www.circuitcellar.com

CIRCUIT CELLAR®

THE MAGAZINE FOR COMPUTER APPLICATIONS

#118 MAY 2000

EMBEDDED APPLICATIONS

Dallas 1-Wire Weather Station

Embedded Networking at the Racetrack

Equipment Grounding

Home Controlled Heating Systems

$3.95 U.S. ($4.95 Canada)

CIRCUIT CELLAR ONLINE

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

— FEATURES —

Making the Change
From Standalone to Internet Appliance
Edward Steinfeld
Plenty of devices have made the transition from standalone product to Internet appliance, but Edward looks at why you might consider making the change in your next project and how you can get the most out of Internet appliances.

Recycle Your Code
Conversion and Optimization Techniques
Stephen Bowling
Portable code and design reusability have become increasingly important as designers are given shorter design cycles and time-to-market goals. As Stephen explains, the PIC18C*xxx* architecture lets you get more mileage out of your programming efforts.

The SHARC in C
Mike Smith
In a recent project, Mike set out to develop DSP algorithms suitable for producing an improved sound stage for headphones. Using the Analog Devices 21061 SHARC, he modified the phase and amplitude of the audio signal before it is sent to the ear, thus creating "virtual" speakers that give the effect of listening via room speakers.

Resource Links
- **Real Time Clocks and How to Set Them**
- **Rugged Data Storage**

Bob Paddock

Test Your EQ
8 Additional Questions

— COLUMNS —

Considering the Details
Looking Good
Using a Graphics-Based LCD Module
Bob Perrin and Tak Auyeung
Advances in technology have brought graphics-based LCDs into the budget range of many embedded applications. Unfortunately, graphics modules aren't as easy to work with as the ubiquitous character-based display. Using C, Bob and Tak explain what it takes to implement a graphics-based module.

Lessons from the Trenches
Embed This PC
Part 4: Designing Peripherals In
George Martin
George has covered the CPU options and selected DRAM and BIOS flash memory devices for the embedded '486 project. In the final part of this series, he ties it all together and takes a look at what kinds of peripherals you might want to add on.

Silicon Update Online
The Retreat of Silicon
Tom Cantrell
Most of the attention being paid to the march of silicon is focused on the mega-chips. But, with the latest in less-is-more solutions, things might be about to change.

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

10 **A Reliable Network for Embedded Systems—**Getting on Track
Val Popescu & Gary Gibson

18 **The Shocking Truth About EMC**
Part 2: Practical Application
George Novacek

28 **Weather Data—**Getting More Than One Wire's Worth
William Beals & Russell Chadwick

32 **Buying Power—**A Low-Cost Power Supply
Robert Kondner

36 **Building a RISC System in an FPGA**
Part 3: System-on-a-Chip Design
Jan Gray

68 **Embedded Living**
HCS = Heating Control System?
Mike Baptiste

60 **MicroSeries**
Op-Amp Specification—Part 2: Getting Some Input
Joe DiBartolomeo

74 **From the Bench**
Digital and Analog Output Control
Taking Route X-10
Jeff Bachiochi

78 **Silicon Update**
On the Road Again
Part 1: Going Places with Silicon
Tom Cantrell

INSIDE ISSUE 118

EMBEDDED PC

48 **Nouveau PC**
edited by Harv Weiner

50 RPC **Real-Time PCs**
Real-Time Executive for Multiprocessor System
Part 1: Introduction to RTEMS
Ingo Cyliax

54 APC **Applied PCs**
Under the Covers
Part 2: Applications via NT Embedded 4.0
Fred Eady

Task Manager 6
Rob Walker
It Takes All Kinds

New Product News 8
edited by Harv Weiner

Test Your EQ 83

Advertiser's Index 95
June Preview

Priority Interrupt 96
Steve Ciarcia
Publishing 101

TASK MANAGER

It Takes All Kinds

We recently circulated a personality test here at *Circuit Cellar* to see what kind of people it takes to put the magazine together. I took the test, and the results came back negative. Well, not exactly, but I'm not sure how to take the news. The test said I was, among other things, logic oriented, contemplative, thoughtful, and a "typical computer geek."

Looking at the characteristics of other personality types, I see that all of the characteristics for the other five personalities are definable terms like extrovert, impatient, loyal, and so on. It seems obvious that whoever came up with this test should add "often stereotypical" to the list of his or her character traits. It's a good thing that I also happen to be a person who "prefers to avoid confrontation," otherwise I might have considered making a visit to the Smell & Taste Treatment and Research Foundation in Chicago where this test originated.

No doubt, there will be some of you who read this and get excited. ("A typical computer geek? All right, he's one of us!") To those of you who may be a little leery about subscribing to a magazine with a typical computer geek as the Managing Editor, relax. The personality test revealed that we also have a few perfectionists on board and even a few "generally successful people" here at *Circuit Cellar*. Now, tell me that isn't reassuring.

If there were a point to all of this, it would be that it takes a variety of people to put the magazine together each month. Anyone who follows our columnists can tell you that.

Tom enjoys monitoring the march of silicon and keeping us posted on what's new and how it effects us. This month he takes us under the hood to explain how the latest automobiles are networked. Ingo likes to dig in and explore the details of topics like GPS, FPGAs, and in this issue, he kicks off a series on RTEMS. Fred Eady knows how to appreciate the simple things in engineering, and each month he preaches that, "It doesn't have to be complicated to be embedded." No matter how much you love your cubicle, Mike Baptiste's home control ideas can make your home a better place to be. And, whether Jeff's riding the elevators here at HQ to test accelerometers or continuing the *Circuit Cellar* tradition of home control coverage via X-10, he always brings us something useful from the bench.

Our monthly themes help provide variety among the feature articles each month, too. This month's theme, Embedded Applications, covers everything from network nodes in high-performance racecars to a weather station that uses amateur radio to transmit weather data to the Internet. Coming up in the second half of the year, you'll be seeing Robotics, Internet & Connectivity, and Wireless Communication issues, to name a few. There will be plenty of great project articles on a variety of topics.

We can't be everything to everyone, but in response to your suggestions, we added a few new themes this year. It's time to set up next year's editorial calendar, so if you have a topic you would like to see covered in 2001, drop me an e-mail. After all, we don't want to forget those people who are "easily bored with routine."

rob.walker@circuitcellar.com

CIRCUIT CELLAR®

THE MAGAZINE FOR COMPUTER APPLICATIONS

EDITORIAL DIRECTOR/PUBLISHER
Steve Ciarcia

MANAGING EDITOR
Rob Walker

TECHNICAL EDITORS
Jennifer Belmonte Rachel Hill
Jennifer Huber

WEST COAST EDITOR
Tom Cantrell

CONTRIBUTING EDITORS
Mike Baptiste Ingo Cyliax
Fred Eady George Martin
Bob Perrin

NEW PRODUCTS EDITOR
Harv Weiner

PROJECT EDITORS
Steve Bedford
Janice Hughes
Elizabeth Laurençot
James Soussounis
David Tweed

ASSOCIATE PUBLISHER
Sue Skolnick

CIRCULATION MANAGER
Rose Mansella

CHIEF FINANCIAL OFFICER
Jeannette Ciarcia

CUSTOMER SERVICE
Elaine Johnston

ART DIRECTOR
KC Zienka

GRAPHIC DESIGNER
Mary Turek

STAFF ENGINEERS
Jeff Bachiochi John Gorsky

QUIZ MASTER
David Tweed

EDITORIAL ADVISORY BOARD
Ingo Cyliax
Norman Jackson
David Prutchi

Cover photograph Ron Meadows—Meadows Marketing
PRINTED IN THE UNITED STATES

CONTACTING CIRCUIT CELLAR

SUBSCRIPTIONS:
INFORMATION: www.circuitcellar.com or subscribe@circuitcellar.com
TO SUBSCRIBE: (800) 269-6301 or via our editorial offices: (860) 875-2199
GENERAL INFORMATION:
TELEPHONE: (860) 875-2199 FAX: (860) 871-0411
INTERNET: info@circuitcellar.com, editor@circuitcellar.com, or www.circuitcellar.com
EDITORIAL OFFICES: Editor, Circuit Cellar, 4 Park St., Vernon, CT 06066
AUTHOR CONTACT:
E-MAIL: Author addresses (when available) included at the end of each article.

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

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

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

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

NEW PRODUCT NEWS

Edited by Harv Weiner

INDUSTRIAL CONTROLLER

The **RPC-2350G** is an industrial controller with EL or LCD graphic display capability. Its numerous analog, digital I/O, and counter lines make it a core processor for OEM applications. A built-in programming language supports graphic functions and I/O. Applications for the controller include instrumentation, analysis, and industrial machine control.

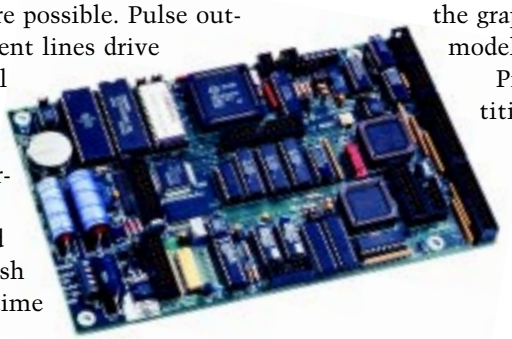
Analog I/O consists of eight channels of 12-bit A/D, two channels of D/A, and four 20-mA outputs. Its 49 digital lines are used for pulse outputs, counting, and timing. A 24-bit counter connects to a proximity sensor, quadrature encoder, or 20-MHz pulses. Frequency and pulse width measurements are possible. Pulse outputs are provided. Nine high-current lines drive solenoids, relays, LEDs, and small motors. Two RS-232/485 network ports are available. An SPI port connects to a variety of external devices.

It has 512 KB of battery backed RAM and 512 KB of supported flash memory. Its battery-backed real-time clock and RAM run for about three years. Programs are stored in flash memory, while formulas, constants, and recipes can be stored in RAM or flash memory. Power requirements are 5 V or 7–30 VDC.

CAMBASIC (an event-driven, floating-point, multitasking programming language) speeds program development. Programs are downloaded using a serial port, then the CAMBASIC operating system compiles and runs the programs. Application programs that show how to use the major functions are included.

The RPC-2350G model includes EL and LCD graphics ports. The RPC-2350 model doesn't include the graphic display port or regulator. Both models include a character display port.

Pricing starts at **$375** in single quantities.



Remote Processing
(800) 642-9971
Fax: (303) 690-1875
www.rp3.com

NEW PRODUCT NEWS

RISC-BASED C-ENGINE

The **ADPl C-Engine** is a low-power compact RISC-based C-engine that can be used in applications where multiple serial ports, analog inputs, analog outputs, digital I/O, and on-board memory storage are required. Applications include environmental and wastewater, test and measurement, control and data acquisition, SCADA, and portable or stationary applications.

Its compact size is ideal for OEMs needing to add features to existing products. The engine can develop new designs. The ADPI C-Engine is expandable with add-on modules available from ADPI or with proprietary designs based on customer specifications.

The system can be programmed using the GNU shareware compiler, which eliminates the need to license a BIOS or operating system. Software libraries are provided by ADPI.

The C-Engine features an H8/3006 processor with 256 KB of onboard flash memory (upgradeable to 2 MB). A compact flash slot that supports up to 64 MB and an onboard 16-bit memory I/O expansion is provided. There are eight 10-bit A/D ports or six 10-bit A/D and one 8-bit D/A port. Its LCD display port supports parallel LCDs up to 4 × 40, and its keypad interface supports up to 3 × 4 matrix. An input voltage of 2–3 VDC battery or 6–20 VDC is required. The ADPI C-Engine costs **$188**.

