the Confirm button is pressed again. This creates an effective and intuitive user interface for the presentation of our multiple-read RFID checkout system.

BUTTON IMPLEMENTATION

The design has three buttons: Up, Down, and OK. The Up and Down buttons are used to select different options in the user interface. The OK button confirms options and advances the checkout stages.

A form of debouncing is needed to guarantee that you can not unintentionally scroll through the different stages of the state machine, which could result in unwanted purchases. The only button that requires this, however, is the OK button because it is the only button that confirms information and advances the screens. Therefore, only the OK button can be recognized once it is released. If you were to hold the OK button down, the checkout stages would not advance until the button was released.

TROUBLESHOOTING & TRADEOFFS

Faced with a $75 budget, we were hard-pressed to find a reader chip that included the anti-collision feature we needed to implement our system as we had envisioned. After failing to find anything less than $80 in the United States, we stumbled upon a chip from the United Kingdom that was $30 operating at the 125-kHz frequency. We started our project using this chip, only to learn that the range was rather poor, the tags were expensive due to the outdated frequency, and it was not mainstream enough for a realistic prototype of our system. Luckily, we bid upon a SkyeTek 13.56-MHz RFID kit sold as-is on eBay and succeeded with a closing price of $16.49. This led us to change our entire platform almost halfway through the project period, wasting time that otherwise could have been spent adding more features, if not for the budget issue.

When switching to the new SkyeTek M1 module, we could not get the chip to work successfully with our microcontroller, although it worked when interfaced with the computer. We followed the guidelines as described, and went through the datasheet regarding the ATmega644's TTL connections. On a whim, we removed setting the baud rate for the TTL transmission send and receive, which fortunately fixed our issue and enabled us to move on to the next phase of the project.

RESULTS

Our RFID checkout system quickly reads all the tags within the read field within a few seconds. This read time is greatly enhanced when the tags are in motion within the read field, instead of when they are stationary. The user interface has no issue at all with keeping up with the items scanned or recognizing button presses. Our RFID system initializes on startup within 1 s, as the On command is continuously sent to the reader to ensure the system is ready to go with the shortest delay possible.

Our RFID checkout system implements a look-up table that only recognizes valid RFID tags. As a result, we never experienced accuracy issues. Our SkyeTek M1 module had cyclic redundancy check (CRC) capability, but we disabled it and never used it due to the short time span of our project. In an industrial world with a lot of interference and millions of items scanned per day, implementing CRC would be crucial. Due to the nature of RFID technology in a retail setting (i.e., that every RFID tag is unique), the system would only recognize those items that haven't already been purchased. For example, if someone has a candy bar in their pocket that had been previously purchased, the system would not count that read item in the running total.

Our antenna design was initially much less accurate. We discovered that even a slight difference in dimensions yielded greatly different antenna characteristics. After modifying the capacitors in our design, with the help of our professor, we had peak resonance around 13 MHz, which was sufficient for our prototype.

REAL-WORLD APPLICATION

Our system would make it easy for everyday shoppers to zoom through the checkout lanes. For those with disabilities, this system would give them the potential to be more independent by simply passing by the reader without having to lift items, place them on a belt, or bag them. This would result in more convenience for all tiers of shoppers. However, the user interface for such an RFID checkout system must be properly designed using human-computer interaction (HCI) design principles. People accustomed to old technology or methods may find it hard to adapt to our system.

Current tag costs are borderline prohibitive to implement in everyday retail outlets. In the coming years, however, we expect this hurdle to be overcome with economies of scale. The transition from the UPC model will take some time, but we believe our prototype is an indication of what is around the corner. Great care must be taken when changing the entire basis of the retail industry, including managing employees and transitioning them in addition to adjusting consumer behaviors and expectations of the checkout process.

With any RF-operated system, interference is an issue. Our system can certainly be jammed at the 13.56-MHz frequency, which is a concern. The main problem, however, will be from interference from other reader/checkout systems operating nearby. As a result, the read range has to be kept to a minimum to ensure that readers near each other do not interfere. Also, it is crucial that customers be spaced properly in line so items aren't confused. Such concerns could be adequately addressed by using a marking tape to signify a read zone and utilizing proper employee instruction.

Last but not least, cashiers not only serve to ring up customers' items, but are a vital aspect of any firm's security and antitheft procedures. RFID tags can be hidden under certain metals to avoid detection, and customers could potentially hide items outside of the reader field upon checkout. Thus, new antitheft technologies will certainly have to be developed, in addition to the monitoring of customers, as is already done at any retail outlet. For example, another quick RFID scan could be completed while customers are leaving the store to ensure no unpaid items are present, combined with traditional employee greeters who look out for concealed products (i.e., those that may be hiding under metal enclosures to avoid detection). Assuming that entry and exits are secured, this would be a viable loss prevention procedure.

Additional losses that would be incurred as a result of this implementation would be offset by the reduced personnel costs for any retailer. Furthermore, retail employees could spread out in the store to assist customers in product selection (while monitoring for theft), instead of being concentrated at the front of the store as is customary in many of today's stores.

CHECKING OUT

We were successful in the implementation of our final project and outperformed our initial goals. Due to our luck on eBay, we were able to implement a multiple-read mainstream RFID checkout system at 13.56 MHz. Our system easily guides users through the checkout process by instantly scanning all the items in the reader field instead of scanning them one-by-one, like the present-day UPC model. Our custom-made antenna performed as expected, greatly increasing the read range possible over the stock antenna and providing a realistic prototype. Given our tight $75 budget, we are content with our accomplishments and we look forward to the realization of our project in the retail world in the coming years. ■

Authors' note: This project was completed as a senior design project for ECE 4760: Designing with Microcontrollers, taught by Cornell University senior lecturer Bruce Land. We are indebted to the class, the professor, and the teaching assistants for our success in prototyping our vision. In addition, Jigar's mother, who worked as a cashier for 15 years, served as inspiration for this project.

Jigar Shah (jjs367@cornell.edu) graduated from Cornell University with a Bachelor of Science degree in Electrical and Computer Engineering in May 2010. He is currently attending Princeton University for his Master's degree in Electrical Engineering.

Kevin Yang (ky238@cornell.edu) is a senior at Cornell University who is pursuing his Bachelor of Science degree in Electrical and Computer Engineering.

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2011/251.

RESOURCES

Cornell University, School of Electrical and Computer Engineering, "ECE4760: Designing with Microcontrollers," http://people.ece.cornell.edu/land/courses/ece4760, 2011.

SOURCES

ATmega644 Microcontroller
Atmel Corp. | www.atmel.com

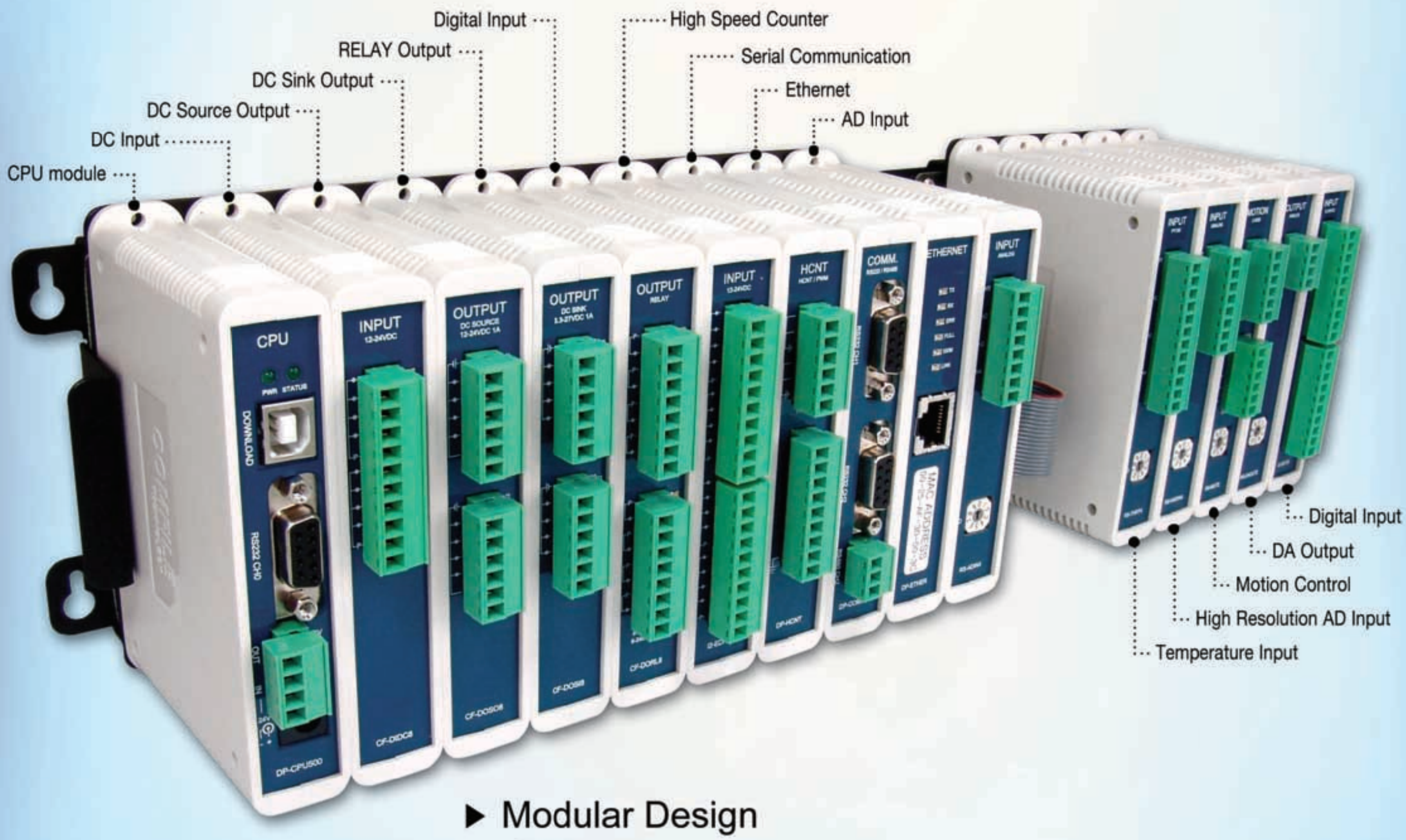
MAX233CPP Multichannel RS-232 Driver/Receiver
Maxim Integrated Products, Inc. | www.maxim-ic.com

SkyeModule M1 HF RFID reader and RFID kit
SkyeTek, Inc. | www.skyetek.com

Tag-it HF-I Transponder IC
Texas Instruments, Inc. | www.ti.com

C-Programmable Modular Industrial Controller

MOACON



- Modular Design
- C Programmable
- 32 bit ARM Processor
- USB Downloading and Debugging
- MOACON Studio (Free)

The MOACON is an ARM-based automation controller designed for an industrial environment. Choose from a diverse, feature-rich selection of I/O modules to customize the MOACON just the way you need it.

COMFILE TECHNOLOGY
1175 Chess Dr., Suite F, FOSTER CITY, CA 94404
call : 1-888-9CUBLOC (toll free)
1-888-928-2562
email : sales@comfiletech.com
www.comfiletech.com

FEATURE ARTICLE

by Custódio Gomes Barcellos (Brazil)

Virtual Sensing

Build a Virtual Alarm System

With the right parts and a clear plan of action, you can build a platform for a variety of virtual sensing applications. This microcontroller-based design, which can detect both nearby and distant movements, outputs sensory data on a basic monitor.

Engineers and programmers are increasingly looking to combine the real with the virtual, which is why "virtual reality" and "augmented reality" are such hot topics. Several months ago, I designed an entirely virtual alarm system that can detect movement without the use of a typical installed "physical" monitoring system. The design generates points on a monitor to show where movement has occurred. A point marks the spot and the value is read and maintained by software that constantly compares new readings and checks for changes. If an image changes at a point, the system detects the movement and activates the appropriate output. In this article, I'll describe how to build such a system with an Atmel ATmega8 microcontroller and a few external components (see Photo 1).

IMAGE SENSING

The purpose of the design is to capture an image with a camera, generate image detection points, and position them as cursors. After the image is read and marked by the points, it's sent to the monitor (see Figure 1 and Photo 2). An alarm is triggered if movement is detected.

Various sensor systems are on the market. For instance, sophisticated DVR systems and PC software can perform the aforementioned sensing tasks. Some drawbacks associated with retail systems are that they tend to be expensive, unadaptable to different applications, and PC-connected, which restricts portability (unless you use an expensive laptop). Another unsatisfactory characteristic of retail systems is that they tend to detect changes in either an entire image or in a single specific area. That is not suitable for my purposes, which is why I developed a relatively universal platform for use in a variety of projects.

You can use the design for projects ranging from surveillance systems to a simple four-button touchpad. A little creativity goes a long way. For instance, to integrate a virtual reality system without a PC, you could replace the joystick

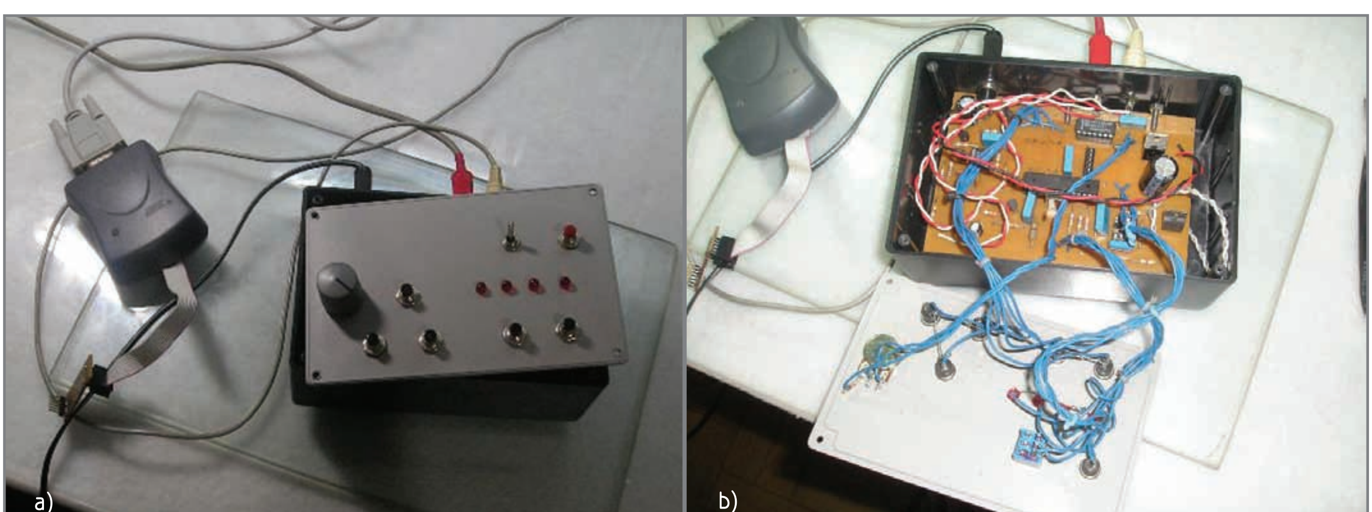


Photo 1a—The alarm system is mounted. b—This is the system's wiring and circuitry.

of an old Atari system with something like a Nintendo Wii controller. Your imagination is your only limitation.

FUNCTIONALITY

My virtual alarm system enables you to detect movement in open areas and at many specific points. You don't need to stare at a monitor. The system warns you when there is a change to an image. Plus, you can use it to monitor objects from either miles or micrometers away, depending on the lens used in the camera.

You can position the alarm system's cursor anywhere, such as in a window, a door, or a patio. The indicator determines which cursor moves. Putting the cursor around an object makes it impossible for someone to remove it without being detected. If an image is white, you can use a black cursor. If an image is dark, you can adjust the cursor to highlight specific spots.

Lastly, I designed the system so that it would be easy to use and install. If you're working with a closed-circuit TV, simply remove the camera cable that goes to the monitor and connect the device. The cable that comes out of the device goes to the monitor (see Figure 1).

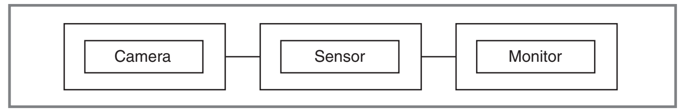


Figure 1—In a system with a closed-circuit TV, a camera cable can be removed and connected to the sensor. The cable then can be connected to the monitor.

SYSTEM DESIGN

The system consists of a few main sections: a sync separator, an amplifier, a video buffer, an ATmega8 microcontroller, an audio generator, keys, and LEDs (see Figure 2). A National Semiconductor LM1881 video sync separator receives video on pin 2. R16 and C12 form a low-pass filter to eliminate interference from the high-frequency video signal and the color burst. Capacitor C13 couples the signal to pin 2 which works as a DC clamp. Pin 3 outputs the vertical sync to the microcontroller. This signal marks the beginning of the video field. Pin 5 outputs to the burst gate pulse to be used for the horizontal timing by the microcontroller. This pulse marks the beginning of a visible line and it occurs right after the horizontal sync. The signal present on pin 2 has a fixed DC level, is stable, and is ready to go to the next stage.

The 4066 IC comprises four analog switches. One of the switches on the 4066 between pins 1 and 2 connects the video signal to the sampling capacitor C1. The pulse that controls the sampling is from the ATmega8, which generates the sampling pulse to be applied to pin 13 on the 4066. When this pin is at a high level, it connects pins 1 and 2; when it is at a low level, these pins are disconnected. The same generated pulse is also applied to pin 12 of the 4066 and connects pins 11 and 10, which connect the center tap of potentiometer R17 to the video signal output circuit. R17 is used to adjust the level of the point generated on the screen. Closer to VCC on the screen tends to be a white point. Closer to GND tends to be a black point. It is useful to adjust the point in scenes that are very bright or very dark so they can become invisible. Since the same pulse generated by the ATmega8 raises the point in the video and makes the sample and hold, it is not difficult to conclude that sampling occurs in each of the points visible on the screen. The charge stored in C1 enters the ADC0 pin 23. This pin is used to convert the voltage contained in the video image at that point and it's stored for use in the software.

The ATmega8's Timer0 is responsible for generating a continuous 1-kHz tone alarm, which is annoying enough to wake up anyone. The TIP120 transistor is a buffer between the low current of the microcontroller's output and the high current of an 8-Ω speaker.

Transistors T1 and Q3 form a video amplifier that isolates the video input from the video output, preventing the pulse being added to the output from contaminating the input signal. The 75-Ω resistor (R26) provides termination on the input. Transistor T2 buffers the video output. The output of this stage is the standard 1 V_{PP} signal with an impedance of 75 Ω.

I used an Arduino board with an

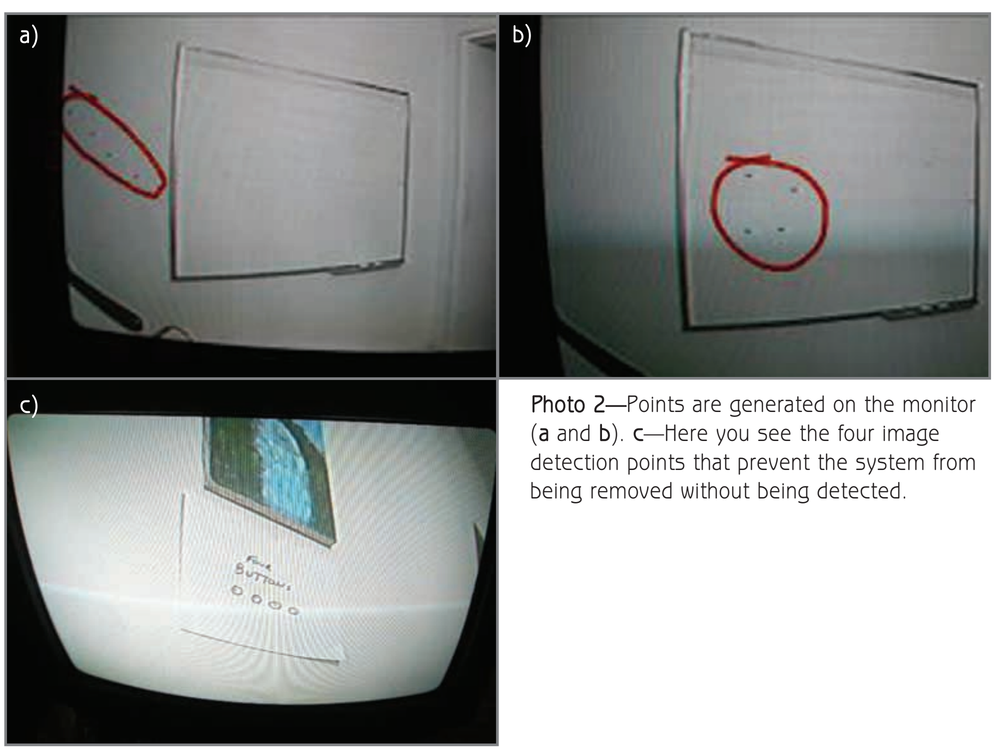


Photo 2—Points are generated on the monitor (**a** and **b**). **c**—Here you see the four image detection points that prevent the system from being removed without being detected.

ATmega8 with a 16-MHz clock. The ADC is set to use the internal reference for channel 0, and its interrupt is active. The pull-ups where the pins are connected to the switches and buttons are all active. Timer0 is configured to overflow once every ms and selectively generate an interrupt. INT0 is also enabled. LEDs 1 to 4 indicate which virtual sensor detected movement.

The system can be expanded, easily connect optocouplers to the LEDs, and activate anything. Buttons S6 and S5 adjust the horizontal position of the cursor on the screen. Buttons S4 and S3 are used to adjust the vertical position on the screen. Button S7 is used as a function key that selects what the cursor is being adjusted for. These buttons can be replaced by a joystick. Button S2 resets the alarm. Button S1 is a switch used in the system to permanently reset the alarm. It is useful when adjusting the cursors position; otherwise, the alarm would sound when the cursor moves on the screen during adjustment.

NTSC TIMING

I won't present you with a tutorial on TV technology. But let's briefly review the topic of synchronization.

A television picture is composed of two interlaced fields. The top of the screen starts writing an odd field, which contains all rows from one to 525, skipping all the even rows. Then the top of the screen starts scanning the even field, which contains all rows from two to 524, skipping all the odd rows. The first field (odd) writes, skipping from line to line. The second field (even) writes lines that were skipped by the first and skips lines that were already written. This arrangement seems complicated and even absurd today, but you know there are historical and commercial reasons for using NTSC. Each field has a length of 1/60 s. Therefore, a complete picture frame lasts 1/30 s. At the beginning of each odd and even field, a vertical synchronization pulse is generated to inform the receiver where the image begins. The vertical synchronization starts before the visible area and marks the beginning of the image at the top of the screen. Each line generates a horizontal sync pulse on the left side of the screen before the visible part of the image. This pulse marks the beginning of the scan line. It exists to tell the receiver where the line begins.

SOFTWARE

The software counts the vertical pulses coming through pin 6 (PD4) and stores them in variable `v`. This variable is used to count the fields and reset other variables at the beginning of each field. The pulses entering pin 4 (PD2) are used to count the horizontal lines and store them in the variable `h`. This variable is used to indicate the current line and it's compared with other variables containing the

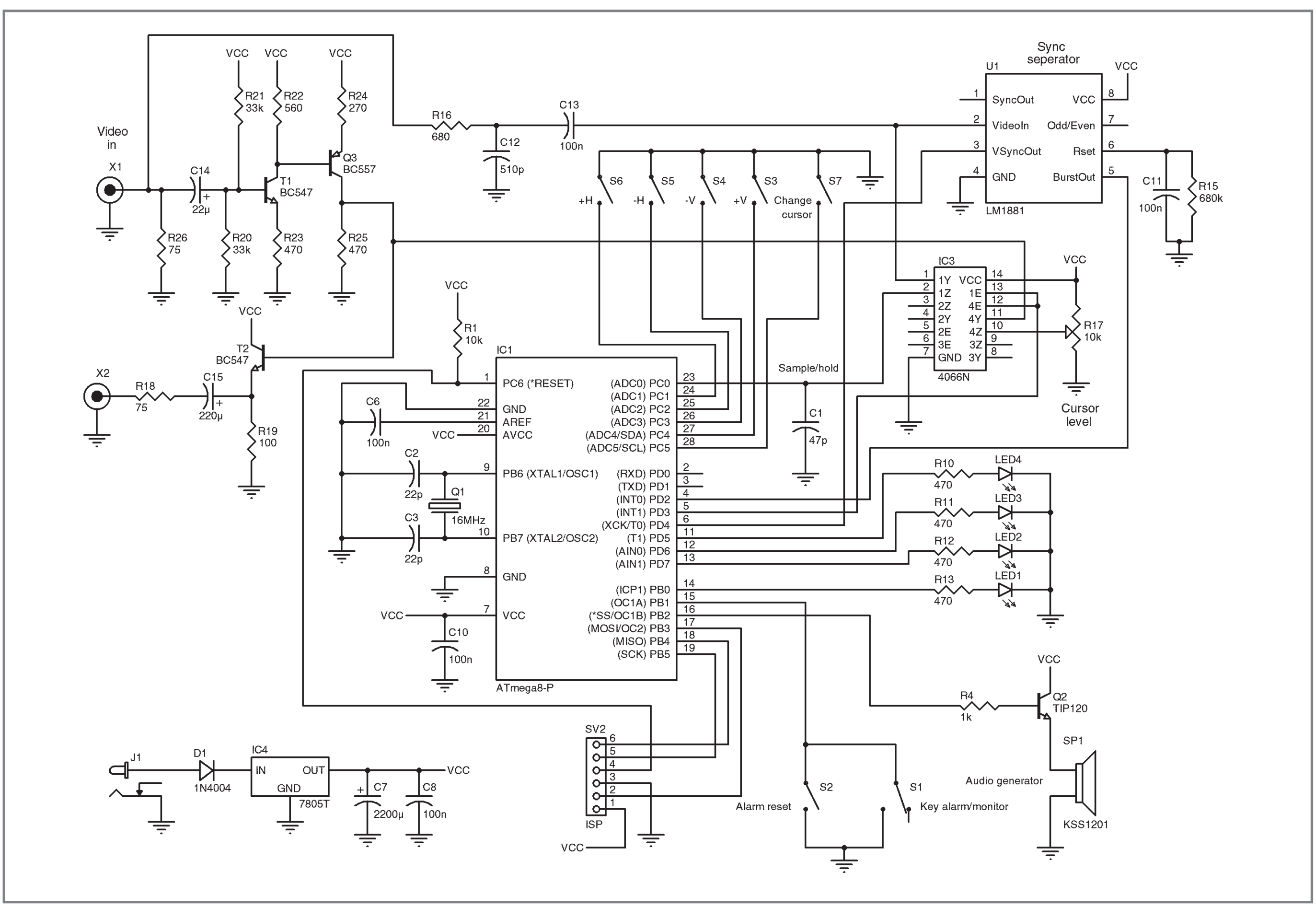
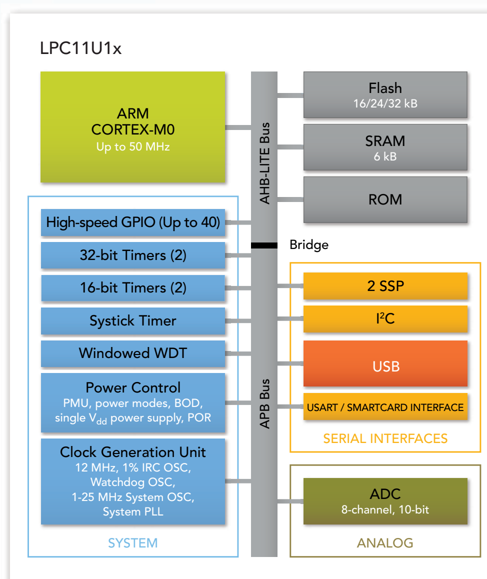


Figure 2—The main blocks of the system include the sync separator, amplifier, video buffer, ATmega8 MCU, audio generator, keys, and LEDs.

LPC11U00
Low cost Cortex-M0 USB solution

- SmartCard interface
- Small form-factor mobile and consumer applications
- Migration path to Cortex-M3
- Better approach to USB with configurable buffer management
- Up to 32K Flash and 6K RAM (2K USB SRAM)



www.nxp.com/microcontrollers

cursor position. The position of the cursor within the current line is adjusted in function v_pos(t), which is nothing more than a delay. With a low value passed to this function, the point on the image stays to the left side of the image. With a higher value, the point goes to the right side of the image.

In the function main() the ports and variables are initialized. The initial position the cursors should take is stored in variables cv1 to cv4 (vertical) and ch1 to ch4 (horizontal). Flags cfg1 to cfg4 mark which cursor the position buttons will affect and they are initially set for cursor 1. Subsequently, these variables will be affected by pushing the cursor selection button. When the vertical sync occurs (PIND4), h is reset to 1. Also, the work variables are set to 0 and the blanking period (blank lines between the vertical sync and the first visible line) is skipped with a delay of 1 ms. All of the reading of the cursor position buttons (PINC.1 to PINC.4) and selection of cursors (PINC.5) is done during the vertical interval.

Each horizontal pulse generates an external interrupt EXT_INT0. In this interrupt, if the line is equal to the horizontal line of the first cursor, if (h==ch1) then function v_pos() positions within the line and illuminates the point on the screen for the first cursor. The same pulse is the sample and hold. At this point, the program starts the A/D conversion. The variable v only enables the conversion to take place once every fourth field for each cursor, in round-robin fashion. The reason is that the A/D conversion is slow. If you place a cursor close to the other cursor, there is not enough time to complete the first conversion. The variables C1 to C4 indicate which cursor is read. When the conversion is completed, an A/D end-of-conversion interrupt is generated.

Inside this interrupt (ADC_INT) the value obtained from the conversion is read separately from the routine of each cursor (C1 to C4). This value is stored in ADCW and compared to the value previously stored in the variables VC1 to VC4 to detect whether there was a change in point since the last reading. The SENSOR definition controls the sensitivity of the detector changes. A small change in the image triggers the alarm. If it's too high, it becomes insensitive. A practical value is around 20. You can adjust it according to your needs. If there is no change in the level of the video cursor that's read, the LED indicator is off. Then the current value stored in ADCW is copied to VC1 in the case of the first cursor. If there is a change, the LED indicator is activated. In the program's main loop, the following instruction detects any enabled LED:

```
if (PORTD.7==1) ||
   (PORTB.0==1) ||
   (PORTD.6==1 ||
   (PORTD.5==1))
   {         }
```

If any are enabled, an interrupt overflow Timer0 is activated to generate the audio alarm, which will be disabled by the Reset button (PINB.1).

The interrupt TIM0_OVF serves to generate a square wave of 1 kHz in the audio output. This interrupt is only enabled when there is no conversion in progress and no pulse count. This prevents unnecessary interference or interrupt conflicts.

SYSTEM USE & LIMITATIONS

Once connected, the points are visible in the middle of the screen and the left side (see Photo 2). The cursor's selection switch is already for cursor 1, which is the initial condition. Make sure that the Reset switch S1 is active so that the alarm will not fire while you move the cursor on the screen. Move the cursor to the first position and then press the selection cursor's S7 button to set another cursor, and so on, until all are placed. Note that while S1 is active, if you use an object like a pencil to invade the area of a point, a short beep will sound and the cursor LED will flash individually for each cursor invaded. If S1 is disabled and an object invades the cursor's area, the alarm will be triggered and only the Reset buttons S1 or S2 will disable it.

This system detects changes in signal level (luminance) at specific points on the screen. This means that a moving object must have a different brightness. For example, if I put a point on a sheet of paper, and then put another sheet of paper with the same level of brightness at that point it will not be detected. But, if I put a paper that's lighter or darker there, it will be detected because its brightness level has changed. Take this into account when positioning cursors. One way to ensure detection is to place the points between a transition, such as a door and a floor of different luminosities. It is impossible for one person to pass through a door with two different luminosities at the same time. The detection in this case is guaranteed. If used as a virtual button, for example, choose a contrasting color so that your finger will stand out.

NEW APPLICATIONS

One way to circumvent the aforementioned limitations is to add a television RGB decoder to the original circuit. The sample and hold could be done in each RGB channel. The points could detect the brightness level, as well as changes in the color of the object. The circuit can be much more precise. For example, you can use it to detect only blue objects.

Other possible improvements include: replacing the LEDs with optocouplers so they can interface with any other electronics; increasing the number of points or cascading multiple virtual sensors to generate even more independent points; improving the sound circuit with something more elegant than a continuous tone of 1 kHz (e.g., a siren red alert); and embedding the circuit inside a monitor or TV.

The sky is the limit for this project. I can think of a few interesting applications. One idea is to interface the device to a PC's parallel port. You can create an application program to detect which of the four LEDs is activated and then play a sound for each. For example, four different musical notes can make a virtual musical instrument. Place the camera on top and record hand positions to play "air instruments."

Another application would involve a Cave Automatic Virtual Environment (CAVE), which is a common 3-D image generator. With a CAVE on your device, you could generate realistic audio feedback. For example, with the cursor positioned at the "edge" of the 3-D object generated, a sound could be emitted every time the object is "touched." So, if the 3-D object generated is water, a water sound is made when you touch it.

The design also could be used as a virtual kiosk to prevent vandalism. Imagine the points placed in virtual drawings of buttons on a steel bar or a wall. The user would press a virtual button instead of an actual button. Or, with some changes to the circuit and software, the system could help a person with severe mobility impairments who is capable of only moving his eyes or fingers. A camera with the cursors placed around either an eye or a finger would enable a person to control a sequence of prerecorded messages, providing basic communication. The system also could be used to control an object, such as a wheelchair.

I'm confident you could use my virtual alarm platform to develop any of the aforementioned applications. What's more, you can accomplish this without the exorbitant costs that are usually associated with this type of technology.

Custódio Gomes Barcellos (custodio.barcellos@gmail.com) runs a consulting firm located in Petropolis near Rio de Janeiro, Brazil. He has 20 years of electronic design experience, mainly in television and robotics systems.

PROJECT FILES

To download the code go, to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2011/251.

RESOURCES

B. Grob, *Basic Television and Video Systems,* McGraw-Hill, 1975.

J. Whitaker and K. Blair Benson, *Standard Handbook of Video and Television Engineering,* 3rd Edition, McGraw-Hill, 2000.

R. Barnett, S. Cox, and L. O'Cull, *Embedded C Programming and the Atmel AVR,* 2nd Edition, Delmar Cengage Learning, 2006.

SOURCES

Arduino boards
Arduino | www.arduino.cc

ATmega8 Microcontroller
Atmel Corp. | www.atmel.com

LM1881 Video sync separator
National Semiconductor | www.national.com

Elektor Shop

The world of electronics at your fingertips!

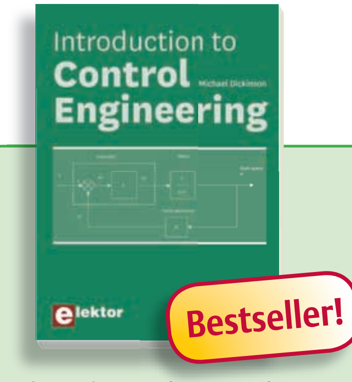
Books

Improve your skills in electronics, programming, networking and wireless design

Linux – PC -based Measurement Electronics

This book is intended as a highly-practical guide for people wishing to build measurement and control systems to be controlled by a local or remote Personal Computer running the Linux operating system. Both hardware and software aspects of designing typical embedded systems are covered in detail with schematics, code listings and full descriptions. There is no need to buy special equipment or expensive software tools in order to create embedded projects covered in this book. You can build such quality systems quickly using popular low-cost electronic components and free distributed or low-cost software tools. Many projects in this book are developed using Matlab® being a very popular worldwide computational tool for research in engineering and science. The book provides a detailed description of how to combine the power of Matlab® with practical electronics. With an emphasis on learning by doing, readers are encouraged by examples to program with ease; the book provides clear guidelines as to the appropriate programming techniques "on the fly". Complete and well-documented source code is provided for all projects.

Approx. 250 pages • ISBN 978-1-907920-03-5 • $47.60

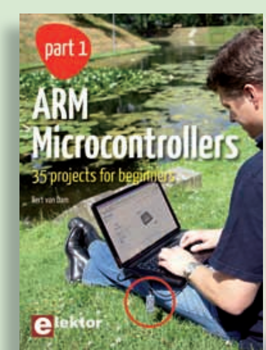


Solutions for control system applications

Introduction to Control Engineering

This book is intended as a source of reference for hardware and software associated with instrumentation and control engineering. Examples are presented from a range of industries and applications. Throughout the book, circuit diagrams and software listings are described, typical of many measurement and control applications. The hardware and software designs may be used as a basis for application by the reader. The book contains examples of PIC, PLC, PAC and PC programming.

164 pages • ISBN 978-0-905705-99-6 • $44.40



Associated starter kit available

ARM Microcontrollers

This is the perfect book for people who want to learn C and who want to use an mbed ARM microcontroller in an easy and fun way. The mbed NXP LPC1768 uses cloud technology. This means you do not need to install software on your PC in order to program the mbed! The only thing you need is a browser and a USB port on your PC. No previous experience or knowledge required. You can get access to your project from any PC anywhere in the world and continue working on it. When you are done a few mouse clicks transfer the program to your mbed hardware.

250 pages • ISBN 978-0-905705-94-1 • $47.60

Prices and item descriptions subject to change. E. & O.E

CD/DVD-ROMs

All articles in Elektor Volume 2010

DVD Elektor 2010

This DVD-ROM contains all editorial articles published in Volume 2010 of the English, Spanish, Dutch, French and German editions of Elektor. Using the supplied Adobe Reader program, articles are presented in the same layout as originally found in the magazine. An extensive search machine is available to locate keywords in any article. With this DVD you can also produce hard copy of PCB layouts at printer resolution, adapt PCB layouts using your favourite graphics program, zoom in / out on selected PCB areas and export circuit diagrams and illustrations to other programs.

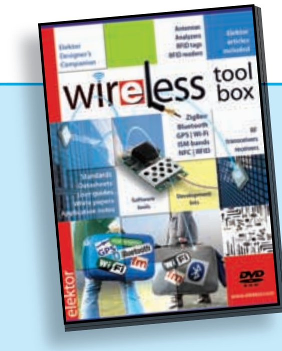
ISBN 978-90-5381-267-9 • $37.90

Elektor's Components Database

CD ECD 6

This CD-ROM gives you easy access to design data for over 70,000 components! The program package consists of eight databanks covering ICs, transistors, diodes and optocouplers. A further eleven applications cover the calculation of, for example, zener diode series resistors, voltage regulators, voltage dividers and AMV's. A colour band decoder is included for determining resistor and inductor values. All databank applications are fully interactive, allowing the user to add, edit and complete component data.

ISBN 978-90-5381-258-7 • $40.20



Packs 1.2 Gigabyte worth of documents and tools

DVD Wireless Toolbox

On this DVD-ROM you'll find a number of technical documents and tools that will enable you to add wireless data exchange to your electronics systems. In accordance with the principle of our Toolbox series, we've brought together technical documentation (spec. sheets, application notes, user guides, etc.) on various devices according to the frequency and/or protocol used. All of the documents are PDF files. Finally, this Wireless Toolbox DVD contains a collection of articles on this topic (RFID, xBee, DCF77, GPS, infrared, etc.) that have appeared in Elektor magazine.

ISBN 978-90-5381-268-6 • $46.00

Elektor is more than just your favorite electronics magazine. It's your one-stop shop for Elektor Books, CDs, DVDs, Kits & Modules and much more!

www.elektor.com/shop

elektor

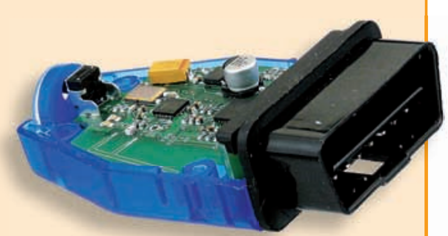
Elektor US
PO Box 180
Vernon, CT 06066
USA
Phone: 860-875-2199
Fax: 860-871-0411
E-mail: order@elektor.com

Pico C Meter

Even upmarket digital multimeters boasting a built-in capacitance meter are useless if you want to check out tiny capacitances like 2.7 pF or 5.6 pF. Usually, you're tied to a lowest measuring range of 2000 pF, which is a good laugh to RF designers and radio amateurs. Although at 3.5 digits the DMM's resolution is 1 pF, any measurement below 200 pF or so will produce coarse if not ridiculous results. Pico C does a far better job. Beating many DMMs hands down, this little instrument easily measures capacitances down to fractions of a picofarad.

Kit of parts incl. Elektor Project Case, programmed microcontroller, LCD and PCB

Art.# 100823-71 • $118.40



Wireless OBD-II

The cheapest way to diagnose faults on a modern car is to connect its OBD-II interface to a (notebook) PC running suitable diagnostics software. However, a wired connection is not always the most suitable, and selfcontained OBD testers are a rather expensive and less flexible alternative to using a PC. An interesting option is a wireless OBD interface with a radio interface to a PC: this homebrew solution allows the choice of using either Bluetooth or ZigBee.

OBD2- Zigbee or Bluetooth interface kit with all parts and enclosure

Art.# 100872-71 • $201.70 (Zigbee)
Art.# 100872-72 • $201.70 (Bluetooth)

Kits & Modules

by Doug Commons (Canada)

The Chameleon Platform

Low-cost Hardware Development with a USB AVR and CPLD

A low-cost hardware development platform is a great starting point for many projects. Here you learn how to develop with a USB AVR and CPLD. This platform is directly supported by a large number of open-source projects that will enable you to work on a variety of interesting designs.

The Chameleon is a versatile low-cost ($20), 8-bit hardware development platform that changes to meet your needs. The board supports a 32-KB flash-based Atmel AVR processor and a 1,600-gate Xilinx XC9572 complex programmable logic device (CPLD). The software support packages are free and include AVR GCC, Xilinx ISE, and a support library for USB operations. This article introduces the hardware and software. The hardware can be used as an AVR ISP, a JTAG programmer, a serial-to-USB CDC, or an embedded component in your next design. (Code to support these features is provided on the *Circuit Cellar* FTP site.)