Analog & Digital Peripherals, Inc.
(937) 339-2241
Fax: (937) 339-0070
www.adpi.com

FEATURE ARTICLE

Val Popescu and
Gary Gibson

A Reliable Network for Embedded Systems

Getting on Track

Getting a low-cost and reliable embedded network on track can be quite a chore. Val and Gary put together a system of network nodes that is not only on track, but on *the* track—in a high-performance racecar! Let's take a ride with LogNet.

There are communication protocols for nearly every combination of speed, media, data structure, distance, and cost. The novelty of another protocol would probably be lost on most engineers unless it addresses a rather peculiar application space.

The LogNet protocol and implementation was initially conceived for connecting data acquisition and control modules in a high-performance racecar. The environment is characterized by intense electrical and magnetic noise, heat, vibrations, little power availability, and tight spaces. The physical layer used was the RS-485 standard. The cost of a network node was under $7, with a half-inch square of board space (inserted in the cable itself). The power needed was between 10 and 20 mW.

The biggest developmental challenge was making the network hardware and protocol implementation sufficiently robust to keep the network running in the harsh environment. Fault-tolerant methods are employed to trace and checkpoint transactions to resume operation without data loss after network node faults. While Cyclic Redundancy Check (CRC) ensures corrupted packets are retransmitted, data integrity must be preserved even in the event that the node fails and is subsequently restarted by a watchdog timer (WDT).

A combination of polling and token passing ensures that each node can become master and control the network. A specially designated Hub node ensures fairness by preventing idling nodes from unnecessarily taking control of the network access.

Although initially developed for a specific racecar application, the fault-tolerant, low-power, low-cost network can find much broader use in industrial and automotive applications. Because of the small size, the network interface can be embedded in the network cable (see Figure 1).

LOGNET HIGHLIGHTS

The cost, power, and size of the node efficiently carries event-driven (instrumentation, control) and constant data rate (telemetry) messages. There is a high reliability and fault tolerance for harsh environments.

A multilevel priority access scheme supports devices with widely varying data rates and required response times. It implements a client/server protocol with peer-to-peer capabilities by using master/slave and token-passing media access methods. Also, there is a token-passing mechanism where the Hub controls the token advance signal going to all nodes.

The efficient network firmware uses less than 2 KB for the network Hub and less than 1 KB for the other network nodes.

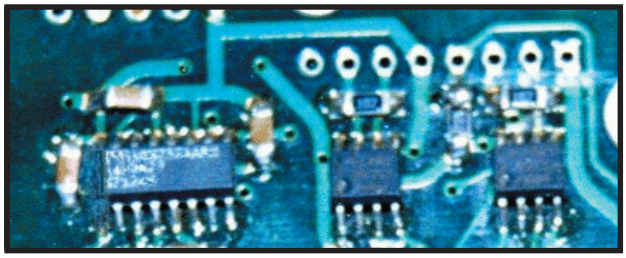


Photo 1—*This is the LogNet node PCB with two RS-485s and a PIC 16C505 that was used in the development of the racecar application.*

LOGNET PROTOCOL LAYERS

LogNet's physical layer consists of two RS-485 electrically-compliant twisted pair signals: Data and TokenClk. In Figure 1, each node has a RS-485 port and a UART (point-to-point) link to the companion module.

The basic concept of the data link layer is a virtual bus using a master/slave methodology. The network uses a Hub that serves as the primary master and arbitrator, allowing for multilevel prioritized network access. The Hub also allows for simplified central diagnostics. Data is transferred through network transactions initiated by the current master and executed by the addressed slave either by responding with data or acknowledging data transfer.

A network transaction is a sequence of one or more discrete framed packets where each packet is a string of characters. These packets are framed by being delimited with symbols and special two-character sequences. Each packet contains CRC data to detect corrupted transmissions.

Serial data is transferred asynchronously on the Data signal in a subframe, using a standard UART format synchronized with start and stop bits. The TokenClk signal is generated by the Hub and read by the other network nodes. TokenClk synchronizes token passing in the network's multimaster mode. In addition to that, TokenClk can be used by the Hub to force a current master to relinquish its master state, enforcing network access prioritization and determinacy.

The network transaction packets are divided into fields that define command opcode, the NID of destination node, control bits, and the message payload carried for the application layer, followed by a two-byte CRC. These packets are delimited by one- or two-byte symbols, beginning with a packet header and ending with a packet trailer. Request, response, and acknowledgement packet headers and trailers use unique symbols.

PACKET DEFINITIONS

The three different packet types are request, response, and acknowledgement. Request packets initiate a network transaction and may contain network data as part of the data link layer protocol, an application message in the case of a write request, or the information necessary to a read operation. Response packets carry application messages or network data from a slave to the current master.

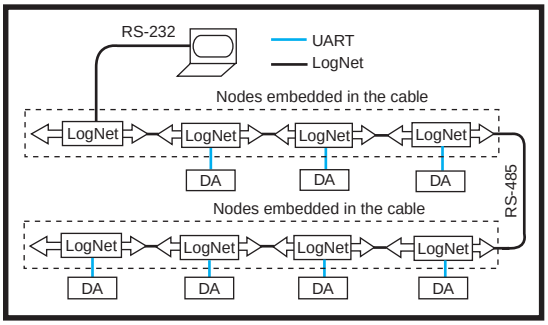


Figure 1—*In a LogNet application, the Data Acquisition (DA) modules are connected via UART links to network nodes embedded in the cable.*

Acknowledgement packets are transmitted by the data recipient to complete the transaction reliably. Figure 2 shows the packet formats.

Network addresses (NIDs) consist of an 8-bit byte allowing addressing of up to 255 nodes. Request packets start with a destination NID byte, then a Command byte, followed by a CtrlLen byte. The Command byte specifies the operation to be performed. The CtrlLen byte uses two fields specifying control information and the number of bytes to be read or written. Zero to 32 data bytes follow, ending in a two-byte CRC. Longer application messages require multiple transactions.

Response packets start with a Slave End byte that uses two fields specifying the status of the transaction and the number of bytes read. Slave End is followed by 0 to 32 bytes of data, ending in a two-byte CRC. The Slave End byte also may indicate that the request packet was corrupted.

Slave End provides status that may include a request to the current master to initiate another transfer with four levels of priority: low, medium, high, and urgent. This status mechanism allows a slave that needs to transfer additional data to secure the network for a longer period of time. The Hub also may use this information to temporarily raise the network access priority of the current master.

Acknowledgement packets contain only an AckStatus byte and a two-byte CRC, and are issued to complete a transaction or to indicate that the previous write request or read response was corrupt (see Table 1).

NETWORK TRANSACTIONS

Completing network transactions transfers data. The current master with a request packet initiates all network transactions. In the case of a read command, the addressed slave replies with a response packet. After the master successfully receives the read response, it issues an acknowledgement packet. In the case of a write, the addressed slave replies with an acknowledgement packet. Most of the commands in Table 1 that aren't labeled as read or write are write requests transmitting network data. The addressed slave acknowledges them. If any packet becomes corrupted or lost in transmission, the entire transaction is restarted.

Protocol guarantees delivery of one copy of each application message or network data in the order they are transmitted. Because a corrupted acknowledgement could cause retransmission of the original packet, duplication is avoided by using

Command	Description
Write NID	Assign an NID to a node, addressed by Manufacture ID
Read NID	Read the NID of a node, addressed by Manufacture ID
Read	Read 1 to 32 bytes
Write	Write 1 to 32 bytes
Read P	Read 1 to 32 bytes, allow partial reads
Write P	Write 1 to 32 bytes, allow partial writes
Set TokenID	Set the system in "Token" mode and the current Token ID
Cancel Token	Cancel "Token" mode
Sync	Synchronize a node (soft reset)
Quiet	Suspend a node's activity (stand by)
Wake Up	Resume the node's activity

Table 1—*Here's a list of the network protocol commands used for request packets.*

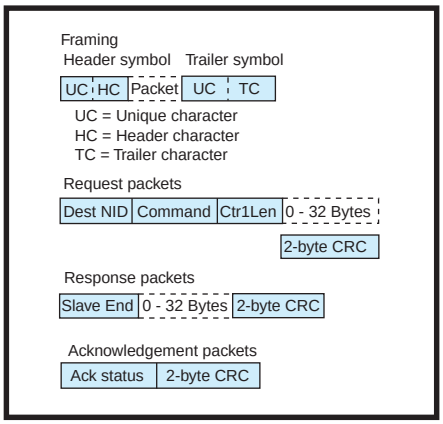


Figure 2—*LogNet uses media access protocol framing and three packet formats.*

a duplicate transmission bit in the command byte.

MEDIA ACCESS CONTROL

The two modes used by LogNet protocol to control media access are polling and token passing. They assume that one of the nodes is declared master while the rest are relegated to slave status.

The network initialization firmware running in the Hub is provided with a configuration file containing the unique manufacturer ID of each of the nodes. From this information, the Hub assigns consecutive NIDs to each node in the system. The application in the Hub can also set the network access priority of each node. After initializing the network, the Hub becomes its first master.

The token-passing mode allows for multiple masters. The Hub manages the token-passing protocol. Figure 3 shows the protocol's main states. The Hub sets the network in the token-passing mode by broadcasting a packet to all nodes to set the current Token ID. This mode is maintained until mastership of the network is accepted by one of the nodes taking the token.

While in the token-passing mode, the Hub generates the TokenClk signal. Each time a rising clock edge is received by the network nodes, they increment an internal Token ID counter. When the Token ID counter equals the NID of the node, the node either accepts the token and assumes control of the network by issuing a TakeToken packet or passes the token by issuing the PassToken packet.

When a node takes the token and becomes the new master, it initiates network transactions as in the polling mode until it voluntarily relinquishes or the Hub forces it to relinquish its master state. The Hub can activate the TokenClk line, causing the TID to increment and thus be different than the node's NID. This alerts the master to pass the token back to the Hub with a CancelToken packet. The Hub then restarts the token-passing mode using the SetTokenID packet and activates the TokenClk signal, allowing the token to be passed to the next possible network master within the priority scheme.

The Hub implements additional error-handling procedures to deal with lost or duplicated tokens, which happens when a node fails to pass a token or token-passing packets are corrupted.

FAULT TOLERANCE

All data transfer transactions include acknowledge packets that report CRC errors. Corrupt packets are retransmitted a number of times (user defined). If packets are still corrupt, the current master sends a message to the Hub asking it to take control and run diagnostics to isolate the point of failure. The failing node (or nodes) is logically removed from the network.

Each node implements a watchdog timer routine that is called each time the node hangs. The routine checks the transaction queues and the current transaction status in order to resume and complete the transaction. If no internal registers are corrupted, the nodes participating in the transaction will see only a delay in finishing the transaction. If the node isn't able to safely resume the transaction, it terminates the transaction with an error acknowledgement. To safely resume suspended transactions, the nodes trace and checkpoint the progress of transactions.

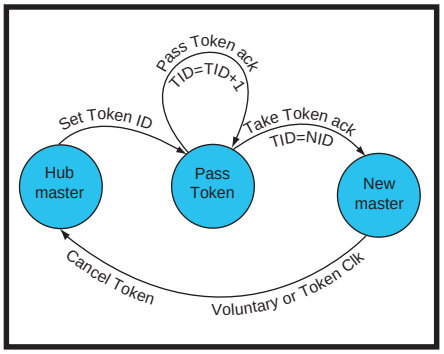


Figure 3—*This figure shows the token-passing mechanism. The Hub issues TokenClk, which makes all nodes increment the Token ID (TID). When a node notices a TID equal to its Node ID (NID), it tells the Hub to stop the TokenClk and acquire master status.*

The Hub sends a synchronization packet when the network is idle (and has master status) just to make sure all the nodes are up and working. If a node does not respond to a synchronization packet within a defined time interval, the Hub assumes that the node failed. Then, the failing node is removed from the network.

The Hub maintains a network activity timer. If the Hub has not seen any message for a period of time, it assumes the current master is idling or has failed and forces the node to relinquish the master status. If there is no response, the Hub assumes the current master failed in the worst possible way, and it drives the network. In this case, the Hub records the fault and cycles the power to the nodes, followed by network initialization and diagnostics.

ENSURING FAIRNESS

Ensuring fairness in accessing the network is the goal of most protocols. LogNet provides the token-passing mechanism to give each node a chance to become master. Fairness also assumes the master will surrender its position, giving the chance to another node. The Hub can also enforce a time-slicing policy by asking the current master to surrender its position at the end of the transaction in progress if it held the position longer than the preset time slice.

PHYSICAL IMPLEMENTATION

Figure 4 shows the implementation of a node using a Microchip PIC microprocessor. To reduce the power requirements, the chip's sleep mode is activated each time the node surrenders its master position. The chip is awakened by a change on an input pin. The power consumption in standby mode is less than 5 µW and in operation is less than 10 mW (at 5 V). The power can be supplied by the

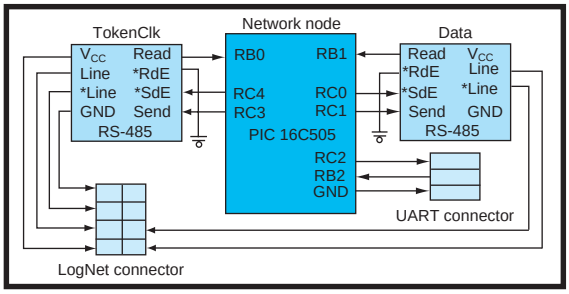


Figure 4—*This is the network node implementation that you can insert right into the network cable.*

companion embedded module or by the network cable.

Port B is used for reading in all three data lines: TokenClk Read (RB0), Network Data Read (RB1), and serial link RXd (RB2). Port C is used for output: Network Data Send (RC1), serial link TXd (RC2), and Network Data Send Enable (RC0). In addition, the master drives the Token Clock: TokenClk Send (RC3) and TokenClk Send Enable (RC4). The serial link is implemented as UART or SSP (see Figure 4 and Photo 1).

The interface node firmware uses less than 1k instructions and less than 60 8-bit registers, whereas the Hub uses almost 2k instructions. The data transfer rate is largely determined by the operating speed of the RS-485 and the clock speed of the microcontroller implementing the node. Peak data rates of up to 4 Mbps can be reached.

APPLICATION EXAMPLE

The first application of this network was with a Formula Atlantic-series racecar. The networked units were two data acquisition modules (Engine and Chassis), a steering wheel mounted LCD and warning LED lights, an infrared beam receiver, and an auxiliary unit for adjusting the brake bias, power shifting (shifting without lifting off the throttle), and voice messages (see Figure 5). The data transferred on the network included status of the steering wheel buttons, messages for the display and the voice unit, turning on or off the LED lights (engine alarm, lap time progress, etc.), wheel speed, engine RPM, and so on.

The Chassis module was designated as the Hub. It uses its UART link for data downloading to a PC and telemetry. Each node has an application using data transferred by its media access layer.

Broadcast messages are often used. For example, many processes in the system are indexed by time (beginning of a lap) and by the car's position on the track, which is calculated by multiplying the number of wheel revolutions by the wheel circumference.

The Chassis node is connected to a wheel sensor that generates a signal for each wheel revolution and is in charge of maintaining the car's position data. Because other nodes needed to know the car's position, the Chassis node broadcasts a new wheel revolution message to other nodes on the network. Similarly, the node receiving and decoding the infrared beam that marks the beginning of a lap broadcasts a new lap message. The Engine node has the engine RPM data, which is useful to other nodes hence it also broadcasts on the network.

Many functions in a racecar are critical in terms of the well being of the car and driver. So, maintaining the operational network is critical.

The Engine node collects engine parameters (oil and fuel pressure, etc.) and compares them to preset limits. If one of the engine parameters is off-limit, the Engine node grabs the token to become master and sends a message to the Display node, which formats and displays the parameter in question on the LCD screen and flashes the alarm LEDs.

The Engine node also sends a message to the Auxiliary node, which composes a voice message ("Oil pressure too low") and feeds it into the car radio. Most racecars are not allowed

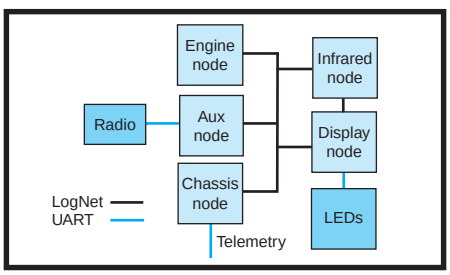


Figure 5—*Here's an overview of the racecar system.*

Photo 2—*Talk about a quick application. This is the Formula Atlantic racecar that was used during LogNet development.*

to have electronic driver aids such as ABS or traction control. But, it is useful to alert the driver that, while braking, the wheels lock up more than necessary or that the simultaneous application of throttle and brakes exceeds a set threshold and could cause premature brake fade. Or even worse, the throttle is slow in returning to idle, indicating a potentially disastrous throttle-stuck condition.

The operating environment can best be described as hostile. Keeping electronics, especially microprocessors, functioning is tough when competing with intense EMI and RFI noise.

In the early tests, getting closer than 3′ from a car with the engine running caused the prototype system to freeze. After implementing the watchdog timer in all microprocessors and adding all known means of suppressing noise, we were able to install the system and start the engine without instant failure. Running the car on the track added another level of error detection and recovery. In the end, the firmware was implemented and debugged to deal with errors and error recovery before all the network main functions were installed.

Val's wife (who drives a Cosworth-powered Sports Racer in the ACRL pro-series) drove the LogNet-enabled car during development so we could reprogram the microprocessors between track sessions (see Photo 2). During development, an internal WDT counter (which counts how many times the watchdog timer restarts a node) revealed that the system occasionally froze. The radio push-to-talk button status is transferred over the network, so the driver knew when the network stopped because she noticed that the crew didn't respond to her call. After the protocol's fault-tolerance was implemented, the system operated without losing data and the driver observed continuous operation of all functions.

The low-power consumption is also appreciated by the battery, which uses a weak alternator so it doesn't rob the engine of too much power. Running a six-wire (two data-line pairs, power and ground) cable is a huge benefit given the extreme space and weight constraints in the racecar. Also, implementing the UART protocol for data download and telemetry is fault-tolerant. Disconnecting the UART cable stops communications. Reconnecting the cable resumes communications with no data loss.

The racecar application posed a challenge in terms of power, space, and a noisy environment that perhaps can be found in other embedded applications.

Val Popescu, the founder of ProDATA Microelectronics, spent more than 10 years as president of Metaflow Technologies, the company credited with inventing the modern version of out-of-order execution architecture. Gary Gibson worked with Val at Metaflow and was in charge of engineering. Both Val and Gary switched careers from the high-end 32-bit multimillion-transistor microprocessors to low-end networks implemented with an 8-bit microcontroller. You may reach Val at val@prodatam.com.

SOURCE

PIC16C505
Microchip Technology
(888) 628-6247
(480) 786-7200
Fax: (480) 899-9210
www.microchip.com

The Shocking Truth About EMC

Part 2: Practical Application

FEATURE ARTICLE

George Novacek

Having covered the principles of EMC in Part 1, this month George sets out to show us how to keep everything in mind when designing electronic equipment. If you design electronic equipment, here's a handy ounce of prevention.

An overview of the aspects of electromagnetic interference and immunity was presented in Part 1 of this article. Part 2 will present practical ways of designing electronic equipment to successfully handle these aspects.

GROUNDING AND POWER DISTRIBUTION

The importance of proper grounding can't be overstated. The principles are straightforward, but the implementation may cause headaches. Figure 1 shows the arrangement of a typical small system with three separate controllers.

The same grounding and power distribution principles apply whether the system shown represents an aircraft with black boxes, an electronic rack with plug-in modules, or a PCB layout. I'll summarize the fundamental rules in this article.

Each module has a single point where all the internal grounds (power returns) converge. No power return can be shared with other modules, therefore, analog and digital circuits have separate grounds. This single point connects to the chassis.

EMI filters and transient protection devices return current to the chassis. Externally, power and power returns are not shared. This equally applies to the external I/O connections, which should have their own returns.

Chassis ground provides safety (EMI and lightning strike return), not for power or signal return. However, this rule is often violated. Cable shielding is terminated to the chassis.

And lastly, the internal grounds must be as low impedance as possible, preferably ground planes. Grounding (bonding) connection between the individual units and the chassis ground must be less than 3 mΩ and maintain low impedance well above 20 GHz. Numerous methods are used, such as woven copper straps and grounding fingers.

Figure 1 also illustrates the headache that driving external devices can cause. The problem is that you rarely have control over the selection of external devices and their installations. For example, cars use chassis for power return from solenoids, relays, control switches, and such. Some loads may have their own power return, such as shown with the dotted line in

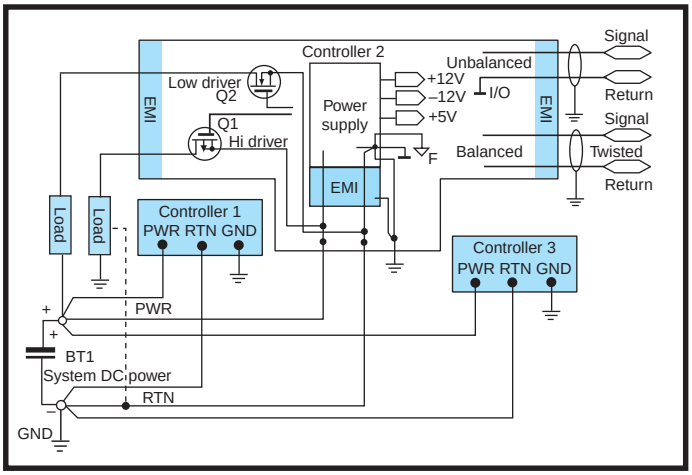


Figure 1—*The wiring of a small system with distributed processing is shown here.*

Figure 1. Whether a low-side or a high-side load is driven, it has a tendency to introduce unwanted current paths. You must carefully identify and control these paths or they will create ground loops and degrade the performance of the expensive EMI protection scheme.

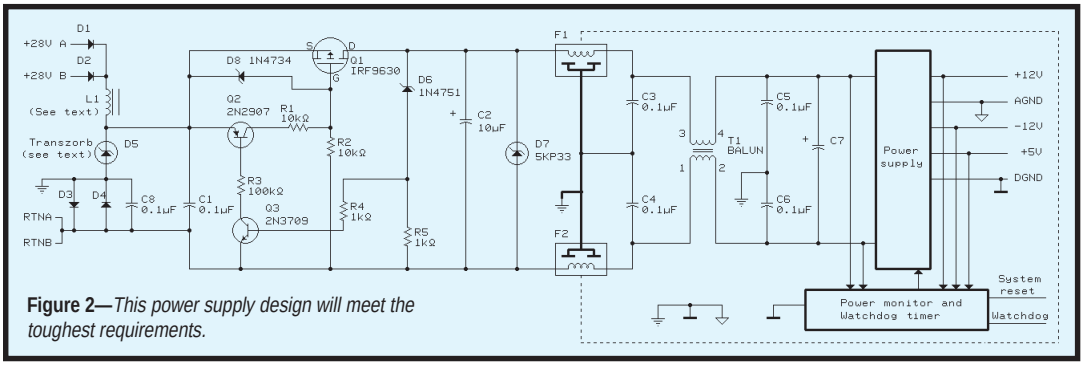


Figure 2—*This power supply design will meet the toughest requirements.*

POWER SUPPLY

Let's start discussing the inner design by focusing on the power supply. Not long ago, with the abundance of three-terminal linear regulators, designers started treating power supplies as an afterthought. Yet, there is more to the efficient, reliable power supply than the regulator. Although getting an inexpensive, reliable linear or switching regulator to satisfy most requirements is nearly hassle-free, the power-supply module remains the critical part of the design. The power supply in Figure 2 will stand up to the toughest requirements.