Some of the features of the Chameleon are its ability to program itself; both the AVR and CPLD can be programmed without the addition of any other hardware. Unlike other devices, the Chameleon can bootstrap itself through self-programming, which means the development cost is limited to only the cost of the board and components. A lot of AVR-only USB solutions are available, but the addition of a CPLD makes the Chameleon unique. The addition of the CPLD required me to develop a JTAG programmer within the hardware. This enables the CPLD to be programmed in place. It also enables the Chameleon to program its own—or any

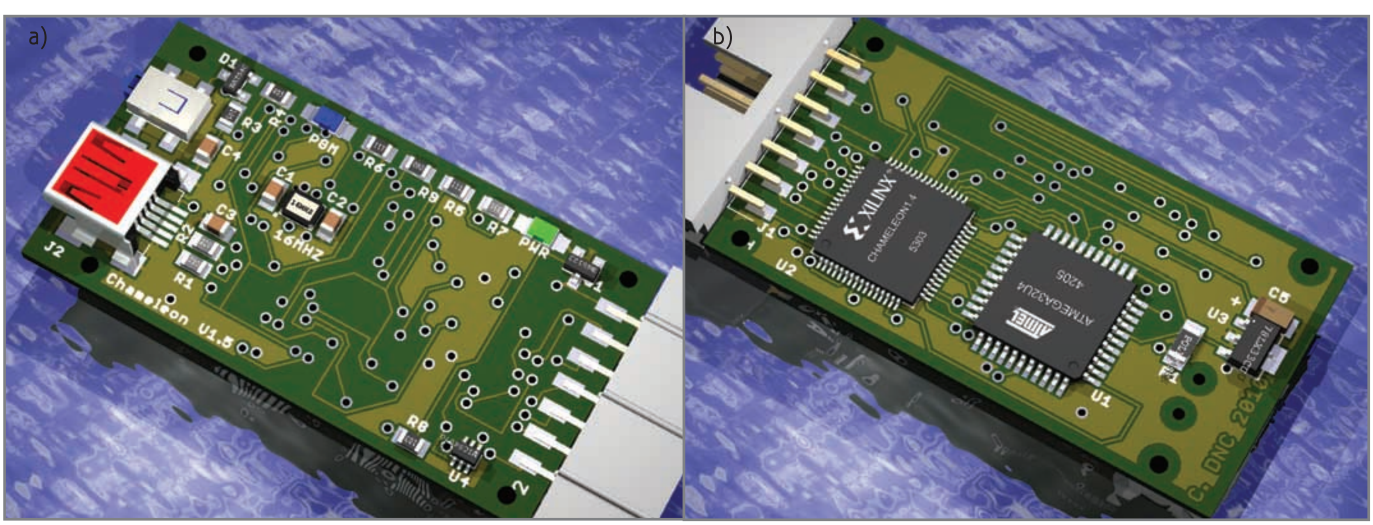


Photo 1—Eagle 3D was used to generate these top (a) and bottom (b) images.

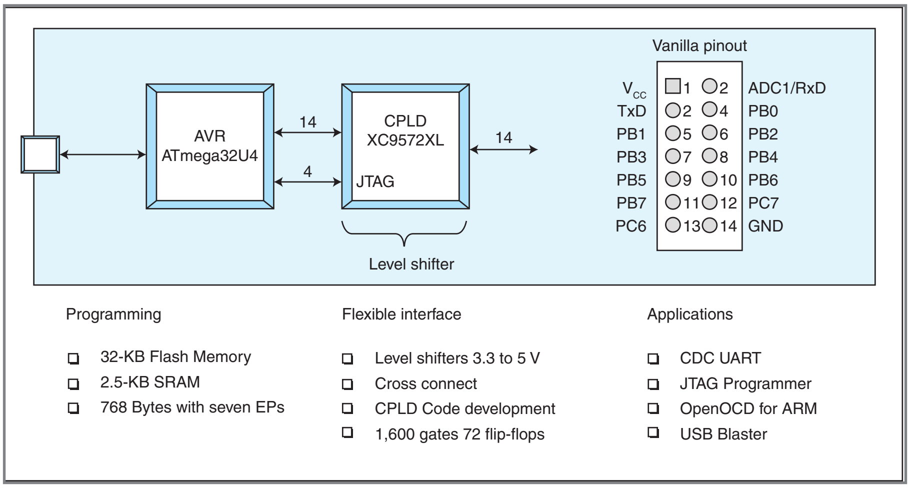


Figure 1—The ATmega32U4-based Chameleon is simple yet programmable.

externally connected FPGA or CPLD—using the JTAG interface. This is significant. For the cost of the AVR (approximately $3), you get a universal USB JTAG programmer along with a USB CDC interface. Porting this design to a different CPLD or FPGA is trivial and cost effective.

I also developed a board that uses a Xilinx XC3S50AN FPGA instead of the XC9572XL CPLD, which gives you 50,000 gates instead of 1,600. Other design variants are possible. Additionally, the development environment supports all major operating systems, making development fast and efficient. The addition of the CPLD also solves the problem of level shifting the I/O for 3.3- to 5.0-V operation while enabling totally flexible pin mapping. The 1,600-gate CPLD also enables you to seamlessly connect the Chameleon to a device under test without requiring any additional glue logic.

The Chameleon development platform is straightforward, but it isn't useful without code support. I developed a Gecko USB library to support the Chameleon hardware. The Gecko library implements all of the functions needed for a high-speed USB 2.0 interface and includes a serial emulation using the CDC class. USB access paired with AVR processing opens up a new level of development convenience and efficiency. The other great thing about the Chameleon is that it is directly supported by a large number of open-source projects or it can readily have projects ported to the hardware. For starters, the AVR is well supported by the GCC compiler, so it is easy to port C-based projects to this hardware. Example projects that are directly supported include LUFA and Opendous, which provide a number of interesting designs.

Another project that I find interesting is the UrJTAG development at SourceForge.net. This host code enables the PC to program a variety of FPGAs using a USB Blaster clone. The code to support the USB Blaster clone is in fact the same code that enables the Chameleon AVR to program the CPLD. Kolja Waschk originally developed this code for the FX2 processor and I ported it to the Chameleon.

CHAMELEON LIBRARIES

The Chameleon design is supported by rich open-source development that has enabled me to create a versatile development platform. First, there are no less than four USB libraries that support this hardware. Of course, Atmel has its own USB support code complete with all sorts of legalese. In a more open environment, the LUFA project provides a well-supported open-source USB library. There is also Stefan Salewski's USB interface. Finally, I have my own USB code called Gecko, which is targeted at creating a USB library to hide as much of the USB complexity as possible.

The Gecko library is a simple USB interface that handles all the standard request functions while requiring minimal user knowledge of the USB interface. The goal of the library is to hide complexity, and as a result, the library features are limited to only those that are required. This has the advantage of requiring only a small number of API calls to manage the device. If you want to use all the available details of the USB interface, I recommend the LUFA project. If you are looking for simplicity, the Gecko library is a good choice. The Gecko library is provided as open-source GPL code and can be used without restriction. Gecko is precompiled so it can be added to your project from within your makefile. The library is precompiled and only needs to be linked. This speeds the build time because the library does not need to be rebuilt every time you update your project.

SIMPLICITY & FLEXIBILITY

The hardware for the Chameleon is deceptively simple, yet it provides tremendous flexibility. The AVR processor provides a C hardware environment for developing firmware with the AVR GCC compiler. The AVR includes an integrated bootloader so that no additional programming hardware is required.

The CPLD performs a number of functions. First, it level shifts between the 5.0-V AVR and 3.3- to 5-V targets. The CPLD also enables the user to map the AVR's I/O to any of the 14 connector pins. This feature enables the user to avoid a multitude of custom cables to attached peripherals. This is particularly useful when the Chameleon is used as a programmer and must map its I/O to the target device pinout. The I/O is fully programmable, which allows for an optimized PCB layout since almost all of the CPLD pins can be configured.

Eagle 3D was used to generate the image shown in Photo 1. It is a useful tool since it enables me to view the finished board long before it is even manufactured. I use it to check the silkscreen and for component interference prior to finalizing the design. When the PCBs arrive and I am assembling the board, I use Eagle 3D to identify component polarization and orientation. And, for the secondary side, which typically does not have a silkscreen, it helps in component placement.

The Chameleon design is based on Atmel's ATmega32U4 microcontroller. The advantage of the ATmega32U4 is that it has 32 KB of flash memory and 2.5 KB of SRAM and supports seven endpoints versus five for the AT90USB162. This means it's possible to use the CDC concurrently with your application, which is a great debugging asset. Additionally, the larger endpoint memory increases the USB bulk transfer performance to 375 KBps since most of the delay is due to the USB endpoint turnaround time. The ATmega32U4 also provides enough features and performance to be useful as a low-cost processor platform.

The hardware for the Chameleon consists of two main components and a few support parts (see Figure 1 and Figure 2). The AVR provides USB support and a generic processing environment. The AVR was selected because of its bootloader, which eliminates the need for an additional programmer to download code. The AVR becomes a standalone, flash-memory based processing environment. The CPLD is the second component and functions as a cross connect, level shifter, and generic programmable logic. With the UrJTAG support code, the CPLD is also field-programmable without needing any additional programming hardware. The CPLD enables small 1,600-gate designs to be developed.

The Chameleon powers the AVR at 5.0 V, extracted from the USB bus, to enable it to operate at 16 MHz. The AVR's speed is derated as voltages are reduced, as are all processors. To maximize performance, the AVR is powered at 5 V for fast processing and 16 MIPS performance.

A 3.3-V regulator provides 300 mA to power the board and any attached devices. The AVR and CPLD also can be powered directly using the AVR's internal regulator; however, this would limit the power to 25 mA. Target power can be derived from CPLD pins programmed high. Each CPLD pin can deliver 20 mA, so a design with two power and ground pins has 10 I/Os and a maximum current draw of 40 mA. Alternatively, two FETs have been provided so that higher currents can be sourced. The FETs are dedicated to pins 1 and 14 of the I/O header, which is less flexible from a pin mapping point of view. To compensate for the pin mapping restriction, FETs offer a 3.3-V, 300-mA voltage source assuming the host enumeration allows it.

A polyfuse resettable fuse physically limits the power to 290 mA to keep short circuits from frying the host PC's power supply. The polyfuse turns off if too much current is drawn. Unlike regular fuses, the polyfuse can be reset. Therefore, if the short is removed and the device is repowered, the polyfuse is able to operate normally and conduct again.

A special reset circuit enables a single Reset button to act as a reset or to activate the bootloader. If the Reset button is pressed for less than 2 s a reset occurs. Holding the Reset button longer causes the bootloader to start. The circuit works by attaching an RC time constant to the HWB line. A brief button press won't pull the HWB line low so a reset occurs. A longer button press has the effect of pulling the HWB low. When the button is

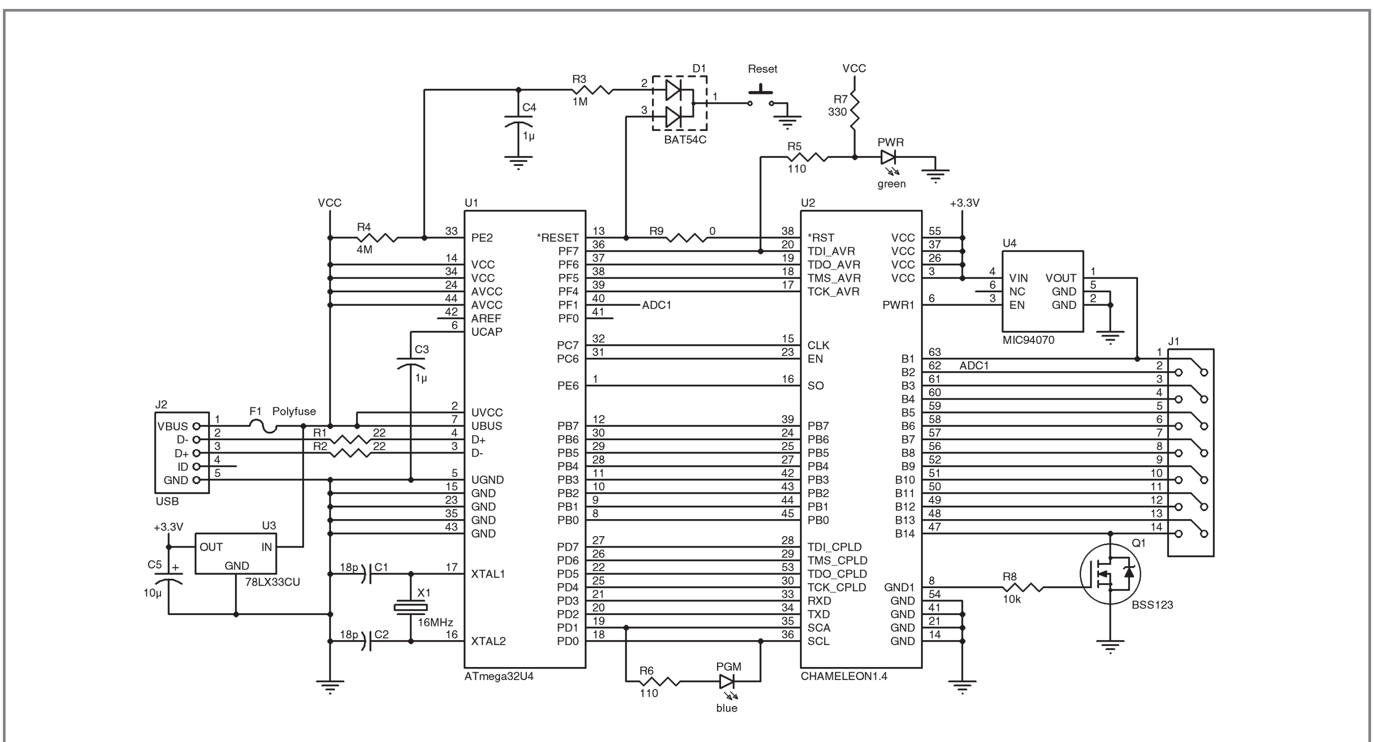


Figure 2—The Chameleon V1.5 circuit

```
jtag> cable usbblaster ftdi 9fb:6001
Connected to libftdi driver
jtag> detect
IR length: 8
Chain length: 1
Device Id: 01011001011000000100000010010011 (0x0000000059604093)
  Manufacturer: Xilinx
  Part(0):         XC9572XL
  Stepping:     vq64
  Filename:     /usr/local/share/urjtag/xilinx/xc9572xl/xc9572xl_vq64.bsd
jtag> svf c:/xilinxsvf/mkiiispv1.4.svf progress
Warning: USB-Blaster frequency is fixed to 12000000 Hz
Parsing   5160/5160 (100%)
Scanned device output matched expected TDO values
Time = 5.843750
Tx = 380777 Rx = 13847
Speed 67.529241 kbps
Total chunks = 6889 6.889000 seconds
chunks64 = 4638. 4.638000 seconds
chunks128 = 1179 1.179000 seconds
chunks256 = 1072 1.072000 seconds
jtag>
```

Figure 3—Downloading an *.SVF file using UrJTAG

released, RST- on the AVR goes high, but the HWB remains low for about two time constants, or 2 s. The sequence of RST- rising before the HWB line raises causes the bootloader to run. This design means you can reset the processor or load new code using "Flip" and the bootloader.

The Chameleon's LEDs are also wired in an odd way to provide more flexibility. First, the power LED is wired to the AVR so that driving the PF7 pin low causes the LED to turn off. Note that you must disable the JTAG port for PF7 to work as a general purpose I/O port. This is done by writing `MCUCR |= 1<<JTD;` twice in a row to the AVR.

What about the power indicator? If you hold the Reset button, the power status is always displayed because the AVR doesn't drive its pins when in reset. Note that the CPLD can also be programmed to drive the LED. As a consequence, if the CPLD is programmed with `TDI_AVR <= '0';`, the power LED will remain off until a new CPLD image is loaded.

The second LED is wired to two AVR pins (anode PD1 and cathode PD0) to enable it to be used as a photodetector or LED. As an LED, the cathode is held low and the anode can be programmed high or low to turn the LED on or off. The photodetector works by charging the reverse-biased LED junction to VCC and then reading the charge to see how long it takes to discharge. The LED PN junction acts as a photo capacitor. When light of the diodes frequency shines on the junction, it conducts and discharges the capacitor. (This reminds me of the "flux capacitor" from the movie *Back to the Future*.)

Timing the discharge rate indicates the ambient light level. The faster the discharge times, the brighter the light. Interestingly, green and yellow LEDs have similar discharge times, while a blue LED discharges much more slowly. I presume that either the junction has a higher capacitance or the spectral content of the light contains very little blue in the diode's frequency range.

CPLD

The AVR directly supports programming the Chameleon CPLD. This feature is key to the use of this design since it enables self-programming. The UrJTAG project on SourceForge.net provides an open-source environment that supports programming CPLD, FPGA, and ARM processors via JTAG. The UrJTAG host code supports a number of programmers, including the USB Blaster.

The Chameleon programmer is a port of the USB Blaster onto the ATmega32U4. The AVR receives and transmits USB packets with the host and toggles the low-level JTAG bits as required. From a performance perspective, you can't just send ones and zeros 1 bit at a time or the USB latency will kill the performance. To overcome this limitation, the UrJTAG programmer supports two communication types. First, bit bang is supported for transfers that need an immediate response and can't be queued (very slow). In addition, when the data to program is an output or input stream, the host can queue up a series of data bytes that will be played out bit by bit by the AVR programmer. This improves performance by eliminating the USB latency by packing more data into each USB packet transfer.

Connecting a USB 2.0 high-speed hub between the host and the Chameleon can create an interesting performance enhancement. The addition of the hub means that the USB sampling interval can increased from 1 ms to 125 µs. The original UrJTAG code without any code improvements takes 55 s to program the CPLD. With the hub this drops to 15 s. Finally, I have added a few improvements to UrJTAG to drop this time to 5 s, which seems pretty respectable. Testing with a 50,000-gate XC3S50AN FPGA took 5 s for an SRAM image and 55 s for a flash burn and verify.

To write code for the CPLD, I used the free Xilinx ISE WebPACK to compile my VHDL code. Even if you are not familiar with VHDL, the design examples I provided are a good starting point. Modifying the examples is a straightforward way to develop new custom applications. The development of a CPLD starts with the creation of a new project in the ISE. I typically import an old VHDL design as a template and modify the code as required. The most important step after your VHDL compiles cleanly is to build an SVF file to download to the CPLD.

Listing 1—The simplicity of using the Gecko library

```
#include "../Gecko_library/include/AVR_Chameleon_V1.4.h"    //Hardware definition
#include "../Gecko_library/include/usb_descriptors.h"       //Gecko library
#include "../Gecko_library/include/CDC_api.h"               //CDC support

//Connect USB driver to stdio
FILE usb_str = FDEV_SETUP_STREAM(usb_std_putchar, usb_std_getchar, _FDEV_SETUP_RW);

int main(void)
{
        usb_init(16, 0x03EB,0x2021);                        //Start USB timers and tasks
        CDC_init();                                         //Start CDC
        usb_putchar(usb_getchar());                         //Wait for host to be ready
        stdout = stdin = stderr = &usb_str;                 //Connect I/O stream
        printf_P(PSTR("\nHello World!\n"));                 //printf from flash
        printf("\nHello World!\n");                         //printf from SRAM
        while(1){
                usb_putchar(usb_getchar());                 //Echo characters to host
        }
}
```

The CPLD is fully programmable, so the I/O map must be defined through a usage configuration file (UCF) that maps the design inputs and outputs to the correct pins on the board. Several examples of the UCF are provided. These are all derived from the schematic circuit pinouts. The final SVF file is supported by UrJTAG and can be downloaded to the Chameleon using the UrJTAG host application (see Figure 3).

To support clocked state machines in the CPLD, the AVR's crystal can be driven out on AVR pin PC7 by programming the AVR's CLKO fuse bit. This provides the CPLD with a 16-MHz master clock. The Chameleon-V1.4_test.vhd image is provided as a state machine design example for developers. The test code simply provides a clocked 16-kHz output and pulses the PGM LED at 4 Hz. It is also possible to clock the CPLD using PC7 as an AVR toggle bit to clock the CPLD if you don't have access to an AVR programmer to program the fuse bits. A third alternative is to provide a clock source on one of the B port pins of the CPLD via the 2 × 7 connector.

HELLO, WORLD!

How does all of this work? There are potentially three steps to creating every application. First, the AVR must be programmed with UrJTAG to enable the CPLD to be programmed. Second, the CPLD is configured by downloading an *.SVF file. Third, the application is loaded into the Chameleon to operate as required.

I foresee several ways to use this hardware. First, as an AVR development board using the Gecko CDC capability along with this hardware, you can download and test AVR code with nothing more than this hardware. The C code is compiled with GCC and downloaded with the DFU bootloader over the USB cable. I find myself frequently wanting to test a new device or chip. With this hardware, I don't need to build anything. I just wire the Chameleon to the device under test and smile knowing that I have a stable working environment to test the unknown secrets of the new part I am testing. The need to build and test a new processor and host connection is eliminated with the Chameleon board providing the AVR processing, CDC connectivity, and AVR I/O in one inexpensive package.

For an example of how to use the Chameleon as a processor platform, have a look at the "Hello, World" code in Listing 1. The code contains a few `include` files, which enable a high level of functionality with little user code. The code waits for a serial character to synchronize to the host, prints "Hello World!" to the host console, and finally loops echoing characters to the host. The Chameleon_CDC.inf file is required to be loaded on a Windows host for it to enumerate the device. This is accomplished by copying the Chameleon_CDC.inf file to the host and right-clicking the file to select "install." Alternatively, you can copy the Chameleon_CDC.inf file to the c:\windows\inf\ directory. After that, you simply access the Chameleon from a terminal interface, such as HyperTerminal.

The other main application of the Chameleon hardware is as a CPLD development platform using the free Xilinx ISE WebPACK. The Chameleon is a low-cost starting platform for learning to code in VHDL or Verilog. The major stumbling block for someone who just wants to test the CPLD water is that they need a JTAG programmer, which most people don't have on hand. The Chameleon solves this by embedding a JTAG programmer into the AVR so the Chameleon can self program itself or an external FPGA. You are now free to try out the world of programmable logic.

Finally, when using the AVR as a testbed for new hardware, there are often applications that require glue logic to complete the interface. Using the CPLD to fill this need eliminates any additional discrete logic in the design. For example, if you wanted to connect the I^2S interface of a voice codec to the AVR that doesn't directly support I^2S you could use the CPLD to map the interface to the AVR's SPI port. Also, don't forget that CPLDs are excellent at adding timers and PWMs to your design, so don't overlook the possibility.

TOOLS & CODE

The Chameleon's toolset must be installed prior to using it with your own code. Installing the GCC compiler for the AVR is easy on a Windows machine. Simply download the WinAVR installer and the compiler tools will be made available. When using Linux Ubuntu, you can download the latest AVRGCC code "gcc-avr 1.4" from the "universe" repository.

For CPLD development on the XC9572XL, I installed Xilinx's ISE 11.4 WebPACK. Unfortunately, I haven't had much luck getting it to run on Ubuntu although it is supported on Linux. This is mostly because I haven't spent the time working on getting it to run. Good luck!

The complete environment of the GCC and the VHDL compiler will enable you to develop any aspect of code for the Chameleon. Compiling UrJTAG for a Windows environment can be difficult, so I have provided a Windows executable. The UrJTAG programmer code that programs the Chameleon CPLD can also be used to program external devices, like FPGAs, using the 2 × 7 header. This is one of the first applications of the Chameleon as a general-purpose USB JTAG programmer. When using an ATmega32U4 processor, it is also possible to perform the CDC function concurrently with UrJTAG. This means you can develop a design to program a CPLD or FPGA and provide a serial port replacement. Since all of the functionality is embedded into the AVR hardware and code, making a low-cost, one-chip solution is easy.

WHAT NEXT?

With the hardware flexibility provided by the Chameleon, it's hard to know where to go next. As I mentioned, you can simplify the hardware to a single-chip solution for embedded applications. You can cut off the Chameleon's CPLD tail and everything will work. A popular project like the Arduino could become a single-chip implementation based on these USB libraries. My hope is that the user community will adopt this platform and publish more interesting applications. In the meantime, I intend to create a few applications of my own.

The Chameleon currently supports a number of great applications designed to support low-cost hardware development. I intend to use this hardware as an open-source platform for many future projects as either a standalone design or embedded into another circuit. As an embedded platform, I envision using the design as an integrated on-chip debug and communication device class interface for an ARM processor. Or it could be a replacement for a UART and JTAG programmer on an FPGA development board.

You can easily remove each piece of the current design to provide an affordable solution for an application. For an embedded application, only the AVR and a crystal are required. Power can be 3.3 or 5 V. For FPGA development, it is possible to add a modern USB JTAG programming interface and CDC UART for less than $3, which is less than the cost of a typical serial port or FTDI solution. Furthermore, it frees up a physical UART port on the AVR. Future standalone applications I am working on include a USB protocol analyzer using this design and an XC3S50AN FPGA.

The ultimate goal for this project was to provide an inexpensive open-source development environment. I think this design achieves that goal. The next step is to port more software and hardware projects to this design using the Chameleon as a base platform. ■

Doug Commons (dougcommons@rogers.com) is a hardware designer with more than 25 years of experience developing telecom products. His interests include embedded programming and open-source development. Doug is also interested in making hardware development more affordable to nonprofessionals and is working on building tools and development processes.

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar/pub/Circuit_Cellar/2011/251.

RESOURCES

Atmel Corp., "USB Mass Storage Support," 2005, www.atmel.com/dyn/resources/prod_documents/doc7549.pdf.

AVR Opendous, http://code.google.com/p/avropendous/w/list.

Four Walled Cubicle, LUFA 2011, www.fourwalledcubicle.com/LUFA.php.

S. Salewski, "Free USB firmware for Micro-Controller," www.ssalewski.de/AT90USB_firmware.html.en.

UrJTAG, http://urjtag.org.

WinAVR, http://sourceforge.net/projects/winavr and USB JTAG http://ixo-jtag.sourceforge.net.

Xilinx, Inc., "ISE Design Suite 11: Installation, Licensing, and Release Notes," 2010, www.xilinx.com/support/documentation/sw_manuals/xilinx11/irn.pdf.

SOURCES

ATmega32U4 Microcontroller
Atmel Corp. | www.atmel.com

Eagle 3D software
www.matwei.de | www.matwei.de/doku.php?id=en:eagle3d:eagle3d

XC9572XL CPLD, XC3S50AN FPGA, and ISE 11.4 WebPACK
Xilinx, Inc. | www.xilinx.com

ICbank
Gateway to Electronics

MP3P DIY KIT, Do it yourself

(Include Firmware Full source Code, Schematic)

· myPIC

Only
$160 qty 100 **$220** qty 1

Powerful feature

- MP3 Encoding, Real time decoding (320Kbps)
- Free charge MPLAB C-Compiler student-edition apply
- Spectrum Analyzer
- Application: Focusing for evaluation based on PIC
- Offer full source code, schematic

Specification
Microchip dsPIC33FJ256GP710 / 16-bit, 40MIPs DSC
VLSI Solution VS1033 MP3 CODEC
NXP UDA1330 Stereo Audio DAC
Texas Instrument TPA6110A2 Headphone Amp(150mW)
320x240 TFT LCD
Touch screen
SD/SDHC/MMC Card
External extension port (UART, SPI, I2C, I2S)

· myWave (MP3 DIY KIT SD card Interface)

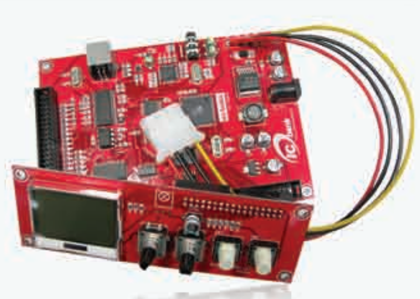
Only
$150 qty 100
$200 qty 1

Powerful feature

- Play, MP3 Information, Reward, forward, Vol+/-
- Focusing for MP3 Player
- SD Card interface
- Power: battery
- offer full source code, schematic

Item	Specification
MCU	Atmel ATmega128L
MP3 Decoder	VS1002 / VS1003(WMA)
IDE Interface	Standard IDE type HDD(2.5", 3.5")
Power	12V, 1.5A
LCD	128 x 64 Graphic LCD
Etc	Firmware download/update with AVR ISP connector

· myAudio (MP3 DIY KIT IDE)



Only
$180 qty 100
$220 qty 1

Powerful feature

- Play, MP3 Information, Reward, forward, Vol+/-
- Focusing for full MP3 Player (Without case)
- IDE Interface
- Power: Adapter
- Offer full source code, schematic

ICbank
Gateway to Electronics

www.icbank.com

ICbank
Gateway to Electronics

3 PORT INTERFACE
RS-485 to Ethernet Converter

Only $170

RS-485 to Ethernet Converter

Powerful feature

- Protocol converter RS485 between Ethernet
- Offer TCP/IP Communication to Devices with RS485 I/F

Specification

Network	: TCP, UDP, DHCP, ICMP, IPv4, ARP, IGMP, PPPoE, Ethernet, Auto MDI/MDIX , 10/100 Base-TX Auto negotiation (Full/half Duplex)	
Serial	: RS485 3 Ports, 1,200~115,200 bps, Terminal block I/F Type	
Control program	: IP Address & port setting, serial condition configuration, Data transmit Monitoring	
Accessory	: Power adapter 9V 1500mA, LAN cable	
Etc	: - DIP Switch(485 Baud Rate setting)	- LED: Power, Network, 485 Port transmission signal

by Ed Nisley (USA)

Plastic Extruder Thermal Analysis

Measuring and understanding the thermal performance of the Thing-O-Matic 3-D printer's extruder head provides some interesting results. This column explains what was measured, how it was done, and what was found. You may find the techniques beneficial—even for projects with less strenuous thermal requirements.

Shortly after attending the Botacon robotics convention in New York last December, I bought a MakerBot Industries Thing-O-Matic 3-D printer kit. As fate would have it, the carton arrived on Christmas Eve: what a nice present!

A 3-D printer performs additive machining: starting with an empty build platform, the printer adds material to create the finished part. In contrast, my Sherline CNC milling machine performs subtractive machining: removing material from an oversized work piece to reveal the finished part. Additive machining can produce parts with internal voids, holes at any orientation, and acute-angle junctions that a subtractive milling machine simply can't duplicate.

The Thing-O-Matic (known as a TOM), in common with most members of the RepRap 3-D printer project's family tree, extrudes molten plastic from a nozzle. Arduino-based controllers drive stepper motors to position the nozzle in 3-D space above the build platform and control the extruder motor to deposit plastic where it's needed. ABS seems to be the most common material, although people have extruded shoestring licorice, granite-filled PVC, and cake frosting, albeit through different extruder heads.

The TOM's MK5 Extruder pulls its stock from a spool of 3-mm diameter filament and, unlike my Sherline, doesn't require manual intervention after starting to build a part. It uses a single "tool" to produce a complete part and doesn't require reclamping the part to access different surfaces.

Having recently finished the thermal measurements that went into my April 2011 column, I was surprised to see a pair of 5-Ω, 10-W resistors on the Extruder. My low-temperature oven used 50-W resistors at 65 °C, but the MK5 Extruder Head melts ABS filament at 200 to 230 °C (see Photo 1).

Even more surprising: those two 5-Ω resistors were connected in parallel across the 12-V power supply: each resistor dissipated 28.8 W!

That seemed like an unusual design decision, so I set out to measure and understand the thermal performance of the extruder head. In this column I'll explain what I measured, how I did it,

Photo 1—The MakerBot Plastruder MK5 head uses a pair of aluminum-cased resistors as heating elements. Solid 3-mm plastic filament enters the red PTFE tube, melts as it passes through the Thermal Core block, and extrudes through a 0.5-mm brass nozzle with green PTFE coating. I added the small brass tubes to hold thermocouple beads at key points on the extruder.

and what I found. You'll find the techniques useful, even in projects with less aggressive thermal requirements.

OVERPOWERING HEAT

Electronic components in most circuits generally dissipate relatively little power, so we tend to overlook thermal design problems until "something smells hot" inside the prototype. As you saw in my April 2011 column, the temperature of a component depends on both its power dissipation and the total thermal resistance between the component and the ambient air surrounding the case. Reducing that temperature requires either less power or lower thermal resistance.

In contrast, both my oven and the Extruder must maintain a specific temperature by varying the power dissipation to compensate for changes in heat loss. The plastic filament passing through the Extruder's Thermal Core (the stainless steel block holding the resistors and nozzle) carries off some energy, but the majority exits through the insulation surrounding the Thermal Core.

The TOM controls the Extruder temperature using feedback from a thermocouple clamped to the Thermal Core by the screw and washer shown in Photo 1. I attached thermocouples to other points on the core, providing data to calculate the heat transfer rates.

The small brass tubes epoxied to the resistors, the Thermal Core, and the heatsink on the Thermal Riser tube hold the thermocouple beads. I attached the tubes with J-B INDUSTRO WELD steel-filled epoxy, although its 260 °C maximum temperature rating was barely adequate for the job.

I assumed that the tubes would provide an isothermal environment for the beads and hold them in place, which worked out reasonably well, although one of the tubes did pop off the resistor when the cable shifted position. You should cement thermocouples in place using permanent high-temperature adhesive, but I didn't want to sacrifice all my thermocouples for this project.

In order to accurately calculate a thermal resistance, you must know the temperature difference and the actual power passing through the boundary. As you'll see, it's reasonably easy to measure the temperatures, but essentially impossible to account for the power. I'll explain some of the approximations as they crop up, but keep in mind that thermal measurements and calculations generally don't have as many significant digits as their electrical equivalents.

CALIBRATION RUN

Several of the digital multimeters in my collection came with thermocouple temperature probes, and I also have an old Fluke Model 52 dual-thermocouple meter. Combined with the TOM's thermocouple, I could measure six different temperatures at once. Unfortunately, the six "room temperature" readings differed by nearly 4 °C: some calibration was in order.

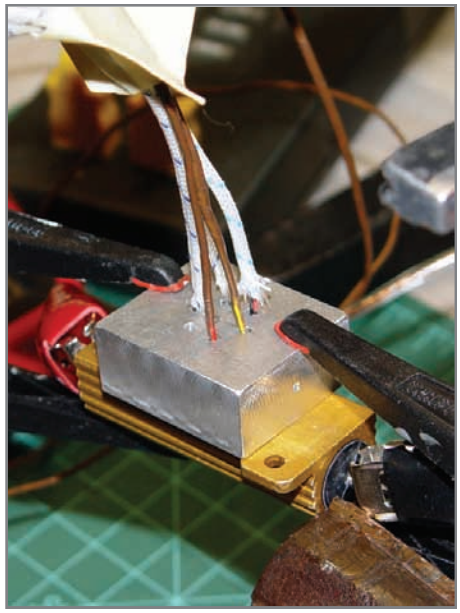


Photo 2—A heated aluminum block equalizes the thermocouple temperatures above the 6 Ω, 50-W power resistor serving as a heater.

Photo 2 shows a simple isothermal block consisting of six holes in an aluminum plate clamped to a 50-W power resistor. A bench power supply heated the resistor and the thermocouples provided the data for the graph in Figure 1. A quick glance shows that no two thermocouples agree at any temperature!

If I had to pick one meter, I'd use the Fluke: it was designed specifically to measure temperature by a company that builds instruments for a living. However, five of the six meters tracked reasonably well, so I used the average of their readings to estimate the true temperature. Later measurements showed that the TOM's thermocouple wasn't properly seated in its hole and I excluded its data from the average.

Figure 1 plots each thermocouple's reading against the computed average temperature for each measurement, but I really needed the six inverse functions that convert each meter's temperature reading to the average value. Applying the spreadsheet's `SLOPE()` and `INTERCEPT()` functions to the data produced the six sets of linear-regression coefficients in Table 1: the slope and intercept values of six lines that map readings to the average.

Three of the meters display their readings in Fahrenheit, which I manually converted to Celsius for Figure 1. The wildly different coefficients for the MPJA and Craftsman meters enable me to type their Fahrenheit readings into the spreadsheet and get Celsius results automatically.

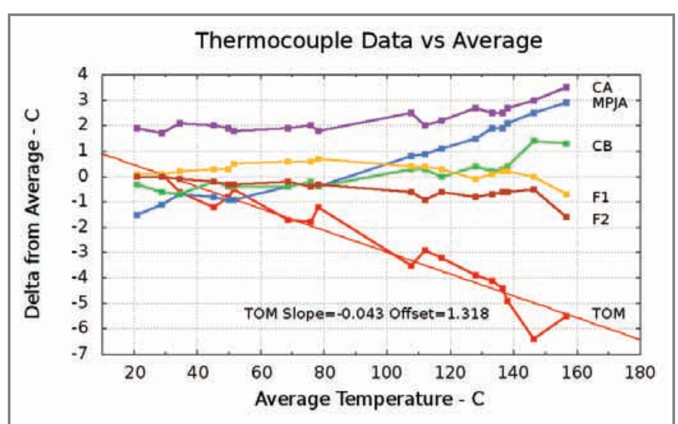


Figure 1—With all six thermocouples seated in an aluminum block, five show similar results. Later tests showed the MakerBot Thing-O-Matic thermocouple didn't make good contact with the block, so the average excludes its values.

	TOM	MPJA	Craftsman A	Craftsman B	Fluke T1	Fluke T2
M = slope	1.05	0.54	0.56	0.55	1.01	1.02
B = intercept	–1.61	–15.37	–19.42	–17	–0.74	–0.39

Table 1—These coefficients map raw temperature measurements from each meter to the best-fit line for the average value of all five meters. The MPJA and Craftsman meters display temperatures in Fahrenheit, so the coefficients convert directly to Celsius.

For example, suppose the Fluke's T1 channel reports the block is at 29.0 °C. The regression equation for that thermocouple works directly in Celsius: 28.6 °C = (1.011227 × 29.0 °C) – 0.742088. At that same temperature, the MPJA meter reads 82 °F. Its regression equation converts from Fahrenheit: 29.2 °C = (0.543354 × 82 °F) – 15/370290.

The two computed "average" values for that temperature differ by 0.6 °C because they represent two different linear approximations. The difference between the converted readings and the average remains under 1 °C across the temperature range shown in Figure 1, so the results are probably good to within ±1 °C. While that's not as good as a single meter bearing a NIST-traceable calibration certificate, my six meters now agree closely enough for my purposes. You can download the entire spreadsheet from the *Circuit Cellar* FTP site to see all of the gory details.

Photo 3 shows the extruder head mounted in the TOM, although it's barely visible in the upper center of the case amid the tangle of thermocouple cables and power wires. I connected a bench power supply to the resistor on the right-hand side of the Thermal Core using bulldog clips. Although the extruder head normally uses both resistors, I wanted to start with a simple, low-power test.

UNINSULATED HEATING

The first test heated the Thermal Core without any insulation to get a rough estimate of the thermal coefficients and power requirements. Table 2 lists the thermocouple locations and their adjusted temperature values at each power level, recorded every 10 minutes.

I doubled the applied power after the temperature stabilized within about 1 °C, which required about 40 minutes at each power level. The highlighted lines mark the more-or-less stable readings at each power level. If I had to do this more often, I'd definitely use a multi-channel temperature logger!

The test stopped when the bulldog clip attached to the power resistor shifted and dislodged the thermocouple. That's a good reason to use the proper cement for a high-temperature project, not to mention making secure terminations even in temporary setups. Fortunately, I planned to end the test at 8 W, because that's as much time as I was willing to devote to data recording.

Table 3 computes the temperature differences between selected locations on the Thermal Core. The differences remain reasonably stable at each power level even as the temperature rises on both sides of the interface.

The R-Edge column gives the difference between the resistor body and the upper edge of the Thermal Core directly adjacent to the resistor, using thermistors in the two brass tubes shown near the middle of Photo 1. Even at these relatively low power levels, the resistors run much hotter than the Thermal Core.

The next two columns indicate that the entire Thermal Core, a block of stainless steel, has about the same temperature throughout. Calibration errors may account for the Edge-Bot differences running slightly higher than the Top-Bot differences.

The Top-HS column shows that the Thermal Riser tube does a reasonable job of isolating the heatsink from the Core temperature. That's important, because solid plastic filament enters at the top of the Riser and melts as it exits from the bottom. Too much heat at the top of the Riser also causes problems with the filament drive motor's gear train.

The last two columns give the temperature differences between the resistor and Core edge to ambient air, assuming that the air remains at the temperature recorded at the start of the test before applying power.

Name	Meter	Location				
TOM	MK5 t-couple	Front of core				
T1	Fluke 52	Resistor				
T2	Fluke 52	Core edge adjacent to resistor				
CA	Craftsman A	Top of core				
CB	Craftsman B	Bottom of core				
MPJA	MPJA meter	Heatsink on thermal tube				
Power	TOM	T1	T2	CA	CB	MPJA
0	20.5	20.9	21.3	21.1	22.3	21
1	27.9	30.4	28.8	28.9	29	22.7
1	32.1	33.7	32.2	32.2	32.3	24.8
1	33.2	35.6	34.1	33.9	34	26.5
1	**34.2**	**36.3**	**34.8**	**34.4**	**34.5**	**27**
2	41.6	45.4	42.3	41.6	41.2	29.7
2	44.7	48.2	45.1	44.4	44.5	31.4
2	45.8	49.1	46	45.5	45	31.9
2	**46.8**	**50.5**	**47.5**	**46.6**	**46.1**	**33**
4	59.5	67.2	60.8	59.4	58.3	36.8
4	65.8	72.1	66.1	64.4	63.3	40.6
4	67.9	74.3	68.6	66.6	65.5	43.3
4	**67.9**	**75**	**69.2**	**67.7**	**66.1**	**44.4**
8	81.6	92	83.5	81.6	79.4	49.3
8	86.9	lost	88.9	86.6	83.8	52.5

Table 2—An uninsulated Thermal Core provides baseline information for subsequent tests. I recorded the temperatures every 10 minutes and increased the power when the temperature changed by less than 1 °C.

Power	R-Edge	Top-Bot	Edge-Bot	Top-HS	R-Amb	Edge-Amb
0	–0.4	–1.2	–1	0.1	0	0
1	1.7	–0.1	–0.2	6.2	9.5	7.4
1	1.5	–0.1	–0.1	7.4	12.8	10.9
1	1.4	–0.1	0.2	7.4	14.7	12.8
1	**1.4**	**–0.1**	**0.3**	**7.4**	**15.4**	**13.5**
2	3.1	0.5	1.1	11.9	24.5	20.9
2	3.1	–0.1	0.6	13.1	27.3	23.8
2	3.1	0.5	1	13.6	28.2	24.7
2	**3**	**0.5**	**1.4**	**13.6**	**29.6**	**26.2**
4	6.4	1.1	2.5	22.6	46.3	39.5
4	5.9	1.1	2.8	23.8	51.2	44.8
4	5.7	1.1	3	23.3	53.4	47.2
4	**5.8**	**1.6**	**3.1**	**23.3**	**54.1**	**47.8**
8	8.5	2.2	4.1	32.3	71.1	62.1
8	fail	2.8	5.1	34	fail	67.5

Table 3—These temperature differences show that the resistors run higher than the Thermal Core, the stainless steel Core is essentially an isothermal block, and the steel Thermal Riser Tube conducts a surprising amount of heat away from the core.