At the core of this module resides a purchased switching regulator (I discovered it's seldom worth designing your own). In demanding applications, a dual-cavity design will be used. The box around the regulator and two feedthrough filters depicts this. The regulator is inside the clean cavity, with the conditioning circuits outside in the dirty cavity. Commonly, the regulator will be packaged in its own sardine can to minimize its radiated interference with internal circuits. In less-demanding applications, some parts can be eliminated, such as the clean cavity, filters F1 and F2, and the pre-regulator represented by Q1 and surrounding circuits.

I'll describe the circuits to clarify the design philosophy. First, the reason for two 28 VDC and corresponding return terminals lies in critical systems' requirement that a power interrupt of up to 1 s doesn't affect the unit's operation (remember the info from Part 1 for Category Z, *Circuit Cellar*, 117, p. 29). In many cases, the necessary storage capacitor would never fit within the equipment envelope. Therefore, aircraft systems use two independent power buses, relying on the probability that an outage will not happen in both simultaneously. The buses are ORed through diodes D1 and D2 to prevent propagation of one bus fault into the other.

Although simple, this method's drawback makes it unacceptable in critical applications. If one of the diodes fails, the fault will remain undetected or dormant until its associated power bus goes, and then the entire system fails.

Notice that the power return flows to the switching regulator in the clean cavity without being anywhere connected to the chassis or internal ground. The switching regulator is fully isolated from its secondary, as is required in sensitive systems where no less than a 10-kΩ resistance between the chassis and the power return is permitted to minimize ground loop currents. For such systems, the isolated switcher is the only practical solution. In less demanding systems where DC resistance can be low as long as a sufficient impedance exists at high frequencies, capacitor C6 is replaced with a short, connecting the return from the regulator to the ground. All the grounds and power returns converge there.

Diodes D3 and D4 connected between the power return and the chassis make sure the return will not swing more than one diode drop away from chassis ground when exposed to a strong electric or magnetic field, or lightning strike. When sizing diodes D1, D2, D3, and D4, remember that they will have to bear the full brunt of lightning transients, which can be 1600 A forward and several thousand volts in reverse bias. Five ohms is considered representative of the lightning source as seen by the electronics.

Transients on the 28-V power line are clamped by D5 to approximately 100 V. Depending on the level of threat the power supply must survive, D5 may have to be replaced with a gas discharge tube (spark gap). L1 can be replaced in less demanding systems with a small resistor. The series impedance is needed to limit the transient current through the clamp D5, but the voltage drop over a resistor may present an operational problem. Consider the lowest operating voltage, subtract the voltage drop across D1 (or D2), L1 (or a resistor in its place) plus Q1, and it may be difficult to select an off-the-shelf switcher that will work through the entire operating voltage range.

The following circuit comprising Q1, Q2, and Q3, is a series pass preregulator that helps the power supply smooth over the 80 V, 100 ms

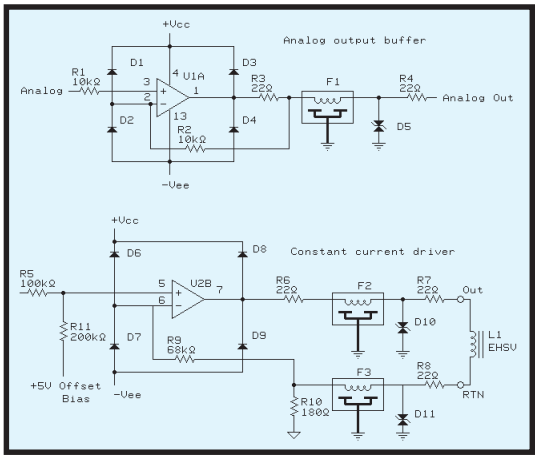


Figure 3—*Analog signal drivers are pictured here.*

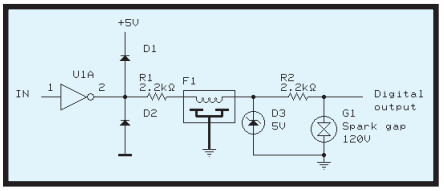


Figure 4—*A digital driver has many similarities to its analog counterpart.*

long transient as required for Category Z equipment (similar to MIL-STD-704 military requirement). Q1 is maintained fully conductive for input voltages below 32 V and clamps everything above back to 32 V. Tranzorb D7 on its output, clamps remnant transients and protects the expensive switching power supply in case of a preregulator failure.

The preregulated supply voltage enters the clean cavity through two low-pass π feedthrough filters. Their primary purpose is to cut down interference generated by conducted emissions of the switching power supply. But, filter size is inversely proportional to its critical frequency. To keep emissions under control in a small envelope, a supply with a 500 kHz or greater switching frequency is needed. Good grounding and ample ground planes are indispensable.

Balun T1, in the clean cavity, minimizes common-mode interference generated by the switcher and conducted to the outside world. Using it will bring conducted EMI under control. Finally, the function block called "Power monitor and watchdog timer" can be built with one or several MAX chips. It consists of several voltage comparators, which shut down or reset the system when input or any output voltage goes outside their predetermined limits.

The watchdog timer is a standard feature of many monolithic system supervisors. It is reset continuously by the microprocessor as it executes code. If the micro freezes, theoretically the timer won't restart within its usual time interval of several milliseconds, and will attempt to revive the program by forcing reset. The feature is usually included for free, so there is no reason not to use it. However, the watchdog timer's reliability of operation is not acceptable for critical systems or as the sole monitor.

ANALOG OUTPUT DRIVERS

A control system often drives analog loads. Figure 3 shows interfacing analog loads. The analog output buffer will work well where output impedance of approximately 22 Ω is acceptable. Reduce the series load and implement a two-stage protection (see Figure 4) for greater loads.

The output resistance is determined by R4, which limits the current into bidirectional tranzorb D5. Tranzorbs are like Zener diodes, but faster and capable of dissipating higher pulse loads. They are available as unidirectional, analogous to a single Zener, and bidirectional, which can be viewed as opposite-series connection of two Zeners.

An inductor can replace the signal voltage drop across R4 if it's too high, as long as the working bandwidth is retained. Resistor R3, also in the signal path and usually of the same value as R4, does not contribute to the voltage drop because it is compensated by the negative feedback introduced by R2. R4's purpose is to limit the current into the IC output drivers if the voltage exceeds (plus or minus) the supply rails. D1, D2, D3, and D4 make sure no pin voltage exceeds the supply rail by more than one diode drop. Filter F1 is a low-pass π filter, which in demanding systems is a feedthrough between the dirty and clean cavities.

The resistor described here limits the current that can flow through the transient protection device, whether it is a tranzorb or a spark gap. Note that the popular metal film resistors are unsuitable for this purpose because they can't withstand high voltage. Because the carbon resistors are nearly extinct, wire-wound devices are practically the only choice.

The constant current driver also shown in Figure 3 is typical of a balanced output driver using the same design elements as the unbalanced driver above it. The driver takes analog 0 to 5 V, centered at 2.5 V from a digital-to-analog converter (DAC) and puts out ±10 mA current into a load from 0 to 1000 Ω. This would be a typical interface for an electrohydraulic servo valve (EHSV).

Before leaving the subject of analog drivers, revisit Figure 1 and remember that the signal returns of the unbalanced outputs and unbalanced inputs need their own returns. They would be treated the same way as the RTN line in the Constant Current Driver (see Figure 3), except that D11 would be replaced with two opposite parallel diodes (such as D3 and D4 in Figure 1), and R10 would be replaced with a short to the appropriate signal ground.

DIGITAL OUTPUT DRIVERS

There is not much difference in driving digital loads, as in driving analog loads. The elements of the circuit shown in Figure 4 are similar to the analog drivers in Figure 3.

Here, a spark gap is shown as part of the two-stage protection of the output line. Why are they necessary? It's a combination of the maximum threat that must be protected against, the energy a tranzorb can dissipate, and the maximum value of the series resistor that can be used based on the load requirements. In this case, I used 2.2-Ω series resistance. An inductor in place of R2 would adversely affect the signal bandwidth. Because the output line must survive a lightning transient of 1600 V/1600 A, the spark gap, which fires when the voltage reaches about 120 V and then clamps it and holds at approximately 24 V, handles the responsibility. The wiring resistance provides the current limit for the spark gap. The tranzorb in the second stage looks after the remnant pulse, with the filter cutting off the remaining narrow pulse resulting from the finite time the clamping devices need before they conduct.

Remember, once spark gaps fire, they keep conducting in the presence of a DC voltage greater than 24 V. This may present a design challenge when protecting a power supply.

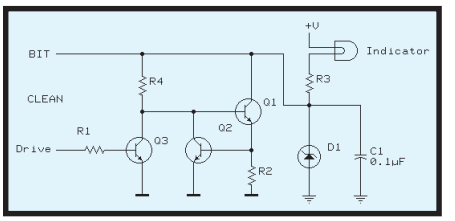


Figure 5—*Interfacing a panel indicator with the controller is more complicated than it appears.*

Why use clamps (such as tranzorb D3 in Figure 4) and still use signal diodes D1 and D2 to protect the IC pin? In many applications, the tranzorbs are sized to allow full swing of the IC output voltage. In the example above, this would be 5 V. A 5-V tranzorb, due to tolerances and poor transfer characteristics, will clamp the incoming transient caused by ESD, lightning, and such to approximately 7 V or 1 V (depending on the polarity of the transient). This is high enough to damage the IC running at 5 V_{CC}. You may decide to rely on the internal ESD protection existing in many of today's ICs. On the other hand, if the design robustness is important to you, for a few more pennies, add the two signal diodes.

INDICATOR DRIVER

So far, the design has been straightforward, more interesting circuits are coming up. Many controllers have remote indicators such as panel lamps. Driving them properly is complex despite apparent simplicity. Figure 5 shows a typical implementation of such an indicator circuit, with features applicable to all the driver circuits discussed here.

This is a low-side driver with several atypical twists. For standard applications, multiple monolithic driver packages with internal overcurrent protection and built-in-test (BIT) are available. The external interface, R3, D1, and C1 are the same as in the previous circuits. They clamp the incoming transients to a safe level that is determined by D1. The low-pass filter isn't here, but in a two-cavity design, the reliable π feedthrough would be inserted between R3 and Q1. Placing the driver circuit in the dirty cavity (depending on the power return requirements) and filtering the low signal level BIT and Drive lines entering the clean cavity is another option.

R2 and Q2 perform current limiting for short circuit protection of the driver. The BIT signal can be used to disable the driver if an overcurrent condition occurs. Q1 must be able to dissipate the short-circuit power (full supply voltage × maximum current is equal to 0.55V/R2) indefinitely, or for the period of time it takes the microprocessor to energize Q3 and remove drive current from the base of Q1 if BIT is used to disconnect it. Or, Q2 may be replaced with a latch.