With all of those numbers in hand, Table 4 calculates the thermal resistances corresponding to the highlighted lines (shown in Table 3) marking the most stable temperatures. Although not all of the power from the resistor crosses the interface to the Thermal Core, I made two simplifying assumptions: it does and no power goes up the Riser.

Neither assumption is precisely correct, of course, because power escapes through the surrounding air, the Core's mounting screws, and the Riser. As a result, calculations using the total resistor power will underestimate the actual thermal resistance. Measuring the actual power at each interface poses a daunting problem that's generally solved with heavy-duty 3-D computational models, using the observed temperature differences as boundary conditions.

Given the limitations of my instrumentation, the calculations give, at best, an order-of-magnitude estimate of the actual resistances. It's reasonable to assume that most of the power crosses large-area metallic junctions and little power crosses a parallel air gap, but if you must have better accuracy, you must apply far more technology to the problem than I have available. However, simple measurements and calculations like these may give you enough of a handle to figure out what's going on in your project.

The R-Edge column gives the thermal coefficient between the resistor and the Core: 1.5 °C/W. Under these conditions, the resistor would run 43 °C hotter than the Core while dissipating 29 W from the 12-V supply. With the core at 225 °C, that puts the resistor at nearly 270 °C.

Power	R-Edge	Top-Bot	Edge-Bot	Top-HS	R-Amb	Edge-Amb
1 W	1.4	–0.1	0.3	7.4	15.4	13.5
2 W	1.5	0.2	0.7	6.8	14.8	13.1
4 W	1.5	0.4	0.8	5.8	13.5	12

Table 4—The resistor-to-Core interface has a relatively high thermal resistance due to the lack of thermal compound. The much larger Core-to-ambient resistance shows that air is a very good insulator.

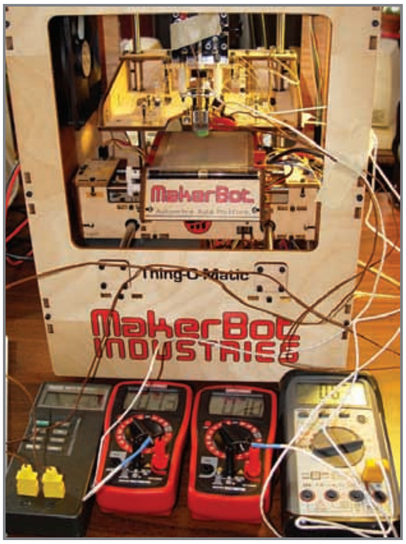


Photo 3—The instrumented MK5 Thermal Core disappears inside the MakerBot Thing-O-Matic 3-D printer. Four meters display five temperatures with a variety of units and resolutions, in addition to the Thermal Core's standard thermocouple.

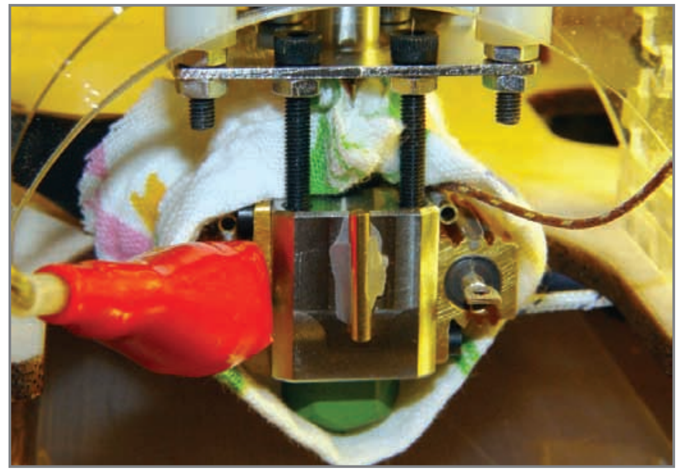


Photo 4—A cotton cloth wrapper simulates the ceramic fiber tape that insulates the Core at normal operating temperature. After taking this picture, I added a horizontal wrap to mummify the entire Core.

The R-Amb thermal coefficient from the resistor-to-ambient air runs about an order of magnitude higher: air is a pretty good insulator! That value declines as the temperature rises and convection cooling becomes more efficient: thermal effects remain stubbornly non-linear.

INSULATED HEATING

A layer of ceramic cloth normally surrounds the entire Thermal Core to reduce heat loss and, thus, the average power required to maintain the proper temperature. Insulation should also reduce the resistor-to-Core thermal coefficient, with a higher Core temperature and a lower differential for a given power input. I took another set of reduced-power measurements to discover the effect of an insulation blanket.

Due to the number of thermocouple and power leads, I threaded a fuzzy cotton cloth salvaged from the household rag bag around the tangle of wires (see Photo 4). After taking that photograph, I mummified the Core with another strip of cloth wrapped horizontally around it. Cotton isn't suitable for long-term use at high temperatures, but it's much more flexible and durable than ceramic cloth.

Those temperature measurements produced another table similar to Table 2, with the resistor temperature hitting 123 °C when the resistor dissipated only 6 W. I'll skip the details that boiled the numbers down to the thermal coefficients in Table 5.

As expected, the resistor-to-Core thermal coefficient has dropped. Assuming this coefficient applies at 225 °C, the resistor would run about 30 °C above the Core temperature at full power: perhaps 255 °C. That's better, but still far too hot.

Both resistance to ambient air increased, showing that the insulation is having an effect. The increase isn't as large as you might expect, because air is pretty good all by itself. In fact, the solid part of most insulating materials just traps the surrounding air.

Although Photo 1 shows a thin line of blue thermal compound squeezed out around the resistor, the numbers so far represent a dry resistor, because that's what the TOM assembly directions specify. Another test run showed that the layer of thermal compound reduced the resistor-to-Core thermal resistance from 1.0 to 0.8 for an insulation-wrapped Core, a significant improvement. The temperature differences, however, became small enough to bump into the limits of my instrumentation.

Common thermal compounds used between PCs' CPUs and their heatsinks have upper temperature limits around 150 °C. I used an industrial compound rated to 200 °C, still lower than the expected temperatures, and it shows signs of distress after a few weeks of full-power usage. This is a surprisingly difficult problem to solve in an inexpensive, small-scale, DIY manner.

FULL-POWER HEATING

With all of those measurements indicating a thermal resistance around 0.8 °C/W, I was ready for high-power tests at normal plastic-extruding temperatures. My bench power supply has a 3-A maximum current limit, so I put the 5-Ω resistors in series and applied twice the voltage to get the

Power	R °C	R-Edge-	R-Amb	Edge-Amb
1 W	39.1	1	19.7	18.4
2 W	56.3	1	18.5	17.3
4 W	90.7	1	17.8	16.8
6 W	122.7	1	17.2	16.1

Table 5—Wrapping the Thermal Core with cotton fabric increased the peak resistor temperature to 122 °C at only 6 W, but reduced the resistor-to-Core thermal resistance. There is no thermal compound between the resistor and Core.

Power	R °C	R-Edge	R-Amb
6 W	90	1.2	22.4
14 W	160.6	1	19.8
20 W	208	0.8	18.6
25 W	222.2	0.8	16

Table 6—Thermal compound under the resistor reduces that interface's thermal resistance. The Core reaches an extrusion temperature at about half the total power, so the MakerBot Thing-O-Matic controller applies PWM to maintain the setpoint temperature.

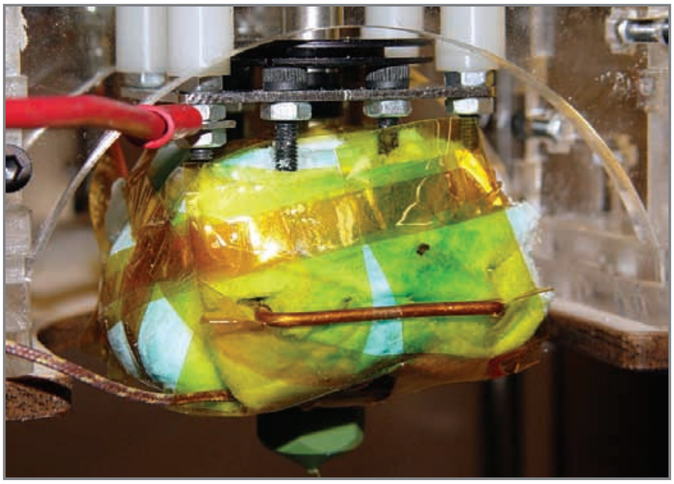


Photo 5—Ceramic fiber insulation reduces heat loss from the Thermal Core during normal operation. The round black heatsink near the top of the picture helps cool the Thermal Riser Tube leading to the filament drive wheel, but better insulation makes that heat loss a larger fraction of the total.

same power at half the current.

Pop quiz: Why not increase the voltage by 1.414 =√2?

Table 6 summarizes the results of a test with the core wrapped in the original ceramic cloth insulation. The Core reaches its normal operating temperature around 220 °C at about half of the normal power dissipation, which coincides with observations that the TOM's temperature controller applies less than a 50% duty cycle to maintain that temperature.

I computed the thermal resistances in Table 6 by dividing the resistor-to-Core temperature difference by half of the total power. In any event, the numbers remain in rough agreement with my previous findings.

Assuming the same 0.8 °C/W thermal resistance applies at full power, which now seems entirely reasonable, each resistor runs about 23 °C above the Core temperature: about 250 °C.

I wrapped the Thermal Core in somewhat thicker insulation, using ceramic fiber batting from an oil burner combustion chamber relining kit (see Photo 5). Although this slightly reduced the power requirement, the losses through the Riser Tube and the nozzle are already a large fraction of the total, so that more insulation isn't particularly effective.

CONSEQUENCES

As I explained in my April 2011 column, the resistor datasheets require reduced power for operation above 25 °C: the maximum-allowed power decreases by 0.4 W/ °C, down to 10% of rated power at 250 °C. As a result, the 10-W resistors should dissipate no more than 1 W at 250 °C. They're overdriven by a factor of nearly 30 at 28.8 W.

I initially thought this was a suicidally bad decision, but it's actually a reasonable extrapolation from previously successful DIY extruders.

Early extruders used Nichrome wire wrapped around a metal body tube as the heating element. Given the difficulty of insulating extremely hot wire, the most common failure mode seems to have been failed open heaters due to burned-out hotspots. Less common, but equally devastating, were shorts to the metal body that vaporized sections of the heater winding.

A later development embedded cylindrical ceramic power resistors in holes drilled into a metal block similar to the present Thermal Core. The resistors had a very high surface temperature rating, but the assembly technique required precise fitting and didn't have good thermal contact between the resistor and Core.

Aluminum-case power resistors feature easy assembly, excellent heat transfer, and a reasonable per-unit pricing. The folks at MakerBot tell me that their initial testing showed uniformly positive results: the resistors worked exactly as intended.

Unfortunately, a 15-dB power overload is not to be trifled with. A small, but growing, number of resistors failed very quickly and produced a puzzling symptom: the extruders continued to work, but took a long time to heat up.

You'll recall that the standard configuration connects the resistors in parallel: losing one resistor just reduces the available power by a factor of two. My experiments showed that the extruder could reach and maintain operating temperature at less than half the rated power. Diagnosing the problem turned out to be quite simple: if the extruder heater resistance measured 5 Ω at the end of the cable, then one of the resistors had failed.

The resistors fail open, but in a few cases the resistive wire winding burned through its insulating liner and connected the +12-V supply directly to the aluminum housing. Unfortunately, the resistor mounting screws provide a good electrical connection to the Thermal Core and the thermocouple bead clamped to it.

The other end of the thermocouple cable terminates in a MAX6675 thermocouple-to-SPI digital interface chip that does not take kindly to having +12-V applied to its analog inputs. A simple resistor failure can easily destroy the Extruder Controller board.

Ouch!

Cartridge heaters, the usual industrial heating solution, embed resistive wire elements in solid ceramic insulation within a stainless-steel shell. The assembly can operate at white heat, although the practical upper limit seems to be 700 °C, far higher than needed in this application.

MakerBot had considered cartridge heaters early in the MK5 design, but rejected them based on per-unit cost and (at the time) the need for more than the +12 V available from a standard PC power supply. A year later, I found some 12-V cartridge heaters that should be a suitable replacement.

Sometimes the correct business decision isn't what the engineers would want: "This works well enough, let's ship it!" In this case, those resistors worked *almost* well enough and the fix won't be too disruptive.

CONTACT RELEASE

MakerBot Industries graciously supplied me with several cartridge heaters for further investigation. The next

Thermal Core design will probably use a single cartridge heater: the price may be higher, but the improved reliability will be priceless! ◼

Ed Nisley is an EE and author in Poughkeepsie, NY. Contact him at ed.nisley@ieee.org with "Circuit Cellar" in the subject to avoid spam filters.

PROJECT FILES

To download a spreadsheet with experimental data, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2011/251.

RESOURCES

Botacon Convention, www.botacon.com.

HydraRaptor, "Yet Another Quick Heater Hack," 2009, http://hydraraptor.blogspot.com/2009/01/yet-another-quick-heater-hack.html.

E. Nisley, "Thermal Performance," *Circuit Cellar* 249, 2011.

RepRap.org, http://reprap.org/wiki/Main_Page.

SOURCES

Arduino-based microcontrollers
Arduino | www.arduino.cc

RepRap printers
Bits From Bytes | www.bitsfrombytes.com

Fluke Model 52 dual-thermocouple meter
Cole Parmer (supplier) | www.coleparmer.com

Cartridge heaters
High-Temp Industries, LLC | www.high-temp industries.com

INDUSTRO WELD Epoxy
J-B Weld Company | http://jbweld.net/products/industro.php

Thing-O-Matic and Plastruder MK5
MakerBot Industries | http://wiki.makerbot.com/thingomatic and http://wiki.makerbot.com/plastruder-mk5

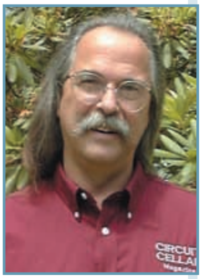
High-temperature cements
Omega Engineering, Inc. | www.omega.com/pptst/OB_BOND_CHEM_SET.html

CNC milling machine
Sherline Products Inc., | www.sherline.com

Resistors
Vishay Dale | www.vishay.com/company/brands/dale

FROM THE BENCH

by Jeff Bachiochi (USA)



Vehicle Diagnostics (Part 1)

The CAN Protocol and OBD-II

There's a lot more moving under the hood of a modern automobile than gears and liquids. Today's vehicles also move essential system instructions and on-board diagnostic data between electronics and dedicated computer applications. This article details the relationship between automotive control systems (e.g., engine and transmission) and OBD-II data. You also learn how to tap into a vehicle's LAN.

I know some readers live in areas where 50″ of snow is the norm, but Connecticut rarely sees anywhere near as much as it did last January. I quickly tired of lifting snow shovels full of the white stuff over my head. At times I felt like I was in a labyrinth, or, better yet, wandering through a mouse maze looking for some spicy pepper jack. I felt foolish spending time cleaning out my ice-filled gutters only to find my neighbors going through the same ritual. As my pastor said one Sunday, "Will whoever is praying for snow please stop it?"

I took the chill off by snuggling into my favorite chair with a mug of hot chocolate and a good book or an engineering magazine, like *Circuit Cellar*. Do you notice that magazines tend to get delivered in batches? I'll spend the night going through them, reading the articles, and ripping out ads that look interesting. I like to hold print issues as their ads tend to rip out more easily than the digital ones. While my Droid has a Kindle app, it doesn't feel the same as when I hold the pages of a real book or magazine.

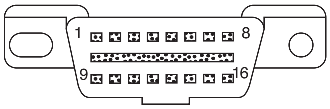
To me this is like going from a standard transmission to an automatic. For those of you who have never driven a standard, you don't know what you're missing. You drive a standard. An automatic drives you. Both will get you where you want to go, but there's a kind of Zen thing you experience with a standard that you can't get with an automatic. I'm certainly not against progress. I'm all for the move toward electrics. It's just a bummer that this moves us further away from that feeling of exhilaration—or is that acceleration?

Truth be told, I've become a wimpy driver. My wife's Prius has replaced my clutch popping desires with an interest in conservation. Just how many miles can be eked out of a gallon of gasoline? I find the performance feedback displayed to be a strong motivator. So much so that I'd like to have that available in my vehicle. The solution is in front of me every time I drive: the on-board diagnostics (OBD).

TREE HUGGING

Allow me to quote Wikipedia: "Smog is a type of air pollution; the word 'smog' is a portmanteau of smoke and fog. Modern smog is a type of air pollution derived from vehicular emission from internal combustion engines and industrial fumes that react in the atmosphere with sunlight to form secondary pollutants that also combine with the primary emissions to form photochemical smog. Smog is also caused by large amounts of coal burning in an area caused by a mixture of smoke and sulfur dioxide."

With the vast number of automobiles in California, it's no wonder people became concerned with what was coming out of their tailpipes. Progressive Californians became the first to demand attention from Detroit. Slowly, Detroit began to curb the production of unburned hydrocarbons and carbon monoxide by adding



Pin #	Description	Pin #	Description
9		1	
10	B-	2	B+
11		3	
12		4	Chassis GND
13		5	Signal GND
14	CAN-L	6	CAN-H
15	L	7	K
16	Battery	8	

Figure 1—While the OBD-II connector contains the protocols for the three major automotive manufacturers, any automobile manufactured after January 1, 2008 must use the CAN protocol.

smarts to the fuel delivery and ignition systems on all of their internal combustion engines. Along with these improvements came a mandate that each automobile must prove its right to run the roads by passing emission testing. Do you remember visiting the testing facility where they would put each vehicle on a dynamometer and check the extent of HC, CO, and NOX? Failure usually meant spending up to $1,000 getting your automobile to comply before it would be eligible for an exemption—at least until it was time for the next test.

You might remember when the "alternator" light was the only indication of imminent failure. Since these new smart emission systems required a computer, it wasn't long until manufacturers began using the computer to handle other tasks. The alternator light was replaced by a new idiot light, the "check engine" light. If you knew the secret sequence, the ignition (key) switch would turn the check engine light into "Gotham City's bat signal" that would actually blink out a number code. Only those with the "secret decoder ring" could translate this into a real error message.

Naturally, as each manufacturer developed a system specific for their line of automobiles, there were no standards for this OBD ability. The Society of Automotive Engineers (SAE) attempted to pull it all together by defining OBD-II, which uses a 16-pin standard connector and a standard set of diagnostic codes and terminology (see Figure 1). You'll note that this standard connector has connections for all of the major systems in use! Some standard, huh? Actually, that's not fair at all since you can't change industry overnight. The OBD-II, as it stood, was required on all automobiles built after January 1, 1996.

In general, Chrysler, European, and many Asian imports use what is known as the ISO 9141 standard (OBD-II pins 7 and 15). General Motors uses the SAE J1850 variable pulse-width (VPW) modulation (OBD-II pins 2 and 10). Ford uses another version of SAE J1850, pulse-width modulation (PWM) (also on OBD-II pins 2 and 10). For the most part, all of these systems communicate at a rate of approximately 10 Kb/s.

In keeping with the need for increased standardization and bandwidth, another protocol is included in the OBD-II definition. The ISO 15765 control area network (CAN), which has been predefined to use pins 6 and 14. Automobiles built after January 1, 2008 are required to use this system.

You might think that by now we all are pretty much driving automobiles that use the CAN interface, but according to the U.S. Department of Transportation, the average lifespan of a vehicle is 12 years, or about 128,500 miles. So, the average car may not use CAN until after 2020.

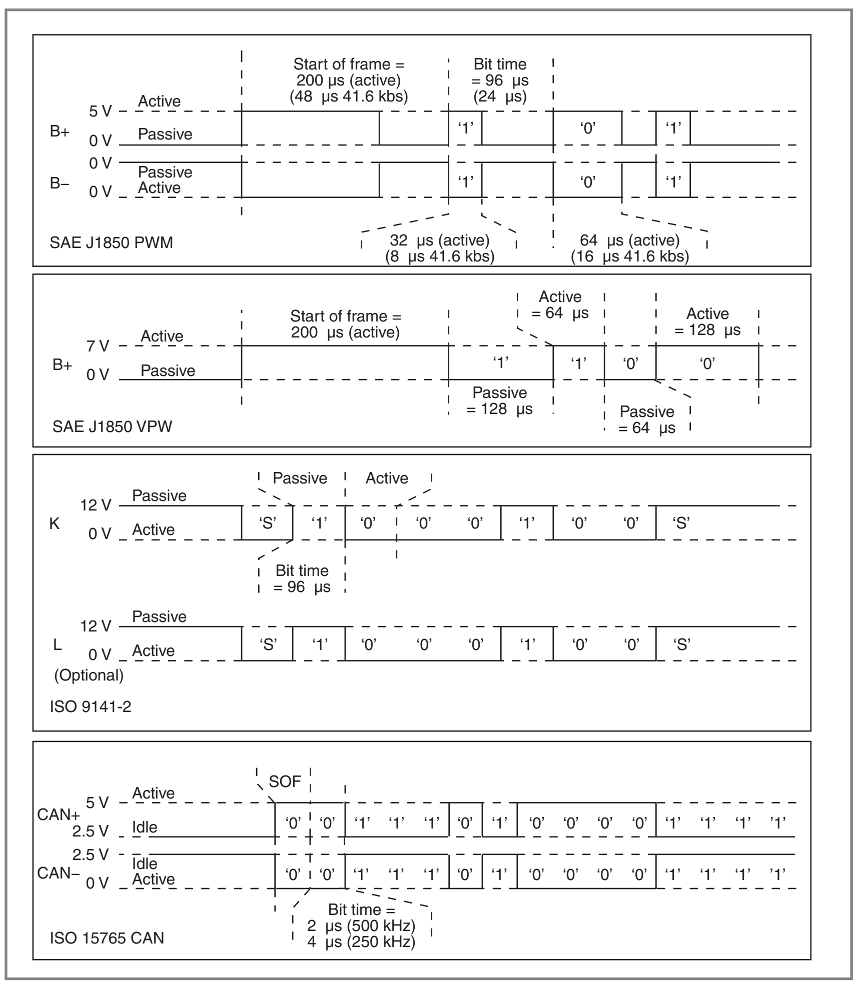


Figure 2—The basic timing parameters for each of the protocols presented on the OBD-II connector

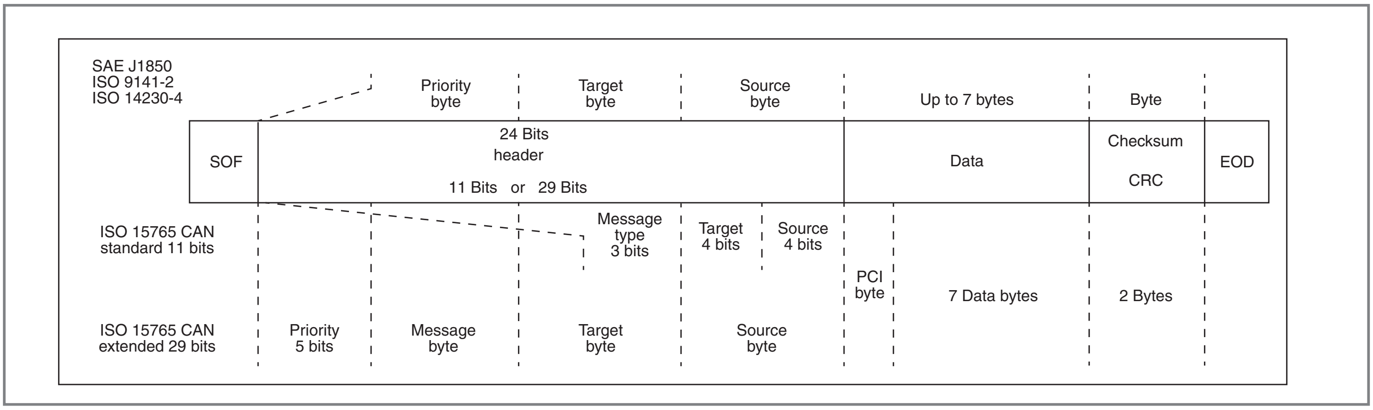


Figure 3—The packet format is slightly different for each protocol, but the data uses standard diagnostic codes and terminology. This makes the transition to CAN easier.

U.S. AUTOMAKERS

Thanks to the government's offer to loan monies to the U.S. automotive industry, we still have viable alternatives to the import market. But this line is becoming fuzzy as many foreign companies have assembly lines here in the U.S. I think many will agree that it is U.S. jobs that are number one, not necessarily that an employer is a U.S. company. The automotive industry will remain important to our economy, even if rocket packs were to become a reality.

Let's take a look at the four major protocols of OBD-II that are in use today. Refer to Figure 2 throughout the following details, as I'll be discussing each of the big three, as well as the CAN protocol, which is already under transition.

I'd like to start with the ISO 9141-2 protocol as it is, perhaps, the easiest to understand for those familiar with serial communications. It uses an asynchronous serial format running at 10,400 bps. While RS-232 normally uses a level-shifting IC (such as the MAX232) for a 12 V/–12 V inverted signal swing, 9141-2 is more like 12 V TTL (if you will), with 12 V being the passive state and 0 V the active state. Each byte begins with an active start bit, followed by 8 data bits, no parity bits, and a single passive-stop bit. Bidirectional communication takes place on the "K" line (pin 7) of the OBD-II connector. Optionally, the "L" line (pin 15) may be used as a unidirectional "wake-up" signal to the vehicles' engine control unit (ECU). Message data is limited to 12 bytes per command.

An alternate protocol to this is the ISO 14230 KWP2000. Besides the initialization (to determine the protocol) the format is identical to ISO 9141-2, except that messages may contain up to 255 bytes.

The next two protocols are both SAE 9150, but the formats are quite different from the previously mentioned 9141-2 and from each other. These are designated SAE 9150 VPW and SAE 9150 PWM. Variable pulse width (VPW) modulation is based on 2-bit symbols, a 64-µs width pulse and a 128-µs width pulse. The VPW signal varies between 0 V (passive) and 7 V (active). Each packet begins with a 200-µs (active) start-of-frame (SOF) pulse followed by data bits. There is a change in polarity for each bit symbol. Since a transmission begins with an active SOF, the first bit is always passive. A 64-µs passive pulse is a "0" bit, while a 128-bit passive pulse is a "1" bit. When the first symbol's pulse has ended, the polarity reverts to the active state. The bit symbols are reversed in the active state with a 64-µs active pulse equal to a "1" and a 128-bit active pulse equal to a "0." Note that the overall data rate will vary depending on the data symbols. This protocol used the B+ line (pin 2) on the OBD-II connector.

The SAE 1850 PWM protocol uses a standard bit time of 96 µs, so the data rate is fixed (10,400) and has about the same average throughput as VPW. This protocol also uses SOF with a 128-µs active/64-µs passive pulse. Each 96-µs data bit always begins in the active state. A data bit is considered a "0" if it remains in the active state for 64 µs and a "1" if it remains in the active state for 32 µs. The PWM protocol also has a 4× mode where the throughput is increased to 41,600 bps. In this mode all timing values decreased by a factor of four.

In an attempt to be more immune to noise, the output signal of the SAE 1850 PWM protocol is transmitted differentially using two lines, B+ (pin 2) and B– (pin 10). The B+ line uses an active state of 5 V, while the B– line uses an active state of 0 V. The passive state of B+ is pulled low and B– is pulled high.

Protocol	Functional target address
SAE J1850 PWM	0x6A
SAE J1850 VPW	
ISO 9141-2	
ISO 14230-4	0x33
ISO 15765-4 (29-bit)	
ISO 15765-4 (11-bit)	0x7DF

Table 1—Each protocol uses a specific functional target address that all of its ECUs must respond to.

Except for the 4× mode of the SAE 1850 PWM protocol, all of these have similar throughputs. In case you haven't noticed, working on your car's engine has become increasingly more difficult. I can remember being able to see the ground when I opened the hood. Today every nook and cranny is filled with components of our smarter cars. The amount of

Protocol	Header			
	Size	Priority	Target address	Source address
SAE J1850 PWM	(24 bits)	0x61	0x6A	0xF1
SAE J1850 VPW	(24 bits)	0x68	0x6A	0xF1
ISO 9141-2	(24 bits)	0x68	0x6A	0xF1
ISO 14230-4	(24 bits)	0xC2	0x33	0xF1
ISO 15765-4	(29 bits)	0x18DB	0x33	0xF1
ISO 15765-4	(11 bits)		0x7DF	

Table 2—The header for each protocol provides the information necessary to route the data that follows into the communication stream.

data flowing under the hood has naturally increased. Multiple computers are now necessary, each handling specific tasks. It wasn't difficult to foresee the original throughputs as limiting. In their infinite wisdom the SAE added the CAN protocol to the OBD-II specifications. At the time, this was not used by any American automotive manufacturer. (Bosch began using the CAN bus in the early 90s.)

ISO 15765 CAN

The CAN protocol specified for use in OBD-II is capable of using either 250 or 500 kbps. (You may read more about CAN and the CANopen protocol in my *Circuit Cellar* columns from June and July, 2006.) Two formats are allowed in the CAN protocol, an 11-bit addressing mode and a 29-bit extended addressing mode. If you include the dual data rates, that's four possible formats for the CAN protocol.

There is no clock transmission associated with the data as in high-speed synchronous transmissions; therefore the CAN hardware employs a mechanism for keeping the receiver "in-sync" with the transmitter. An integrated phase lock loop (PLL) syncs on-state transitions. To ensure transitions, bit stuffing is used. Bit stuffing is the procedure of adding an opposite state bit whenever the data state remains the same for five consecutive bit times. The receiver automatically tosses out the next bit whenever it receives five consecutive bits of the same state.

Similar to the SAE 1850 PWM protocol, the output signal of the ISO 15765 CAN protocol is transmitted differentially using two lines, CAN-H (pin 6) and CAN-L (pin 14). The CAN-H line uses an active state of 5 V, while the CAN-L line uses an active state of 0 V. The passive state of both the CAN-H and CAN-L lines is undriven, about 2.5 V. The idle state is considered a logic "1." Therefore a "0" bit of either 4 µs (250 kbps) or 2 µs (500 kbps) indicates the SOF. Data bits follow using the same bit timing.

DATA FORMAT

While different methods of transmitting information exist, the data

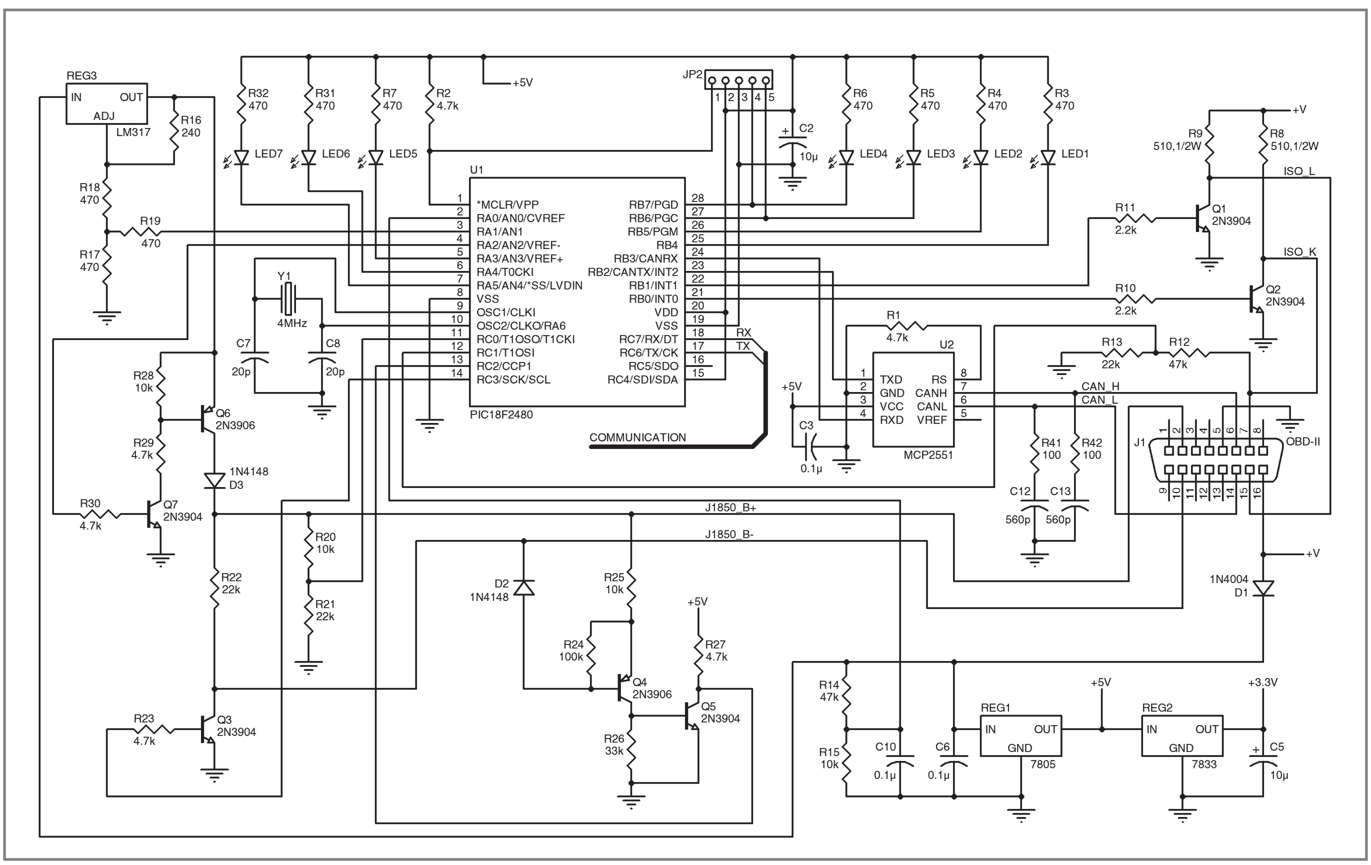


Figure 4—This schematic contains circuitry to connect to any protocol on the ODB-II connector. The Microchip Technology PIC18F2480 microcontroller used in this project can be directly replaced with an Elm Electronics ELM327 interpreter.

Mode	Operation
0x01	Show current data
0x02	Show freeze frame data
0x03	Show stored diagnostic trouble codes (DTCs)
0x04	Clear DTCs
0x05	Oxygen sensor monitoring, test results (non CAN)
0x06	Component monitoring, test results for all (including oxygen for CAN)
0x07	Show pending DTCs
0x08	Control operation of on-board components
0x09	Request vehicle information
0x0A	Permanent DTCs

Table 3—The first data byte usually indicates the mode and is used to request certain types of information.

remains consistent throughout the original protocols. One important point to note is that of bus arbitration. The idle or passive state of any bus can always be overridden by an active state by any node. This creates a priority where "0" (active data) wins out over "1" (passive data). Nodes are required to listen to the bus as they send data. If any node detects an active state while it is attempting to send a passive state, it must quit sending and wait for the bus to become idle. Therefore, smaller values have priority over larger values. In this way an important message, such as engine timing, can get through while a request for water temperature might have to wait.

Note that all buses are considered to be in their idle state if there has been no activity for some amount of time, called end-of-data (EOD). Each packet begins with a SOF pulse (see top of Figure 3). Header bytes are used to direct the data. The priority byte enables arbitration of the message. The target and source bytes help direct the request to the proper node and indicate where the response should be directed. The data field can include up to 7 bytes of data (more on this later). A running checksum is used to give the packet some level of quality (ensuring that the data hasn't been corrupted).

The CAN format is pretty much the same (see the bottom of Figure 3). The header can use either the standard 11-bit or the extended 29-bit format. Eight data bytes are always passed with the first byte PCI, used as formatting information on the following data bytes. Corruption control is strengthened with a 15-bit CRC value (see last month's column for more on CRCs).

OBD-II DATA

Most requests will consist of a mode byte and a parameter ID (PID) byte. But how do we know who's out there for targets? The OBD-II requires one functional address that must be responded to. This functional address can be different depending on the protocol. For the most part, these are target addresses (see Table 1). The specifications allow for up to eight ECUs, although they may have to do with systems other than the engine, such as the transmission control module (TCM) or the anti-lock braking system (ABS). All ECUs must respond to a query to the functional address (this is like a public shout). You'll notice that if there is more than one ECU, the responses arrive according to the priority of each ECU. I may be getting ahead of myself a bit here, but when they respond they will include their physical address. This physical address can be used in the future to communicate with each one-on-one (more like a personal communication).

So, to send out a broadcast to the functional address we start by making sure the header of the message contains the right information. As shown back in Figure 3, this header might be 24 bits, 19 bits, or 29 bits in length. Refer to Table 2 for the header information necessary to get a data packet ready to send to the functional address, which will give us a response from any ECU on the bus.

Now, what can we ask for? Presently, there are 10 modes defined in the standard that can be requested (these are listed in Table 3). Not all modes are supported by all manufacturers. Our initial request will use mode 1, "show current data." What data can we see? We can see any parameters defined by the second byte in the data portion of the message packet, the PID byte. For instance, PID 0x00 must be supported as it requests which other PIDs (PID 0x01 to 0x20) are supported. It returns 4 bytes (32 bits) one bit for each PID. When any bit position = "1" that associated PID is supported. Note: If the first bit, the MSBit (for PID 0x20), is = "1" then PID 0x20 will indicate which other PIDs (PID 0x21 to 0x40) are supported, and so on. See the reference at the end of this column for PIDs too numerous to list here.

PHYSICAL INTERFACING

We're now ready to communicate with OBD-II. Well, except for one small problem, we need some support circuitry to enable our packets to be sent into the OBD-II connector. The circuitry shown in Figure 4 supports all of the protocols used in the OBD-II interface. You can choose to support all or just the one needed for your vehicle. I used the same basic circuit as referenced in Elm Electronics's application note, "AN02 – ELM327 Circuit Construction." Elm Electronics is a Canadian company that produces a preprogrammed microcontroller that (along with this circuitry) creates an OBD-II-to-serial interpreter. Many diagnostic tools are based on using this interface chip (or at least the AT-style commands it offers). By using this basic circuit you can create your own interface that can be driven by either the ELM327 or your own code using a Microchip Technology PIC18F2480 microcontroller. Once you understand how this all works, your application may not need the "AT-style" user interface, as the microcontroller will enable you to handle those interactions necessary for tapping into the data required for your application without external intervention. What do you want to know?

Next month, we'll look at a little code to make this circuitry sing, or at least begin to unlock the treasures under the hood. Between now and then you also might want to look at www.scantool.net. It is an American company that not only offers OBD-II scan tools and components, including a functional equivalent of the ELM interpreter and a module complete with interfacing circuitry, it also offers a simulator that enables you to bench test your OBD-II circuitry, sans car. Just when you thought working on your car was a thing of the past, the tools necessary to get answers have become more available to the general population. You can harvest this growing bounty of information by tapping into your vehicle's LAN. For now, this doorway is OBD-II, and although there is already work on the next generation, with FCC and "invasion of privacy issues," it isn't going to happen any time soon.

Jeff Bachiochi (pronounced BAH-key-AH-key) has been writing for Circuit Cellar *since 1988. His background includes product design and manufacturing. You can reach him at jeff.bachiochi@imaginethat now.com or at www.imaginethatnow.com.*

REFERENCES

J. Bachiochi, "Can You Speak CAN? (Part 1)," *Circuit Cellar* 191, 2006.

———, "Can You Speak CAN? (Part 2)," *Circuit Cellar* 192, 2006.

Elm Electronics, Inc., "ELM327 OBD to RS232 Interpreter," ELM327DSH, 2010, www.elmelectronics.com/DSheets/ELM327DS.pdf.

———, "AN02 – ELM327 Circuit Construction," AN02 rev A, www.elmelectronics.com/AppNotes/AppNote02.pdf.

ScanTool.net, OBD-II Scantools, www.scantool.net/scantool/downloads/101/ecusim_2000-ug.pdf

SOURCES

ELM327 Interpreter
Elm Electronics, Inc. | www.elmelectronics.com

PIC18F2480 Microcontrollers with ECAN technology
Microchip Technology, Inc. | www.microchip.com

by Robert Lacoste (France)

Stay Cool!

A Heat Management Primer

Thermal design problems can lead to unpleasant results. Testing your system and using thermal simulation software early in the process can help eliminate problems before it's too late. This article provides some basic rules about thermal management to help you determine whether or not your design is thermally correct.

Welcome back to The Darker Side. Chip designers do incredible things. I just read that the latest Intel Itanium baby, code-named Poulson, integrates no less than 3.1 billion transistors. That's probably reasonable, as it includes eight 64-bit cores and 54 MB of cache memory on chip. Well, Intel also proudly announced that this chip is less power-hungry than its predecessors, requiring "only 170 W." To me, this emphasizes that Intel's thermal engineers are at least as clever as their silicon engineers! Let's make a comparison: Your electric home heater is roughly 60 × 60 cm, whereas the chip die is made of maybe one square cm of silicon. If your heater had the same power density as this Itanium chip it would dissipate no less than 60 × 60 × 170 W, which is 612 kW!

Even if your designs are far from these extreme power densities, it is not unusual to have some watts to dissipate on a PCB. Motor drivers, RF, or audio amplifiers come to mind, but high-power dissipation is also usual on high-speed digital boards. For example, my company's last FPGA project eats up close to 10 A on the 1.8-V power rail.

This month I will devote my column to thermal design, a topic that Ed Nisley recently covered in his April 2011 article ("Thermal Performance," *Circuit Cellar*, 249). Let's be honest. I'm not a specialist in thermal problems, but I did have some unpleasant surprises while working on a couple of designs (which meant that some chips exploded with heavy smoke). Niels Bohr once said that an expert is a person who has made all the mistakes that can be made in a very narrow field. I'm far from a thermal expert, but I will try to limit your personal exposure by providing you with some basic rules.