Note that this fault indicator must be illuminated when the controller is not functioning, including when its power is off. The driver is energized through R4, thus the indicator lamp is illuminated by default. This is why bipolar transistors with merely 0.55-V base-emitter voltage are better suited than a MOSFET. The controller keeps Q1 de-energized like a dead-man's switch through Q3.

BIT monitors the status of the driver in response to the drive commands. Therefore, it detects problems like a burnt or missing lamp or short circuit.

The momentary overcurrent situation which occurs when driving an incandescent lamp is a common trap of this application. When the lamp is turned on, its onrush current exceeds (for a short period of time) 10 times the nominal current value. The BIT circuit and especially power-up diagnostics must allow sufficient time, approximately 100 ms, for the lamp current to settle down before reporting a problem or turning off the drive current. For Q1 to function properly, it must be rated to survive full short-circuit dissipation for that length of time.

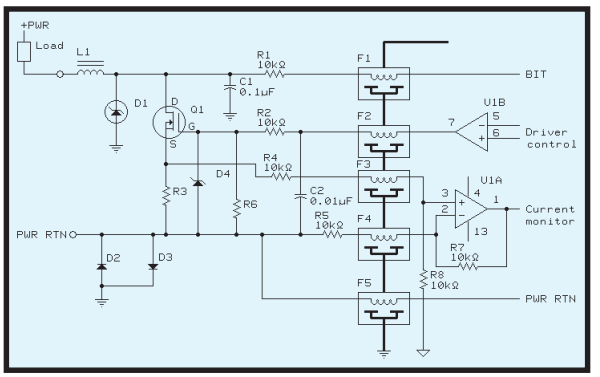


Figure 7—*This is a standard low-side driver with overcurrent protection.*

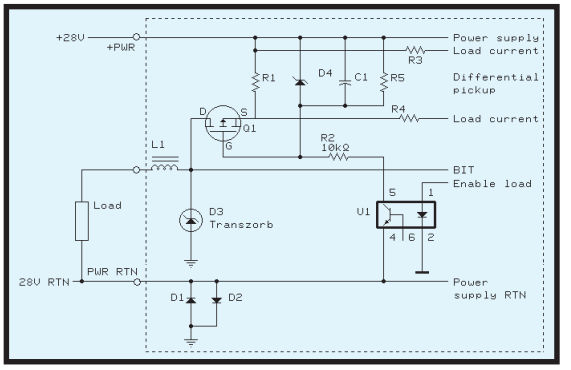


Figure 6—*High-side drivers are more difficult to implement than low-side drivers, yet they have some important advantages.*

The next section will analyze power drivers and crucial design issues of power and return. The same considerations given to the power drivers must be given to all external load interfaces.

HIGH-SIDE DRIVER

High-side drivers use a few more components than their low-side cousins, but are popular with system designers for two reasons (see Figure 6).

First, some systems (e.g., automotive) use a chassis for the load power return. Second, when wiring fails it is likely that the connection will break or the line will short to the chassis. This means MOSFET Q1 (short-circuit protected) can't energize the load, but often this is the preferred default in case of a failure. The same fault occurring to the low-side driver will cause the load to be energized continuously. This failure mode may not be acceptable for safety reasons.

The circuit in Figure 6 uses an optocoupler to drive power MOSFET Q1. The return drive current path would have to go through chassis ground in a system with an isolated switching power supply (see Figure 1). This is possible, but not a good design practice as far as EMI is concerned. Alternatively, isolation through resistors >10 kΩ, especially when a non-isolated switcher is used, is a good solution. Isolation amplifiers such as the ones manufactured by Burr-Brown are options.

Similar to other circuits, the load works in series with choke L1 to limit the transient current through tranzorb D3.

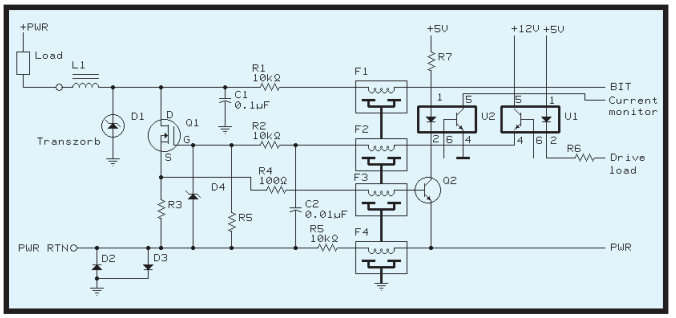


Figure 8—*Using optocouplers will reduce the number of components and provide the highest degree of isolation.*

R1 connected between the power bus and the source terminal of the power MOSFET monitors the load current for overload protection. As a result of the supply voltage variation between 16 to 32 V, voltage developed across R1 needs to be monitored in a differential manner to suppress the common-mode signal. In addition, an acoustic signal immunity test (a standard requirement) superimposes about 2-V RMS ripple of up to 50-kHz frequency on the positive voltage power bus. Resistors R3 and R4 should be at least 100 kΩ each to have minimal effect on the power isolation.

For high-power switching such as 28 V/30 A that some electric actuators require, the selection of P-channel MOSFETs is limited. Sometimes it is better to use an N-channel device and monolithic charge pump to generate the needed 10 V higher than the 28-V line to drive it.

LOW-SIDE DRIVERS

Low-side drivers referenced to the ground are simpler to implement than high-side. The latter's disadvantage is an inability to disconnect the load during a wiring harness short. But, often that's acceptable.

Like high-side drivers, low-side drivers are complicated when maintaining the power isolation needed to prevent ground loops. In Figure 7, the required isolation is degraded only by 10-kΩ resistors R1, R2, R4, and R5.

The problem these circuits face is that the power return potential may be different from the common internal ground. The solution is simple when using a power MOSFET for Q1. Diodes D2 and D3 ensure the PWR RTN excursions around the ground do not exceed one diode drop. This has no effect on driving the Q1 gate, but would be difficult to handle if Q1 was a bipolar transistor. Make sure the load-current sensing performed by R3 is not affected by the return line noise. Use differential voltage pickup by amplifier U1A.

The power drivers usually are located inside the dirty cavity to limit the need for low-pass filters to the signal lines only. Figure 8 shows another approach to driver circuit by using optocouplers. This maintains complete isolation from the primary power bus. The BIT line can be interfaced through an optocoupler, as well.

An optocoupler also can perform the overcurrent monitoring. It can rarely be used alone because of the LED's high forward voltage. In Figure 8, Q2 fixes the problem. Most applications don't need current measurement, detection of exceedance of a threshold usually is enough. In addition to standard current transformers and shunt resistors, new current monitoring devices, such as the ones based on Hall-effect diodes appear daily.

TOTEM-POLE DRIVERS

Although not directly related to electromagnetic compatibility design, a review of output drivers wouldn't be complete without the totem-pole topology. These circuits comprise several high-side or low-side drivers described here as building blocks. Figure 9 illustrates the principle of the full totem-pole driver used in flight-control systems where the design requirement states that "no combination of two faults shall allow inadvertent energization, or prevent de-energization of an actuator."

Even with any two MOSFET drivers shorted, the system can de-energize the load. Controllers using these drivers are at least dual redundant, meaning two parallel processing channels are in action. Triple redundancy would require three series MOSFETs in each branch.

More often this topology is simplified to use only two low- or two high-side transistors in series together with dual-redundant processing.

CONTROLLING AC LOADS

With the increasing selection of power semiconductors such as thyristors, IGBTs, enhancement MOSFETs, and bipolar devices, relays are no longer the only choice for large AC loads. Assuming reader familiarity with the operation of thyristors, I'll concentrate on the EMC aspects.

Figure 10 shows an actual high-power switch in a safety-critical application. Focusing on the EMI issues, the major problem lies in suppression of emissions to the levels satisfying standards such as MIL-STD-461 or DO-160D. A zero-crossing trigger renders the best EMI performance.

This circuit is a reliable design, switching 200 V, 400 Hz, 3-phase 20 kVA to a furnace. Optocoupled triac U1 with internal zero-crossing detector (Motorola MOC 3083) is the reason behind its simplicity. The EMI

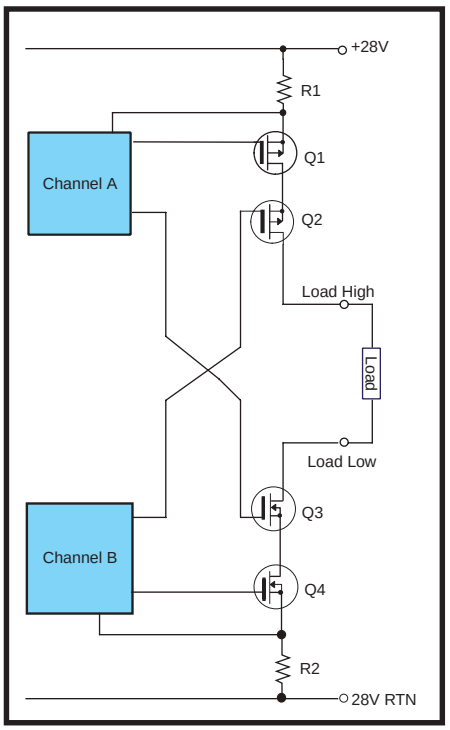


Figure 9—*No two failures of the power semiconductors will prevent the load from being de-energized.*

performance is on par with more conventional and presumably more precise (in terms of zero-cross triggering) circuits using pulse transformers. Its cost and complexity are lower. Because it drives a purely resistive load, it doesn't need a snubber, which would be a series RC network placed in parallel across the thyristors.

The current transformer CT1 performs load monitoring and overcurrent protections. The power circuit is located in the dirty cavity with low-pass filtered interface for control and current monitoring. With the thyristors rated at voltages and currents exceeding the lightning-induced threat, no clamp or other transient protection is needed.

Be prepared for the conducted emission of this circuit to be a problem in the 50- to 200-kHz range where it exceeds maximum levels allowed by MIL-STD-461 by approximately 12 dB. On the 200-V power line, this represents less than 20 mV of the unwanted RF signal, which is lower than the conducted interference levels commercial applications worry about. But, there is no practical solution to reduce this emission.

Thyristors need anode voltage before they can fire so that the latching current can flow and be maintained, therefore triggering at true zero-crossing is impossible. A delay of a few degrees of phase after zero-crossing is needed for sufficient voltage to build up across the device. This voltage is

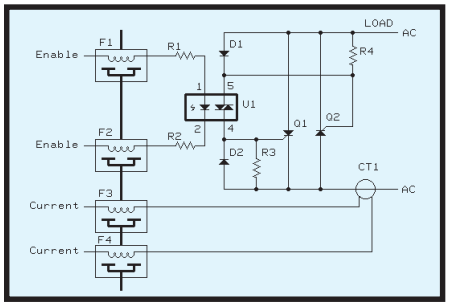


Figure 10—*Driving AC loads is best performed by optocoupled thyristors.*

greater than the anode voltage required by the optotriac, so performance isn't sacrificed by using it. Still, the residual anode voltage of about 6 V peak across the thyristors contains sufficient harmonics to exceed allowed emission limits. A filter with critical frequency low enough for effective attenuation in the offending frequency range while carrying a 100-A load current is impractical to build within a reasonable envelope.

RELAYS

As discussed, relays will be energized by low- or high-side drivers. Internal relays switching external signals should be placed inside the dirty cavity with tranzorbs to suppress contact sparking, MOVs, and RC networks to protect the contacts and reduce unwanted emissions. Usually, relay coils are equipped with a free-wheeling diode to suppress the back EMF generated by the collapsing magnetic field upon the coil de-energization. Unfortunately, the diode also will significantly slow down the relay's response. A combination with a Zener diode, capacitor, or resistor will improve the response time at the cost of a larger EMF kickback.