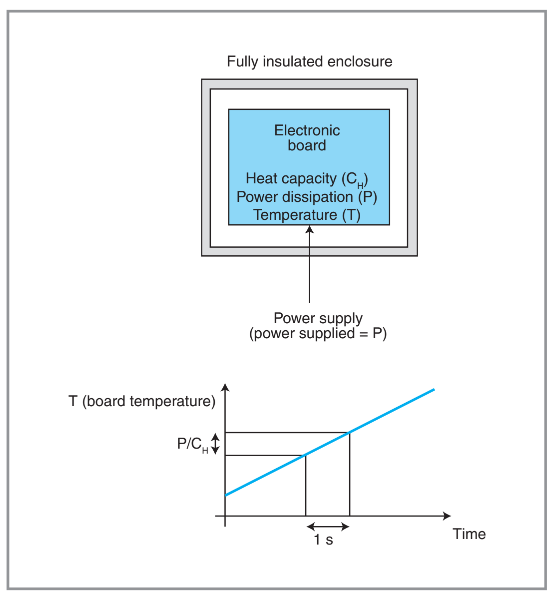


Figure 1—Thermal engineering is all about thermal transfers. If you put an electronic board inside a hypothetical perfectly thermally insulated enclosure, its temperature will rise up to a meltdown with a rate proportional to the dissipated power and inversely proportional to the heat capacity of the system.

ENERGY DISPERSAL

Let's start with the basics: Thermal management is all about energy dissipation. This is, in fact, a direct consequence of the first law of thermodynamics, which states that energy can't be destroyed (or created). Imagine that you have an electronic board that is continuously dissipating a small power $P_{THERMAL}$ equal to 1 W. Assume for the moment that you have a perfectly insulated enclosure. Put the board inside this enclosure and switch the power on. What will happen?

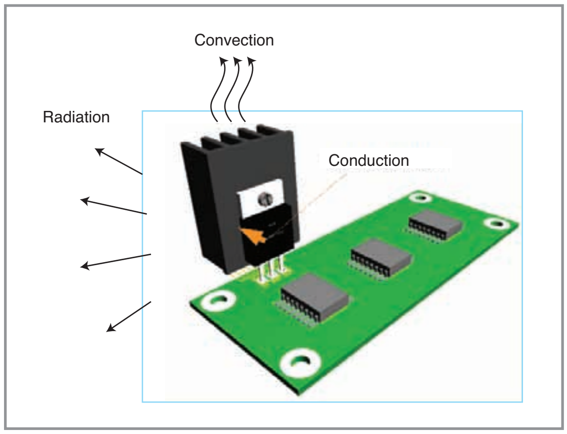


Figure 2—Heat transfers can be conduction transfers, convection transfers (either natural or forced), or radiated transfers.

As there is no thermal exchange between the inside of the enclosure and the rest of the universe, the board temperature will linearly increase over time (see Figure 1). Every second there will be an energy $E_{THERMAL} = P_{THERMAL} \times$ time = 1 W × 1 s = 1 J dissipated inside the enclosure by your electronic board. This energy can't go anywhere, so every second the board temperature will increase by $dT = E_{THERMAL}/C_H = 1/\ C_H$ degree. In this formula, C_H is a constant called the heat capacity of the board. If the board is homogeneous, say made of a full sheet of copper, then its heat capacity will simply be proportional to the mass of the board. More specifically, it will be equal to the mass of the board multiplied by the so-called specific heat of the copper. You will find this value in any physics book; for copper it is 0.385 J per gram per 1°C. To be exact, we should also add the heat capacity of the air around the board, but it is small enough to neglect.

So, the math is straightforward. For example, imagine that you fix a heavy 1 kg copper heatsink to the 170-W Itanium processor I mentioned before and put the assembly in a perfectly thermal-insulated enclosure (which would be a bad idea). Every minute the assembly temperature will increase by 26.5°C (i.e., (170 W × 60 s) / (1,000 g × 0.385 J/g°C). After 15 minutes the processor and heatsink will exceed 400°C, and your expensive processor will definitively melt down.

THERMAL TRANSFERS

Fortunately, perfectly insulated enclosures don't exist. With any actual packaging a part of the energy dissipated inside will find its way to the outside, more or less efficiently depending on the design. This thermal transfer will limit the temperature rise of the electronics up to an equilibrium state, where the thermal losses of the environment are equal to the energy dissipated by the electronic components.

How is thermal energy dissipated to the outside world? There are three kinds of heat transfers (see Figure 2). The first, and usually the most efficient one, is conduction. Joseph Fourier (the inventor of the Fourier transform) stated in 1822 that the thermal flux between two closely spaced points is proportional to the temperature gradient and opposite in sign. Let's illustrate that with the example of a thin sheet of homogenous material (see Figure 3). Fourier's law then says that the heat transfer (in W) between the two sides of the sheet will be proportional to the temperature difference of the two sides, proportional to the area A of the sheet (the larger the better), inversely proportional to its thickness t, and flowing from hot to cold sides. The multiplicative coefficient is called the thermal conductivity of the material. Its value is, for example, 383 W/m.K for the copper which is one of the best thermal conductors. The formula is the following:

$$Q_{1\ to\ 2}(W) = k \times (T2 - T1) \times \frac{A}{t}$$

This seems reasonable, doesn't it? Let's go back to electronics. Imagine that you have a power transistor on a very large heatsink, assuming for the moment to stay at a constant 25°C. What will the temperature of the transistor die be if the transistor dissipates a given power $P_{THERMAL}$? When you switch on the system the transistor is obviously at 25°C, too. It will then rise, depending on its own heat capacity, then thermal energy will flow to the heatsink. At a given time, an equilibrium will be reached, which means that the heat transfer between the transistor and the heatsink is exactly equal to the power P dissipated by the transistor. In a nutshell, the respective heat capacities of the parts define the dynamic thermal behavior of the design, but their respective heat transfers define the final temperatures. Then what is the static temperature difference between the transistor die and the heatsink? Assume

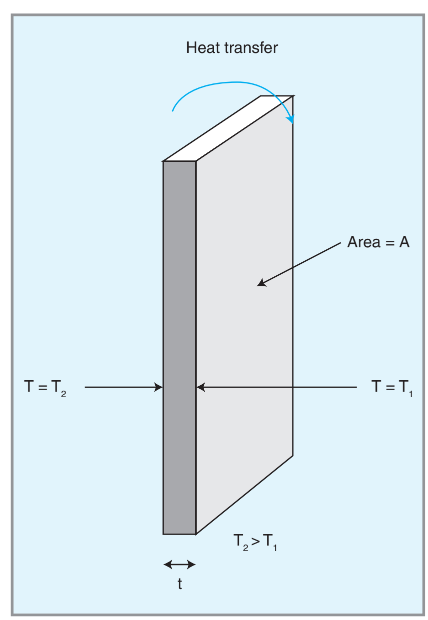


Figure 3—An example of conduction heat transfers illustrated by a thin sheet of material. The heat transfer (in watts) between the two sides of the sheet is proportional to the temperature difference of the two sides, to the area A of the sheet, inversely proportional to its thickness, t. It is flowing from hot to cold sides. The overall multiplicative coefficient is called the heat resistance.

BUJD203AX

NPN power transistor with integrated diode

Rev. 01 — 27 September 2010 **Product data sheet**

4. Limiting values

Table 4. Limiting values

In accordance with the Absolute Maximum Rating System (IEC 60134).

Symbol	Parameter	Conditions	Min	Max	Unit
T_j	junction temperature		-	150	°C

5. Thermal characteristics

Table 5. Thermal characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$R_{th(j-h)}$	thermal resistance from junction to heatsink	with heatsink compound; see Figure 5	-	-	4.8	K/W
$R_{th(j-a)}$	thermal resistance from junction to ambient	in free air	-	55	-	K/W

Photo 1—This is an excerpt from the datasheet of NXP's BUJD203AX NPN power transistor. It provides both the maximum junction temperature and thermal resistances with or without a heatsink. (Source: NXP Semiconductors)

that the transistor package is a sheet of metal and just invert the prior formula:

$$P_{THERMAL} \times \left[\frac{s}{(k \times A)}\right] = P_{THERMAL} \times R_{THERMAL}$$

Therefore, this temperature difference at equilibrium is simply proportional to the power to be dissipated: if it is 20°C for 1 W, then it will be 100°C for 5 W. In fact, components manufacturers directly give the multiplying coefficient, called the thermal resistance, in an easy to use "°C/W" unit. For example, look at the extract of an NXP BUJD203A high-voltage transistor (see Photo 1). With such a datasheet, you know that the typical junction to ambient thermal resistance when the transistor is in free air is 55°C/W. Therefore, if the transistor dissipates 2 W then its die temperature will be 110°C (i.e., 2 × 55) above ambient temperature.

This is conduction. The two other ways of heat transfer are convection and radiation. Convection is the process of heat transfer to a moving fluid, which is usually air, or it may be water in your car engine, or another exotic fluid in your last PC's thermal pipe system. Let's restrict it to air. When airflow passes around an object it takes part of the object's heat. There are two convection modes: the easiest is forced convection, where the airflow is set by a fan or something similar. In this situation, the heat transfer through convection is roughly proportional to the temperature difference between the object and the cooling fluid, and proportional to the airflow. Forced convection is similar to conduction. The natural convection, where the airflow is generated by the temperature difference itself, is more complex as the relationship between temperature difference and the heat transfer is no longer linear. The principle is, however, the same. Interested readers could have a look at *A Heat Transfer Textbook*, by John H. Lienhard and son, available on MIT's server. Roughly natural convection is close to conduction when the temperature difference between the heatsink and the ambient air is high, above 75°C, but is significantly less efficient when it is lower, as the airflow is very low. I will give you an example later.

To be complete, heat radiation is the process of heat transfer through electromagnetic radiation, meaning through the natural infrared radiation of any object hotter that –273°C. However, radiation is usually low compared to conduction and convection, except at extremely high temperatures. I will assume it could be neglected in your design.

HEATSINKS

Let's move on to a classic application. You have an electronic component that has to dissipate a given power and you must find an adequate heat dissipation solution. Suppose you are building a high-voltage linear power supply based on the BUJD203A transistor I used in the prior example (see Photo 1). Assume that the input voltage is 110-V DC, that the output voltage could be set between 1 V and 100 V, and that the maximum output current is 100 mA. Lastly, the product will only be used in a lab with an ambient temperature of up to 35°C in the summer. Do you need a heatsink, and if so, which one? First, you have to calculate the worst case power dissipation of the transistor. For a linear power supply, the worst case is achieved when the output voltage is minimal, here 1 V, and of course when the output current is maximal. Here the corresponding power dissipation will be 10.9 W (i.e., (110 V – 1 V) × 100 mA). If you had the silly idea not to use a heatsink, then the temperature difference between

Photo 2—This 54 x 38 x 100 mm Aavid Thermalloy 05518-100B heatsink gives you what is actually a 2.2°C/W heatsink for natural convection mode.

the transistor die and the ambient temperature would be, based on the $R^{th}_{(junction\ to\ ambient)}$ value given by the supplier, 600°C (i.e., 10.9 W × 55 °C/W). Smoke around the bench for sure.

To calculate the required heatsink you have to take the opposite approach: From the transistor datasheet you know that the absolute maximum junction temperature is 150°C. It is safe to not get too close to this limit, so let's take a maximum of 135°C. If the heatsink is installed outside the enclosure it will dissipate through natural convection and the ambient air around the heatsink will be at the maximum 35°C. Be careful, as a significantly higher ambient temperature would need to be calculated (or measured) if the heatsink was installed inside the product, as the "ambient" temperature would then be the maximum temperature inside the enclosure, which then depends on the enclosure inside to outside thermal resistance.

In our example, the worst-case temperature difference between the transistor die and the ambient is then 100°C (i.e., 135°C – 35°C). As the power to be dissipated is 10.9 W, then the maximum allowed thermal resistance between the die and the ambient air is 9.18°C/W (i.e., 100°C/10.9 W). However, the datasheet states that the thermal resistance between the transistor die and the heatsink, with thermal compound, is already 4.8°C/W. Hopefully, this transistor is in a fully plastic TO-220 package so you won't need to add an insulator sheet, which would add another small thermal resistance. Therefore, you need to find a TO-220-compatible heatsink with a thermal resistance lower than 4.38°C/W (i.e., 9.18 – 4.8).

Are your problems over? No; for two reasons. First let's calculate the maximum heatsink temperature: it will be 82.7°C (i.e., 35°C + 4.38°C/W × 10.9 W). This temperature is probably above a safe limit for a user-accessible part. If you want to keep it, say, under 65°C, then the maximum heatsink resistance becomes 2.75°C/W (i.e., (65°C – 35°C)/10.9 W). Second, as the heatsink is used in natural convection mode, you must include a derating factor if the temperature difference between the heatsink and the ambient air is not higher than 75°C, as the air flow will be very low. You can find the required temperature correction table on Aavid Thermalloy's website (see references). For a 30°C difference between air and heatsink temperature the derating factor is 1.25, so you will need to find a heatsink with a thermal resistance below 2.2°C/W (i.e., 2.75/1.25).

That's already a large baby. For example, the Aavid Thermalloy OS518-100B heatsink (Photo 2) will just fit the bill: 2.2°C/W. It is 54 × 38 × 100 mm. But to be safe, I would use a slightly larger one.

If you are space restricted, one way to use a smaller heatsink is to use forced convection. This means putting the heatsink inside an enclosure and adding a fan and whatever else is needed to force the fan airflow through the heatsink and out of the box. You will also find the corresponding performance factor table on Aavid's website; the improvements are impressive even with a very small fan. Take care that the heatsinks optimized for forced air convection are different than the ones optimized for natural convection: their fins are more closely spaced. Look at the heatsink on any high-end PC video card, for example. Lastly, if you need to build a heatsink yourself, you will have to measure its actual thermal resistance, which is not hard to do, or calculate it. The calculation is not difficult for forced air heatsinks; there are even free calculators on the web (see Photo 3). In the case of natural convection this may be more tricky.... Let me know if you find good references or tools adequate for this purpose.

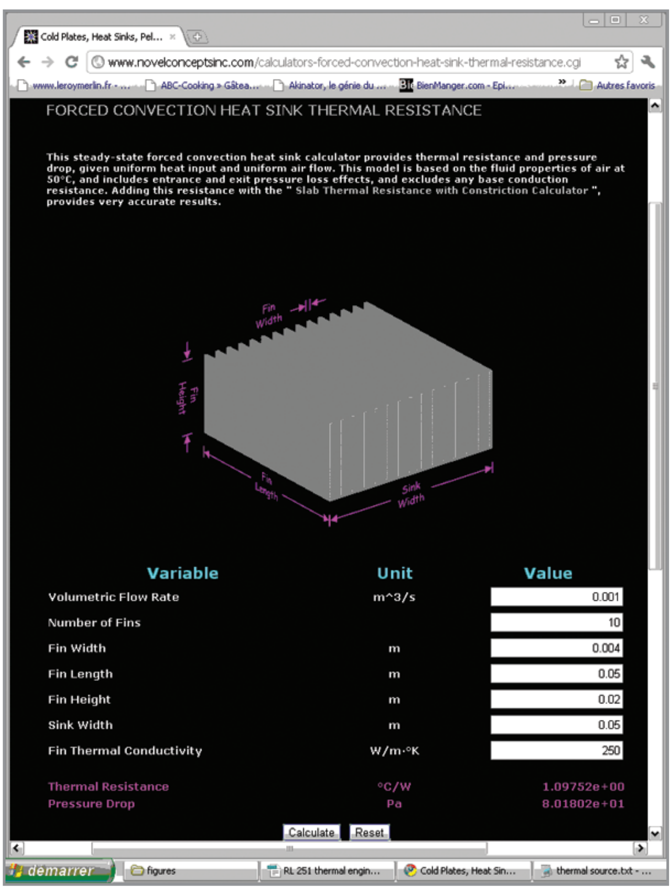


Photo 3—Novel Concepts provides interesting thermal calculators on their website, such as this forced-air heatsink calculator.

ELECTRICAL MODEL

I assume that you are more fluent with electrical schematics than with thermal problems, right? Then you will be pleased to know that there is an easy way to model a thermal system through its electrical equivalent. Remember that in the case of conduction, the temperature difference is the product of the thermal resistance and the thermal power transfer. Remember Ohm's law, V = R × I? If we map a temperature to a voltage, a thermal resistance to an electrical resistance, and a thermal power transfer to a current we have a very good analogy!

Moreover, let's recall the first example I gave you about a system in an insulated enclosure. Its temperature was rising infinitely with a rate proportional to the heat capacity of the system. And, of course, you have already guessed that the corresponding electrical model of heat capacity is simply a capacitor. Do you remember the relationship between the charge of a capacitor and the voltage across its terminals? It is simply Q = C × V. Furthermore, the heat energy stored in a system is its heat capacity (similar to C) multiplied by its temperature increase (similar to the voltage V).

In summary, you can easily model a thermal system through voltage sources, resistors, and capacitors. You can then use any electrical simulation software, such as Simulation Program with Integrated Circuit Emphasis (SPICE), to get a simulation of the thermal behavior of the system. Let's try it using the example of the high-voltage power supply discussed earlier in order to find out the evolution of the transistor die and heatsink temperature over time. We already know the power dissipated by the transistor (10.9 V), as well as the die to heatsink resistance (4.8°C/W) and the heatsink to ambient air resistance (2.2°C/W). We are missing the heat capacitance of the transistor and heatsink. This specific heatsink weighs 164 g, so its heat capacity is 164 g times the specific heat of the aluminum which is 0.902 J/g°C, resulting in a heat capacity of 148 J/°C. Similarly, we can guess that the heat capacitance of the transistor will be very low as it is mainly plastic, say 0.5 J/°C. We then have everything needed to simulate the system, as illustrated in Figure 4. The heatsink temperature rises as expected up to 60°C, with 58°C reached in about 1,000 s, or 15 minutes. The die temperature quickly rises up to 88°C then follows the heatsink temperature increase up to a safe 112°C. I have done this simulation using Labcenter's Proteus, but you can, of course, use any SPICE-based tool.

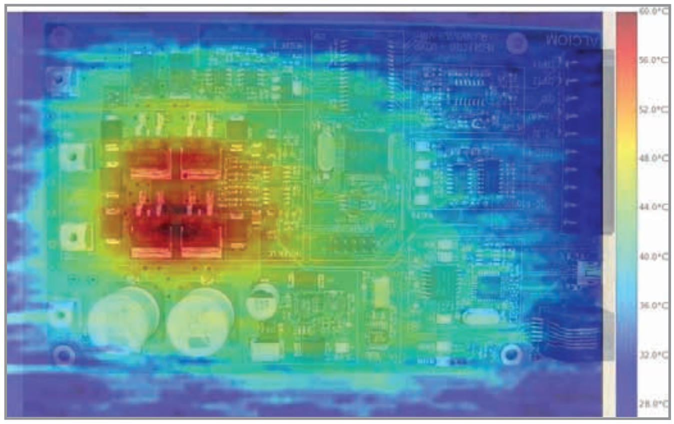


Photo 4—Using an infrared imager is an efficient way to get an analysis of the actual thermal performances of a design. Unfortunately, such tools are still quite expensive. This picture of a 50A DSP-based high-speed coil driver was taken by a low-cost thermal scanner prototype, code-named Thermicam, developed by my company. I hope we'll find time to commercialize it soon.

SIMULATION TOOLS

I'm sure that from time to time you have power dissipating components on a PCB, without any external heatsink. For example, if you build a small DC/DC converter then the switching chip or transistor may have to dissipate some hundreds of milliwatts and will usually be an SMT chip. Hundreds of milliwatts may not be much, but you will need to get this heat out. Usually the best solution is to design a PCB with as many copper planes as possible, as copper conductivity is several orders of magnitude better than epoxy! Four layers of PCBs help dramatically, as a full layer could be dedicated to the ground plane. You must also optimize the thermal transfer between the dissipating components and the copper planes. This means adding plenty of vias everywhere, and in particular, behind the key components. If the external enclosure is metallic, then of course you must provide a good heat transfer between the PCB and the enclosure through large surfaces of contact and good screws. You can also use thermal transfer pads, which are low-thermal resistance foams. Gluing one of them on top of a problematic FPGA and pressing the assembly against the side of a metallic enclosure has solved a lot of designers' dilemmas as it enables a direct thermal path.