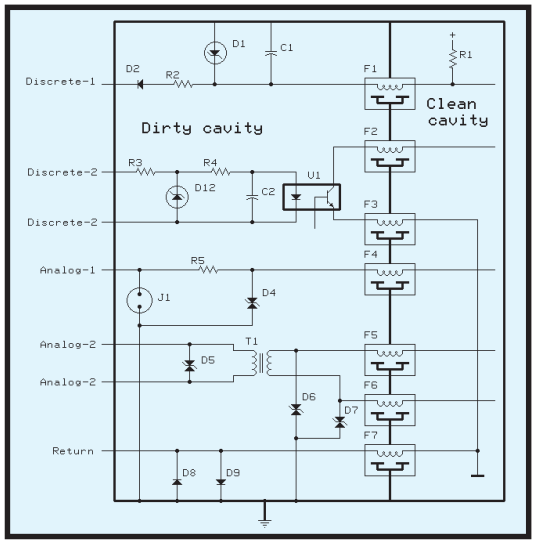


Figure 11—*A collection of input circuits is shown here.*

INPUT CIRCUITS

Figure 11 shows how the input circuits use the same principles for EMC and lightning protection as the output circuits discussed in this article. The circuits in Figure 11 provide adequate protection from EMI, lightning, and ESD at the highest threat levels. When dealing with the input

lines, recall using signal diodes to protect IC pins by clamping their voltages to within one diode drop of their V_{CC} and V_{EE} for output circuits. Use a similar approach here.

Examples will clarify the process. The input line labeled Discrete 1 is a typical active-low input circuit (invert polarities for active-high). R2 provides transient current limiting for tranzorb D1. R1 is a pull-up resistor that will be needed even when the input source does not require one, to ensure known state when nothing is connected to the input. A standard requirement is that discrete lines have a minimum 3-V noise margin. To achieve that, R1, R2, and the following circuits must be carefully selected. Figure 12 shows the different situation when interfacing electromechanical switches.

Ideally, each discrete input should have its own return line, or they should join at the same connector pin, labeled Return in Figure 11. This is not always practical. Many systems use a chassis for return. This will not be a problem with discrete lines as long as the 3-V margin is maintained.

Although tranzorbs are excellent for dissipating high current pulses, depending on the size of R2, they may not withstand sustained short of the input line to, for example, 28 V. Some input sources may have their own pull-up that could turn into a power source for the de-energized controller. Hence, D2 allows the input current to flow in a single direction only. Make sure that the diode takes the reverse voltage and the forward current induced by the lightning transients, and also is fast enough for the signals.

Discrete 2 is an optoisolated interface that cannot be beaten for its simplicity, ruggedness, and isolation properties. Tranzorb D3 limits the maximum current U1 LED will be exposed to and protects it against reverse polarity. The optoisolator's excellent isolation properties guarantee that the input can be left completely floating if desired. Or, its return line can be tied to the chassis through the opposite-parallel diode combination such as D8 and D9 in Figure 11.

When you need to interface an analog device such as an LVDT, the approach labeled Analog 1 may be the way to go. Because R5 can't be large enough to limit the transorb D4 current sufficiently, the two-stage protection is used. Analog 2 uses a transformer for isolation. Similar to the optocoupler interface, it can be left floating or tied to the chassis through opposite-parallel diodes. Both the optocoupler and the transformer input are hard to beat when large common-mode input swing or great circuit isolation is needed.

Figure 12 shows the most effective, yet simple interface for electromechanical switches and relay contacts. Mechanical contacts need a minimum current in the range of 10 mA (usually more for the tin-plated type), therefore the interface, comprised of Q1, R1, R2 and R3, is essentially a current-to-voltage converter. A series-blocking diode can be added in series with R1, like in Figure 11, although it is rarely justified. The 1-kΩ R1 resistor (no metal film!) limits severe transients well below the peak pulse the tranzorbs can safely take.

R4 and C1 debounce the signal and pass it to a CMOS Schmitt Trigger circuit operating off the internal logic supply. Signal diodes D2 and D3 and R5 ensure the signal at the IC input doesn't exceed safe limits.

Before concluding the subject, I want to emphasize two important aspects of designing EMI/HIRF lightning protection for the I/O lines. First, performing rudimentary calculations presented in Part 1 determines the worst-case interference signal levels that can reach the system. As long as the high-frequency interference signals are reduced to less than one diode drop (set the limit to 100 mV at the highest), there is little chance they will interfere. Shielding, filtering,

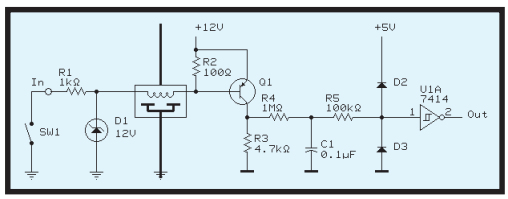


Figure 12—*Preferably, interfacing electromechanical switches is accomplished with current, rather than voltage, interface.*

layout, and packaging achieve the goal. As long as the interference is below one diode drop, the internal circuits' operating bandwidth is limited and outside the interfering signals, the interference will be suppressed and not affect the operation. RF susceptibility usually is the result when the interference signals exceed the base-emitter junction voltage of bipolar devices. The result is rectification, demodulation, and such playing havoc with the equipment.

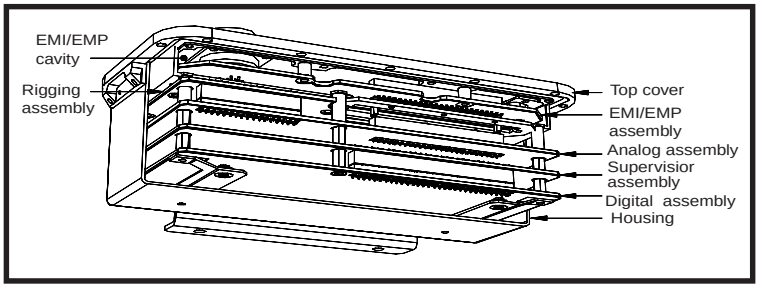


Figure 13—*Here's a self-contained controller with a front panel holding connectors and controls. The metal tub that forms the dirty cavity is attached to the front panel from below. All connections between the clean and dirty cavities pass through feedthrough low-pass filters installed in the bottom of the tub.*

Second, design of the transient suppression can have a similar negative effect. While designing the clamps to protect the microelectronics against damage, recognize that the working environment isn't static, but rather with pulses and bursts of RF. Once the clamping voltage is exceeded, the pulses or induced RF bursts will be rectified and result in a DC offset or a demodulated low-frequency signal that will be introduced into the system. Be careful selecting unipolar clamps. They may appear to be the correct choice for TTL signals, but their asymetrical characteristics will be responsible for rectifying interference signals at the controller input as low as one diode drop. This can be troublesome when considering the system's behavioral response to lightning. Repeated bursts of high-frequency oscillations are common.

There are several things that can bring such a situation under control. First, the clamps should be selected at the highest safe voltage to prevent rectification from occurring as much as possible. Ensure the discrete inputs have sufficient noise margin; the analog circuits need large, dynamic range coupled with the common-mode input range and common-mode voltage rejection. And, validate the input signal so that if it is obliterated by interference, the system recognizes it and takes appropriate action (e.g., shuts down). Optocouplers and transformers still are the most effecient tools for bringing EMC under control.

CABINETS

Well-designed packaging is an important step towards electromagnetic compatibility. Thanks to continuing miniaturization, multilayer PCBs, and SMT, many commercial products achieve unprecedented EMC performance without special packaging effort. But, when it comes to operation at e-fields exceeding 200 V/m and satisfying the most stringent emission requirements, the dual-cavity design still reigns supreme. Figures 13 shows an example of a dual-cavity cabinet.

I included a reference list regarding the art of electronic packaging design for interested readers.

This two-part article covered aspects of designing electronic equipment for electromagnetic compatibility. In today's environment of increasingly abundant electrical pollution, engineers have a responsibility to design equipment not only immune to such overabundance of interference, but also equipment that will not add any new problems.

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

RESOURCES

W. Boxleitner, *Electrostatic Discharge and Electronic Equipment*, IEE Press, 1989.

F.A. Fisher and J.A. Plumer, *Lightning Protection of Aircraft*, Lightning Technologies, Inc., Pittsfield, MA, 1990.

G. Fuller, *Understanding HIRF*, Avionics Communications Inc., Leesburg, VA, 1995.

O. Hartal, *Electromagnetic Compatibility by Design*, R&B Enterprises, West Conshohocken, PA, 1991.

M. I. Montrose, *EMC and the Printed Circuit Board*, IEEE Press, 1998.

G. Novacek, *Testing 1, 2*, Circuit Cellar Online, www.chipcenter.com, July–Oct. 1999.

R. Standler, *Protection of Electronic Circuits from Over-voltages*, John Wiley & Sons Inc., New York, NY, 1989.

L. Tihanyi, *Electromagnetic Compatibility in Power Electronics*, IEEE Press, 1995.

D.R.J. White and M. Mardiguian, *Interference Control Technologies*, Don White Consultants Inc., Gainesville, VA, 1986.

SOURCES

DO-160D
RTCA, Inc.
(202) 833-9339
Fax: (202) 833-9434
www.rtca.org

W181 Peak Reducing EMI Solutions
Cypress Semiconductor
(408) 943-2600
Fax: (408) 943-6848
www.cypress.com

FOR MORE INFORMATION ON DESIGNING AND TESTING EQUIPMENT, CHECK OUT GEORGE NOVACEK'S FOUR-PART SERIES "TESTING 1, 2" IN THE CIRCUIT CELLAR ONLINE ARCHIVE

@

WWW.CHIPCENTER.COM

Weather Data

FEATURE ARTICLE

William Beals & Russell Chadwick

Getting More Than One Wire's Worth

Russell and William rolled a weather station from Dallas and a Motorola 6808-based electronic interface display unit into an award-winning project in last year's design contest. And now, here's an inside look at how the project came together.

The weather is a common conversation topic. Now, you can build a weather station that will provide plenty of weather data to discuss. Unlike commercial stations, our approach is less expensive and eliminates the need for a dedicated computer.

In addition to acquiring weather data and displaying it locally, we interfaced the weather unit to amateur radio so the weather data can be sent via airwaves and the Internet. This involves a new and growing part of amateur packet radio called Automatic Position Reporting System (APRS).

Weather data is what interests us here, but information like position data, velocity, and heading data, and telemetry are also carried via APRS signals. Digital repeaters (digipeaters) relay these information packets to extend coverage areas. Some digipeaters function as Internet gateways (I-gates) that relay data to a central server. The packets are collected and sent via the Internet to anyone logged onto the server. This configuration is deceptively simple, yet powerful.

A single I-gate can collect and relay weather information from dozens of weather stations, no matter how remote. If you have a Java-enabled web browser, visit www.aprs.net/usa1.html to see what the coverage looks like in your area. To see the weather data collected by amateur radio operators, go to www.findU.com/wx. To see our data, use calls n0xga for William (Denver, CO) and kb0tvj-5 for Russell (Boulder, CO). More information on this new aspect of ham radio can be found at www.tapr.org.

If you want your data to be available on the Internet, in addition to this project, you will need some other items. The first is an amateur radio license like the technician class license. It has no Morse code requirement, and with it you have access to all amateur bands above 30 MHz, including the 2-meter band (144–148 MHz), which has most of the APRS activity. You can get more information on amateur radio licenses from www.arrl.org.

You will also need a TNC (Terminal Node Controller) and a VHF radio. The data side of the TNC connects to one of the COM ports of your computer, and the analog side connects to the microphone input and speaker output of the radio. The TNC transfers data back and forth between the computer's digital world and the radio's analog world.

BUILD VERSUS BUY

Professional quality weather stations cost several thousand dollars, but there are citizen weather stations on the market priced from $300 to $700. However, we wanted to build a high-quality unit that was also inexpensive. Now, there's a challenge! We decided to buy the sensors that measure the parameters (wind speed, wind direction, temperature, pressure, rain, and humidity). But, we built the data interface to acquire data from the sensors and then process and display the weather information.

The solution was around half the cost of the lower-end citizen weather station, and needs no PC. It is entirely digital and can run from a wall transformer or solar power. Only one DC voltage is needed to run the entire project. Because we were in control of the software, we used data processing algorithms to improve the quality of the measurements and reduce error.

THE 1-WIRE UNIT

As we shopped around for a sensor unit, Dallas introduced a promotional weather sensor unit for $79 to demonstrate one of their product lines. They have a family of digital sensors and components based on a bus technology called 1-Wire. The sensor unit will be available until at least August of 2000 and is described at www.ibutton.com/weather/index.html.

The wind speed is measured with a rotating cup anemometer, which is assembled with a shaft connected to the center of an arm with small magnets on either end. Two magnets are used to balance and stabilize the arm, which turns 2400 rpm in 100-mph wind! The magnets pass above a reed switch and cause a switch cycle for each pass. The switch is connected to a 1-Wire counter chip mounted on the circuit board inside the housing. The software samples the counter value at regular intervals and converts the counts to wind speed.

We tested the process in a calibrated wind tunnel to determine the conversion from count number to wind speed. Our tests showed that the anemometer responds to winds of about 1.5 mph.

A wind vane is connected to a magnetized arm that moves over eight reed switches spaced 45° apart around a circle. Either one or two switches are closed depending on whether the magnet is above one of the switches or between two of them. The processing software determines which switches are closed and then determines which of 16 equally spaced directions the wind vane is pointing toward. This is a clever way to get 16 directions with only eight switches.

The temperature is measured with a dedicated IC that gauges from –67°F to 257°F in 0.9°F increments and has a digital output. The digital output measurement is inherently more immune to interference and line loss.

THE 1-WIRE BUS

1-Wire technology is a neat way to provide chip power and two-way communications capability over the same wire. There is another wire that runs along with the 1-Wire and serves as ground. The 1-Wire signal and ground form a multidrop bus. The bus can be up to 1000′ in length with CAT 5 network cable. You can hang an arbitrary number of devices on this bus and add sensors and other equipment after your initial hardware design. All you have to do is update your software to include the extra chips.

Each of the 1-Wire chips has a unique serial number to identify it. The first step of any 1-Wire bus driver is to interrogate the chips to determine serial number, quantity, and locations.

The communication scheme is similar to PWM. The idle state for this bus is a resistive pullup. The pullup powers the bus and the entire system is designed to consume little power—about 0.5 W from a 12-V supply (more than half of this power is dissipated in the 5-V regulator chip).

Data is transmitted on the bus using short pulses for 1s and longer pulses for 0s. The devices retain enough supplied energy to continue functioning during communications pulses. An advantage of this setup is that all the data traveling over the bus is digital. There are no analog signals going over long wires.

DISPLAY UNIT HARDWARE

The hardware for this project is a Motorola MC68HC908GP32, a smattering of interface support logic, and a 2 × 20 LCD display (see Photo 1).

Such a design can serve many purposes. A fully stuffed board connected to the 1-Wire weather sensor unit is a weather station, but there is nothing about the hardware on the microcontroller circuit board that makes it a weather station. The board also supports the free P&E programmer/debugger and an RS-232 interface. With this, you could run the existing code, modify and debug the code, or write a new 1-Wire application for a different sensor set.

The schematic for the display unit hardware is shown in Figure 1. All references to the 6808 manual refer to

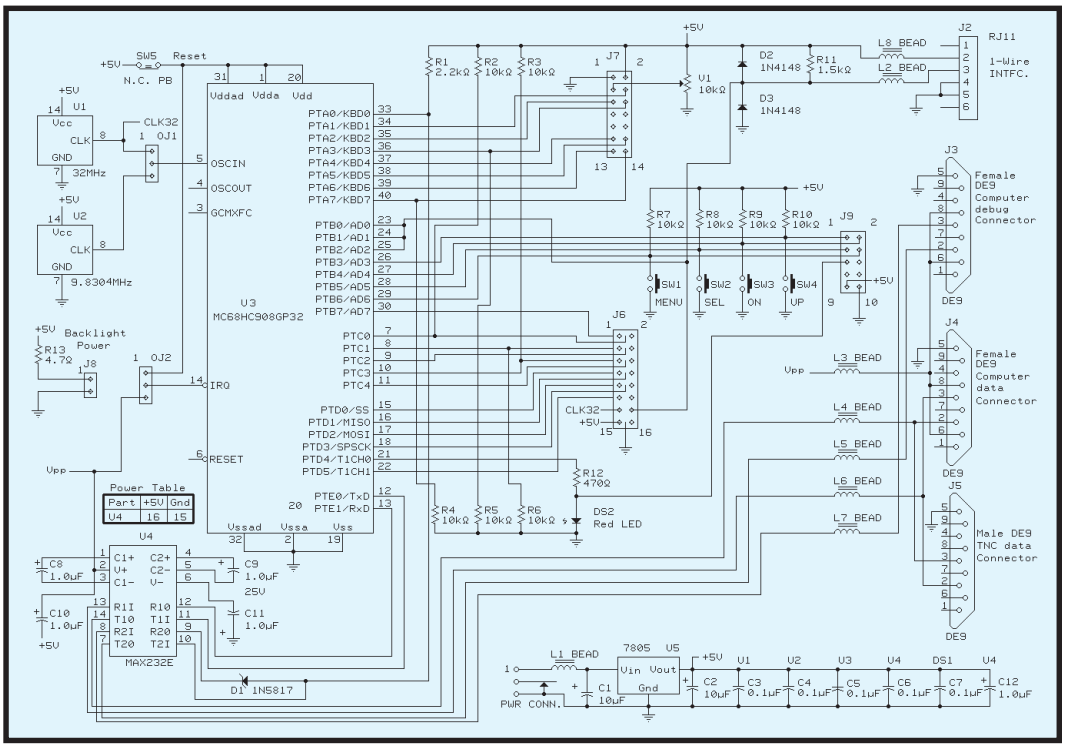


Figure 1—*Here's a look at the indoor part of the weather station.*

document number MC68HC908GP32/D Rev 2.0. Components SW5, OJ1, U2, and OJ2 form most of the debug support hardware. A normally closed switch, SW5 is used to interrupt power to the 6808. Removing power along with asserting a high voltage to the IRQ pin assures that the 6808 will come up in debug mode no matter what is in the flash memory. Jumper OJ1 and U2 select the needed frequency to run the 6808. A frequency of 9.8304 MHz is needed for debug mode, while U1 provides the normal mode clock source.

U4 and its support components are the RS-232 interfaces and V_{PP} generator. Two different serial ports exist. The main interface uses the 6808 TX and RX pins and is used to send and receive data to the TNC or computer. Male and female DE-9 headers exist for the data because we hate adapters!

A 9-V V_{PP} is needed to get the 6808 in debug mode. The voltage created by the MAX232E chip generates 9 V (which we used for the debug mode) for its RS-232 drivers.

Motorola's design guidelines for the debug interface are shown in figure 15-1 of the 6808 manual. They suggest a couple of tristate drivers to convert the normally separate TX and RX signals to the half-duplex signal connected to PTA0. We got the same result by replacing them with a single Schottky diode. A silicon diode won't guarantee a logical low to pin 9 of the MAX chip.

Resistors R1–R6 make sure various signals are in the right state during reset to keep either the LCD happy or to ensure that the 6808 correctly comes up in the debug mode when requested. The 6808 manual shows the various ways of getting into debug mode. We're using the second and third entries depending on whether or not the reset vector is programmed.

The LCD plugs into header J7 and, to save pins, we used the 4-bit interface. In addition to the standard 14-pin header, we also had pins for a backlight. Resistor R12 and DS2 provide a heartbeat indicator to show that the micro is running. J6 is an expansion header and switches SW1–4 are the main user input. Buttons are labeled Up, Down, Select, and Menu.

The 1-Wire interface starts with three 6808 I/O pins. We used one for readback and drove all three high when the 1-Wire bus needed to be driven with a low-impedance 5-V source. Diodes D1 and D3 provided static protection, while R11 provided the resistive pullup on the 1-Wire bus.

SOFTWARE

The software began as 6805 assembly code. For the basic 1-Wire drivers, we started from scratch using the 1-Wire bus spec and the example C code for items like CRC calculations. Early in 1999, we gained TAPR as a sponsor, who suggested the 68HC08 and mentioned the Design99 contest.

A quick review of the chip, the tools, and the contest rules showed that switching to the 6808 made sense. The port from 6805 to 6808 went smoothly, the only problem was not entirely understanding the ubiquitous role of the H register. All indexed instructions use H:X (it took a while for that to sink in). If you never use H, the 6808 instruction set is fully compatible with the 6805. The first time you touch H, you have to account for it everywhere in your code.

Since porting to the 6808, we made two major changes to the codebase. First, we added the loader, which allows someone without debugger hardware and software (and its requisite crystal) to download new application code into the chip.

It also provides a simple set of tools for the application to treat an area of flash memory for variable storage by providing erase and program routines. The loader only erases the application area, so any permanent variables are not erased when you load a new application. This is handy for the weather station because relearning sensor IDs requires manual positioning of the wind vane, which means hiking up to the weather sensor unit. The PC-side downloader program is currently a DOS program. We want to port it to Windows (any volunteers to write it for us?).

The second change was taking all the data gathering and reporting routines and making them entirely interrupt driven. We have plans for a more flexible user interface, so we moved the operation of the machine to the background with a 100-ms timer interrupt, trying to limit each interrupt to 10 ms or less of processing.

Of the 50 interrupts available in each 5-s sample period, we used only 16 for data gathering, statistics, and retransmission. Seven are for temperature acquisition, one for wind speed, four for wind direction, one for statistics gathering, two for transmiting the data, and one for post-processing.

The combination of flash memory, the downloader, and the extensible 1-Wire bus means that new weather station features are limited only by the amount of flash memory available.

ENHANCING ACCURACY

Data sent from the transducers down the 1-Wire bus has CRC words included. The software checks these and drops any erroneous data before processing.

Data that passes these CRC and limit checks is averaged to improve accuracy. The basic data acquisition period is 5 s and is used for local display. However, the data that is transmitted over packet radio is averaged over a period of 5 min. to improve accuracy and to avoid overloading the limited radio channel. Because the wind speed is calculated from the number of rotations of the anemometer in 5 s and then scaled, the averaging is a matter of counting the total rotations over 5 min. and divide by 60 before scaling.

The temperature data on the local display is the single reading taken every 5 s. The average temperature transmitted over packet radio is the average of 60 of these single readings.

Averaging wind direction causes a problem because of the crossover at north where the direction jumps from 359° to 0°. We use a consensus averaging technique that gets around this problem and gets rid of unrepresentative values at the same time. The 5-s direction measurement returns one of 16 directions, and the count of how often that value is measured, is stored in bins 1–16, each representing 22.5°. After the 5-min. averaging period and all the data occurrences are in bins 1–16, we create four new bins, 17–20,

Photo 1—*The entire display unit for keeping track of Mother Nature is available from TAPR.*

and copy the numbers from bins 1–4 into them as representative of directions greater than 360°.