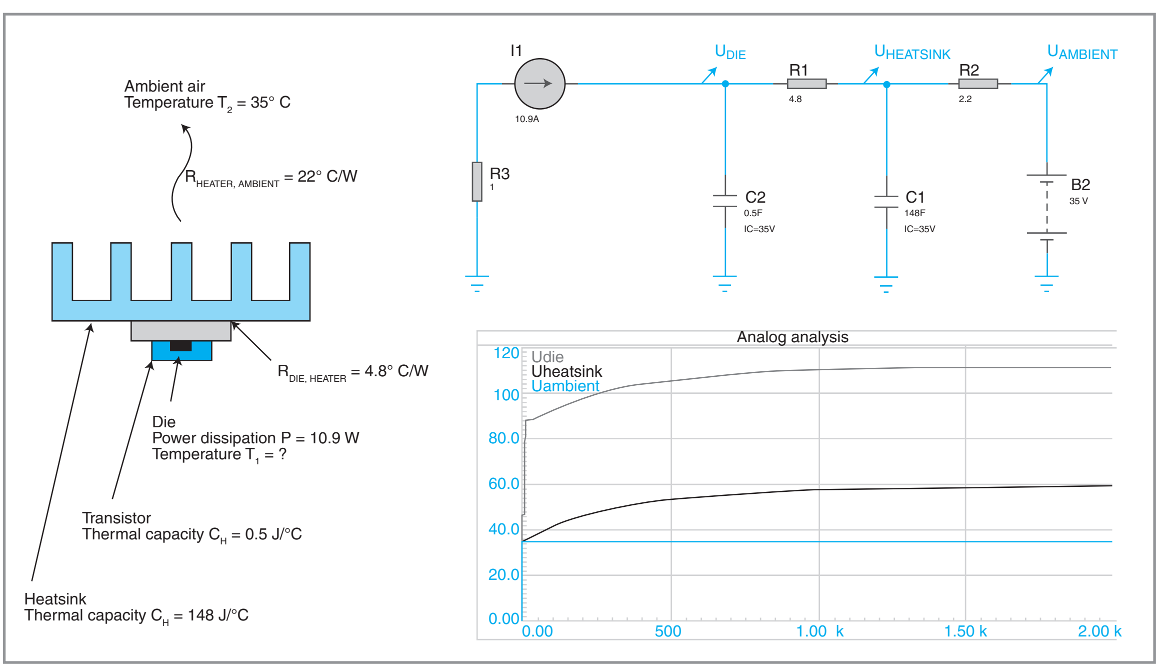


Figure 4—A thermal system can be modeled by its equivalent electrical model. Temperatures map to voltages, heat power transfers to currents, thermal resistances to electrical resistances, and thermal capacities to capacitors. Here the unknown value is the die temperature, but the power to be dissipated is fixed and modeled through a current source under Proteus.

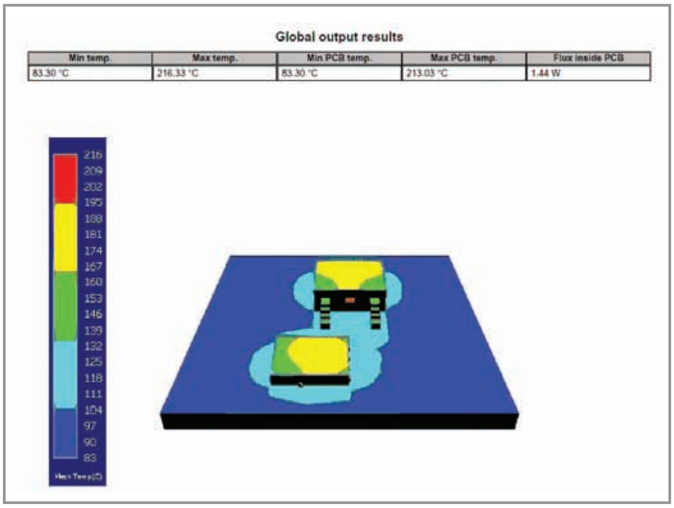


Figure 5—Here is the output of Vishay's ThermaSim online thermal simulator. Just put some Vishay piece of silicon on a board, fill in some parameters, and you will get the simulation via e-mail.

How do you know if the design is thermally correct? The first solution is to build it...and to test it. Switch on the board in the worst operating condition and measure the temperature of all critical components. This can be done manually with a low-cost hand-held infrared thermometer, but your life will be much easier if you are lucky enough to get access to a thermal imager (see Photo 4).

The other solution is to use thermal simulation software as early as possible in the development cycle. There are several good high-end packages on the market, such as Mentor Graphics's FloTHERM, but these tools are usually, well, far from cheap, and as far as I know, they need a significant level of expertise to be efficiently used. Fortunately, there are now at least two free solutions: the first one is WebTherm. This tool is offered by National Semiconductor as part of their web-based WebBench DC/DC converter design tool suite which is based on FloTHERM. It enables you to quickly evaluate the thermal performances of powers supplies developed under the WebBench.

The other solution, which I find interesting, is Vishay's ThermaSim. I love this one: it just enables you to select a couple of semiconductors (from Vishay's catalog, of course), place them wherever you want on a PCB, indicate how much power will be dissipated by each component, select the characteristics of the PCB (size, number of layers, copper spreading, etc.), click on a button, and a couple of minutes later you receive a pretty thermal simulation via e-mail (see Figure 5)! It's impressively flexible and simple to use. Give it a try.

PREVENTING DISASTER

Here we are. I know that thermal problems are a significant pain for a lot of electronic designers. Often the issue is discovered too late, and this can lead to burdensome development delays and high re-engineering costs, as everything, including the mechanical design or even the product's concept, may need to be reworked. I hope that this introductory article will help you head off the problem before it is too late. Once again, I'm not a thermal expert, but I've suffered more than once. Perhaps my advice will help you to save some MOSFETs from heat destruction!

Robert Lacoste lives near Paris, France. He has 20 years of experience working on embedded systems, analog designs, and wireless telecommunications. He has won prizes in more than 15 international design contests. In 2003, Robert started a consulting company, ALCIOM, to share his passion for innovative mixed-signal designs. You can reach him at rlacoste@alciom.com. Don't forget to write "Darker Side" in the subject line to bypass his spam filters.

RESOURCES

Aavid Thermalloy, "Temperature Correction Considerations," www.aavidthermalloy.com/technical/correct.shtml.

———, "Forced Air Correction Factors," www.aavidthermalloy.com/technical/convect_table.shtml

J. Lienhard IV and J. Lienhard V, *A Heat Transfer Textbook*, Fourth Edition, 2011, web.mit.edu/lienhard/www/ahttv201.pdf.

NXP Semiconductors, "BUJD203AX: NPN Power Transistor with Integrated Diode," Rev. 01, 2010, www.nxp.com/documents/data_sheet/BUJD203AX.pdf.

SOURCES

OS518-100B Heatsink
Aavid Thermalloy | www.aavidthermalloy.com

Proteus CAD Tool suite and analog simulator
Labcenter Electronics | www.labcenter.co.uk

FloTHERM Simulation suite
Mentor Graphics | www.mentor.com

WebTHERM Online thermal simulator and WebBench DC/DC converter tool suite
National Semiconductors | www.national.com

Online forced convection heatsink calculator
Novel Concepts, Inc. | www.novelconceptsinc.com

BUJD203A High-voltage transistor
NXP Semiconductor | www.nxp.com

WinTherm Simulation suite
ThermoAnalytics, Inc. | www.thermoanalytics.com

ThermaSim Online thermal simulator
Vishay Intertechnology, Inc. | www.vishay.com

CROSSWORD

Across

2. Used in lieu of text, e-mail, IM, etc.
6. Gives instructions to operate
7. Uses reasoning
10. Plan
11. 1:2
13. Most basic form of communication
14. A device cats are especially fond of
15. A designated location
16. The distance between two points
17. May denote exclusivity
18. Speaks the language of programmable devices
20. An energy value

Down

1. Search
3. Woven by a very large spider [three words]
4. S/H [three words]
5. Allows for sharing, if you're in the loop
8. A communication process that converts information into symbols
9. Provides performance details
12. Storage
19. Minimum of two

The answers will be available in the next issue and at www.circuitcellar.com/crossword.

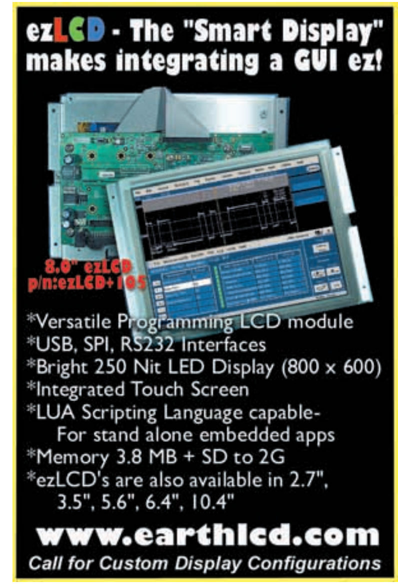
IDEA BOX

THE DIRECTORY OF PRODUCTS AND SERVICES

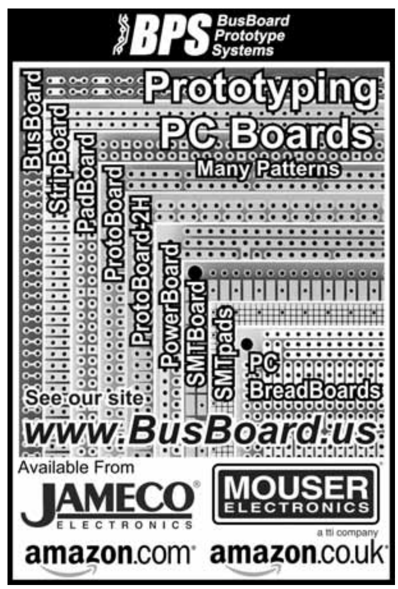
The Vendor Directory at www.circuitcellar.com/vendor/
is your guide to a variety of engineering products and services.


MaxSonar
Perfect for OEMs & Engineers
XL-MaxSonar-EZ
• High acoustic power
• Real-time calibration
• Tiny size
• $39.95USD qty. 1
MaxSonar-WR (IP67)
• Industrial packaging
• Weather resistant
• Standard ¾" fitting
• $99.95USD qty. 1
LV-MaxSonar-EZ
• Power-up calibration
• Smallest MaxSonar
• Low power, 2.5V-5.5V
• $29.95USD qty. 1
MaxSonar-WRC (IP67)
• Compact packaging
• Weather resistant
• Quality narrow beam
• $99.95USD qty. 1
www.maxbotix.com

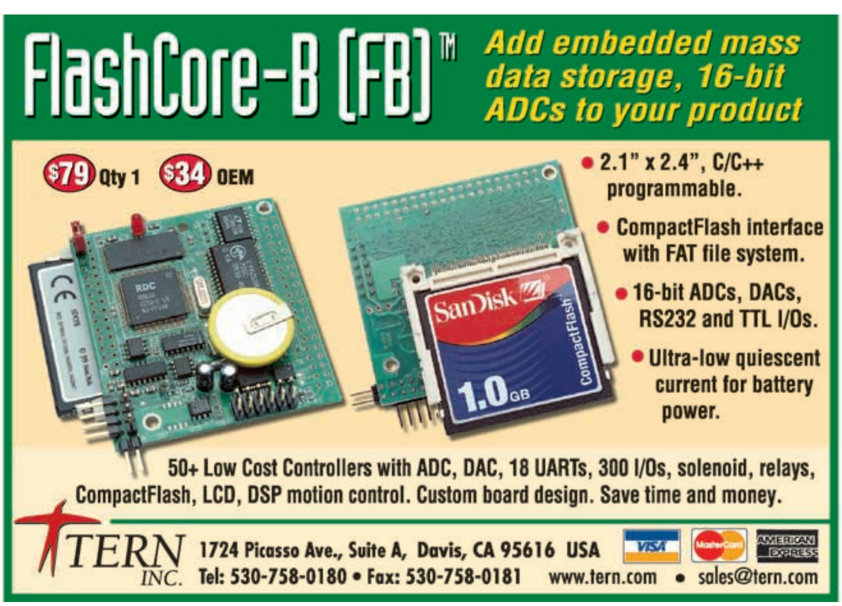

Disk On Chip
Ready for Delivery
From 16MB to 128MB Available!
Call 530-297-6073 Sales@jkmicro.com
www.jkmicro.com
JK microsystems, Inc.
International Orders Welcome

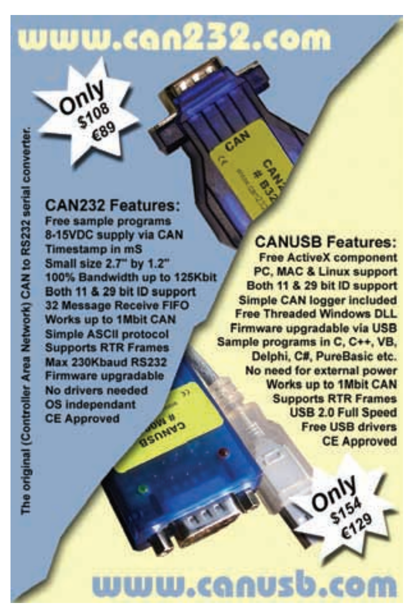

ezLCD - The "Smart Display" makes integrating a GUI ez!
8.0" ezLCD
p/n:ezLCD+105
*Versatile Programming LCD module
*USB, SPI, RS232 Interfaces
*Bright 250 Nit LED Display (800 x 600)
*Integrated Touch Screen
*LUA Scripting Language capable-
For stand alone embedded apps
*Memory 3.8 MB + SD to 2G
*ezLCD's are also available in 2.7", 3.5", 5.6", 6.4", 10.4"
www.earthlcd.com
Call for Custom Display Configurations


Your Microcontroller
Our Display Module
Add a Touch Screen to Your Embedded Product
• No special OS or Library Required.
• Programming GUI is Simple.
• Development Kit = Up and Running in Days.
20
Learn more at www.reachtech.com, call 510-770-1417 or email info@reachtech.com.
REACH TECHNOLOGY INC.
See us in booth 1245 at ESC San Jose.


BPS BusBoard Prototype Systems
Prototyping PC Boards
Many Patterns
BusBoard
StripBoard
PadBoard
ProtoBoard
ProtoBoard-2H
PowerBoard
SMTBoard
SMTpads
PC BreadBoards
See our site
www.BusBoard.us
Available From
JAMECO ELECTRONICS
MOUSER ELECTRONICS
a tti company
amazon.com amazon.co.uk


i2cStick™
Personal, Powerful...
Pocket-Size
MCC
Micro Computer Control
www.mcc-us.com


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


www.can232.com
Only $108 €89
The original (Controller Area Network) CAN to RS232 serial converter.
CAN232 Features:
Free sample programs
8-15VDC supply via CAN
Timestamp in mS
Small size 2.7" by 1.2"
100% Bandwidth up to 125Kbit
Both 11 & 29 bit ID support
32 Message Receive FIFO
Works up to 1Mbit CAN
Simple ASCII protocol
Supports RTR Frames
Max 230Kbaud RS232
Firmware upgradable
No drivers needed
OS independant
CE Approved
CANUSB Features:
Free ActiveX component
PC, MAC & Linux support
Both 11 & 29 bit ID support
Simple CAN logger included
Free Threaded Windows DLL
Firmware upgradable via USB
Sample programs in C, C++, VB, Delphi, C#, PureBasic etc.
No need for external power
Works up to 1Mbit CAN
Supports RTR Frames
USB 2.0 Full Speed
Free USB drivers
CE Approved
Only $154 €129
www.canusb.com

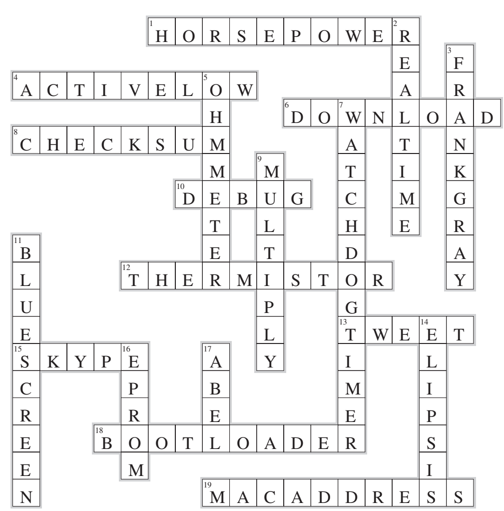
CROSSWORD ANSWERS from Issue 250

Down

2. **REALTIME**—A quick response
3. **FRANKGRAY**—Founded a binary system [two words]
5. **OHMMETER**—Used to measure electrical resistance
7. **WATCHDOGTIMER**—Performs an intervention when things go wrong [two words]
9. **MULTIPLY**—What it means to *, ·, or ×
11. **BLUESCREEN**—Stop error [two words]
14. **ELIPSIS**—Used to trail off
16. **EPROM**—Exposure to ultraviolet light makes it disappear
17. **ABEL**—Design language (related to Cain)

Across

1. **HORSEPOWER**—HP (hint, not a printer maker) [two words]
4. **ACTIVELOW**—Not a lot of logic
6. **DOWNLOAD**—Tap into your computer's memory
8. **CHECKSUM**—Ensures things are running smoothly
10. **DEBUG**—To exterminate, in computer terms
12. **THERMISTOR**—Sensitive to the degree
13. **TWEET**—An update a bird may comprehend
15. **SKYPE**—Software used to talk over the 'Net
18. **BOOTLOADER**—Needs to run before an OS can start
19. **MACADDRESS**—a unique ID that follows IEEE-managed rules [two words]



INDEX OF ADVERTISERS

PREVIEW of July Issue 252

Theme: Internet & Connectivity

Passwords Through a Touchpad: A Gesture-Based Security System

The TROBOT 3.0: A Miniature Articulated Robot

Find a New Direction: Build a Low-Power Digital Compass

Battery Management Electronics: Build a Li-Ion Battery System Solution

---

THE CONSUMMATE ENGINEER **Sine Wave Synthesizer**

FROM THE BENCH **Vehicle Diagnostics (Part 2):** Packet Data and User Interface

LESSONS FROM THE TRENCHES **Design Development (Part 2):** Product Implementation

PRIORITY INTERRUPT

by Steve Ciarcia, Founder and Editorial Director

Live by the PC, Die by the PC

I don't know whether to eat crow or simply throw myself at the feet of the masses and beg forgiveness. For 30-plus years, I've been using PCs without significant problems and simply smiling whenever anyone else described their "blue screen of death" episodes. This is the first time it happened to me! Don't get me wrong. I haven't been stupid. I back up all of my data with one of those external USB drives that can reload the entire hard drive from an earlier date if you get a virus or otherwise toast your drive contents. Of course, when that doesn't work, you're screwed.

Because I travel to the cottage each year, I changed my main PC from a desktop to a "transportable" laptop three years ago. Hardly a "laptop," the 11.5-lb, quad-core, 17″ gaming computer was the hairiest piece of portable hardware on the market at the time. Of course, there's also a message when a laptop comes with a 200-W power supply and three fans! The 135-W Q6600 processor, along with two hard drives and the graphics card, created a veritable blast furnace out the back. But, however insane the packaging, it worked fine for almost three years.

From the beginning, I've always run two video monitors along with piles of external USB devices. Three or four times in the last three months both displays suddenly started flashing and displaying extra "artifacts." Resetting the computer immediately cleared the problem, so I ignored it, even as I upgraded my extreme-computing model to include two more USB-connected video monitors. The four monitors—a nice feature, I might add—worked, but I sensed I was pushing the envelope. Windows Task Manager said the CPU usage was 85% (10% to 20% was typical) and the fan exhaust seemed especially hot. Then it happened: artifacts, flashing, and death!

Rebooting was curious. The password sign-on screen came up on USB monitor 4, not the laptop direct-connected monitors! After entering the password, the hard drive went off doing its regular flashy thing and then nothing: four black displays! I could not even boot into Safe mode after disconnecting all of the extra monitors. Okay, it was time to load the recovery CD-ROM—spin, spin, crank, crank. The disk seemed to be doing something, but with no display, there was no way to tell the recovery program to reload whichever drive or to do whatever. Basically, with a dead video card—which I now realize must have been the true problem—I was dead in the water. With 15 days left on my three-year warrantee, the computer company happily told me that labor was still free, but I could expect to pay $600 to $1,000 for a replacement of the custom "MXM-IV" Nvidia 8800 GTX video card in that laptop. Gotcha!

Okay, buying the best fire breather three years ago was a great idea, but there was obviously a significant downside to the packaging. If my extreme-performance computer had been a desktop instead of a laptop, I could've easily swapped out the 8800 GTX card for about $85 at current eBay prices. It would have been a bit heavier to transport, but right now I have 11.5 lb of junk.

Fortunately, this old dog does learn new tricks. I'll always continue to keep a laptop as a backup—currently in use writing this editorial—but I'm going back to a desktop as a main computer. Like before, I still want the biggest bang for the buck—okay, so I put in a few more dollars than you would—and even though I've never played a computer game, it's a gaming computer.

I buy fire breathers, so I don't feel left behind quite as fast. I've been coveting the i7-980X for a while. But hear this, Intel: I'm not spending $1,000 on a processor, no matter how many cores it has! The Q6600 quad-core I have been using for the last three years has a PassMark CPU score of 2,975. The i7-980X scores 10,515. The new Sandy Bridge processors appear to be a more cost-effective alternative, but try and buy a box with one. Forget HP and the regular desktop manufacturers. As of this writing, even the $4,000 Dell Alienware quad-core gaming computer still uses an i7-960. It has a respectable score of 6,889, but still not enough fire.

Thanks to all of those crazy gamers, however, there are a few online sources ahead of the curve. After two days of comparing benchmarks and reviews, I ordered the following system: a mini-tower desktop with a 3.4-GHz i7-2600K quad-core CPU (PassMark score 9,297) additionally configured for 10% to 30% over-clocking, 8 GB of DDR3-1600 MHz memory, two 1-TB SATA-III drives, USB 3.0, and a 2-GB GeForce Nvidia 560 Ti video card. The good news is that this time my fire breather is also liquid cooled.

So what happens with my three-year-old $3,000 doorstop? During my web search for a replacement card, I came across an interesting entry in one of the technology forums about repairing bad video cards. Apparently "artifacts" are often caused by solder whiskers and fissures. Popping your video card in the kitchen oven for a few minutes can solve the problem by reflowing the solder. I'm certain there is a lot more to it, but this and other technology myths sure sound like good editorial.

steve.ciarcia@circuitcellar.com

Steve