Next, we find the five adjacent bins out of the 20 with the greatest number of occurrences. Then, find the mean value of the direction using the occurrences in these five bins. If the mean is greater than 360°, then subtract 360.

A problem with the 1-Wire weather station is that the temperature transducer is inside the sealed plastic unit, which can lead to high readings when the unit is in full sun. Our October to November tests showed an error of about 16°F in the full sun with calm wind. Averaging the data won't reduce this error, so we developed a ventilated solar radiation shield of thin aluminum that reduces error by about a factor of four in calm winds and more in higher winds.

FUTURE PLANS

An embedded processor with flash memory and a downloader means that you can improve the software without changing the hardware.

We plan to add data storage capability for logging long-term weather data offline, then we'll have the ability to download it to a computer.

William Beals is an electrical engineer for a satellite TV company in Denver, CO. He has been a ham radio operator for five years, lately enjoying the digital modes.

Russell Chadwick is with NOAA's Forecast Systems Laboratory. He has been measuring features in the clear atmosphere using electromagnetic techniques for more than three decades. He holds degrees in electrical engineering from Montana State, Wyoming, and Purdue and is a senior member of the IEEE.

SOFTWARE

All improved source code, object code, and the PC monitor program are available at www.ourworld.compuserve.com/webpages/wmbeals.

SOURCES

MC68HC908GP32
Motorola
(512) 328-2268 • Fax: (512) 891-4465
www.mot-sps.com

Weather station, iButton sensors
Dallas Semiconductor
(972) 371-4448 • Fax: (972) 371-3715
www.dalsemi.com

Display unit
TAPR
(940) 383-0000 • Fax: (940) 566-2544
www.tapr.org

FEATURE ARTICLE

Robert Kondner

Buying Power

A Low-Cost Power Supply

If you've ever designed a small embedded system, only to find that the power supply costs more than the remaining components, you might want to take a closer look at Bob's project and consider rolling your own power supply.

Despite the availability of components that simplify power supply design, many engineers treat power supply design like the bubonic plague and purchase only commercial supplies.

Although there are good reasons to buy power supplies, there are also some good reasons to roll your own. The purpose of this article is to show you what can be done to simplify the powering of an embedded system.

One good reason for buying a power supply is to avoid line circuits and potential liabilities. Given a choice of input power sources, I'd pick a wall transformer every time.

I agree that wall warts tend to clog the receptacles on a power distribution strip, but they're also available with input cords and standard plugs. Wall transformers also provide a degree of internationalization because a variety of input voltages and receptacles are available. If you have an application that requires less than 15 W at a single unregulated voltage, a wall transformer is a no-brainer.

Unfortunately, many embedded applications require several voltages with some level of regulation. A DC/DC converter with multiple output voltages is ideal for turning a wall transformer into a small power supply.

At this point, you have to form an architecture for your power system. If you limit your power requirement to less than 20 W, a DC converter with multiple regulated outputs is ideal. Power from an external wall transformer (AC input, unregulated DC output) feeds an inexpensive DC converter circuit built on the application PCB. Looking at a databook, you quickly realize that, in order to design a converter, you need to deal with magnetic components.

Magnetic design isn't that difficult, but it's a task in which many engineers have limited experience. Transformers are one of the most difficult components to acquire off-the-shelf. Even if you design a transformer and build a prototype, where do you purchase a few hundred? The answer better be offshore. Small high-frequency transformers are labor intensive, and domestic labor rates make building these units expensive.

A DC converter reference design with a source for the magnetic components is required. This article provides a reference design, a supply of low-cost magnetics, and a discussion of the tradeoffs made during the design phase. With this knowledge, you can adjust the design to a specific application. Switching converters come in various configurations, and each configuration has its own advantages.

In this application, I'm looking for less than 20 W, low cost, and multiple regulated outputs. I want all the components to be PCB-mounted using a minimal amount of PCB space. The 20-W limit with multiple outputs suggests a flyback converter. Flyback

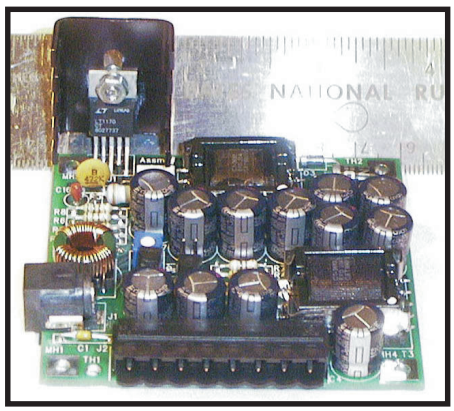


Photo 1—*The complete QuickConvert circuit is fairly compact.*

converters tend to be noisy, but this is a low-power supply, so filtering shouldn't be a problem. I'll address the noise issue later.

Current-mode PWM controllers are widely available from a number of sources. One advantage of current mode PWMs is that they limit primary currents during fault conditions.

Physical design constraints kept the circuit small and inexpensive. Keeping the parts at a low height also made the design easy to adapt. The problem with controlling component height was capacitor selection. Keeping capacitors to less than 0.5" and getting low ESR is difficult. In addition to detachable 0.200" spaced terminal blocks, a 2.1-mm power jack provides direct connection to a wall wart.

WINDING DESIGN

Using flyback converters provides multiple regulated outputs. Using a feedback loop to regulate a single output allows additional output winding to generate regulated outputs. Various output voltages are generated by controlling the turn ratios between output windings.

Although flyback regulation is fine for input variations, load changes introduce output variations. Winding design in the flyback magnetics significantly affects cross-regulation. Output windings close to the primary can be expected to pickup extra output energy from primary leakage flux. Likewise, any one winding that is heavily loaded will also affect nearby windings.

The Quick Convert output windings shown in Figure 1 were designed for 5-, 3.3-, 13-, and –13-V outputs. Targeting 13-V outputs lets you use precision 12-V linear regulators for flash memory or other voltage-critical applications. Op-amps and other linear circuits work without post regulation.

Applications requiring additional voltages can rearrange secondary winding to increase output voltages. Connecting a 13-V winding in series with the 5-V winding provides 18 V (great for VEE supplies for many LCDs). If you need 0.5 A on the 13-V supply, you can connect the two 13-V windings in parallel. Another trick is to pull a little energy during the conduction phase of the PWM switch. This operates the transformer in a forward converter mode. A 12-V input transformer turn ratio provides 24 V on the 13-V windings and –9 V on the 5-V winding.

These additional voltages aren't regulated as they track the input voltage. Using a regulated wall adapter will result in regulated outputs. Be sure to keep the filter capacitors at low values and use small series resistors with the rectifier. Operation as a forward converter results in load impedance being reflected directly to the PWM switch. Large capacitive loads can trigger PWM current mode limits and reduce available load power.

Using all these tricks can provide 5, 3.3, 13, –13, –9, 24, and –24 V. You also get another supply voltage—the feed from the wall transformer. LCD backlight supplies operate from a wide input voltage range and regulate lamp current for a constant lamp output. This arrangement allows a backlight to operate directly from the wall transformer output, saving regulator capacity and adding another 4 to 8 W to the power budget. Because the supplies for digital logic tend to be the most critical, a single taped winding was used for both 5- and 3.3-V windings. Direct regulation of the 5-V (or the 3.3-V) output provides good regulation on the other digital output.

A taped winding is difficult to construct, but the quality of the regulation is good. The 13-V outputs can be expected to increase if they are lightly loaded and the digital outputs are heavily loaded. Increase the 13-V load current to compensate. A voltage adjustment control is provided, which can be used to adjust the overall placement of all outputs.

The difficult part of designing the windings is producing the number of required turns with the proper amount of copper cross section. You want to keep windings designed so that a single traversal of the bobbin provides a completed winding. Using two or more strands of a smaller wire size adjusts the winding width to the bobbin width.

NOISE CONTROL

Control of switching transients is provided with an RC network (3.9 Ω and 4700 pF) across the primary winding. This AC dummy load reduces dv/dt on the primary switch at the expense of some efficiency. Cost was controlled by using electrolytic capacitors as both input and output filters. The ESR rating of these capacitors is important. Be sure to use low ESR units. The Panasonic units listed in the parts list are 0.12 Ω. A better capacitor would reduce input and output ripple, but the real EMI offender is usually higher frequency common mode currents.

An electrostatic shield between primary and secondary circuits would reduce common mode currents, but this design uses common mode chokes on both input and outputs.

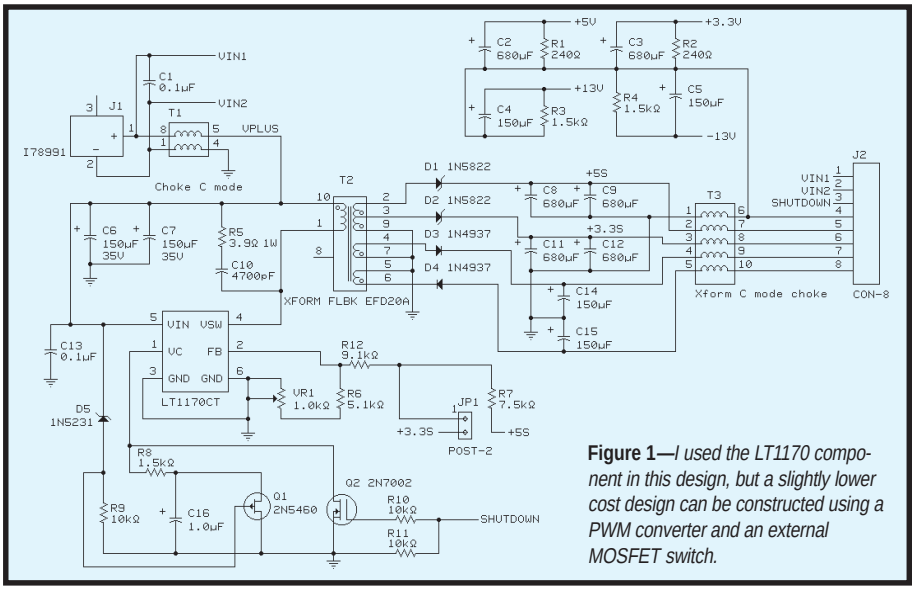


Figure 1—*I used the LT1170 component in this design, but a slightly lower cost design can be constructed using a PWM converter and an external MOSFET switch.*

When placing this circuit with an application on a PCB, it is unlikely that both chokes would be used. Conducted EMI is controlled with the input choke, and the output choke is useful only if you have significant cable runs from the converter to a PCB. In either case, be careful that the output common is not connected to the input power negative terminal. Connecting these will result in choke bias currents, which saturate choke magnetics and reduce the effectiveness of the common mode chokes.

When porting this circuit to an application, pay attention to the PCB artwork in the design package. Several nodes are connected using a section of copper pours. These are areas with high di/dt currents. Ignoring these nodes introduces high-frequency noise.

CONTROL CIRCUITS

A few additional control circuits are required to make the power supply more robust. A zener diode and P Channel JFET form a low-voltage cutoff that shuts down operation if the input voltage falls below a minimum operating level.

Unfortunately, small DC converter circuits tend to draw excessive currents at low input voltages. Adding a low-voltage cutoff limits current in switch and magnetic circuits and reduces transient loads on the power source. Remote shutdown is supported using a transistor switch.

A set of dummy resistors is provided—one dummy load resistor is connected to each output line. The rectifier diodes and output capacitors make good energy capture and storage circuits. Leakage inductance within the flyback transformer causes a small amount of energy to be transferred on every switching cycle. Without the dummy load resistors, the capacitor voltage would increase until either the capacitor or the diode fails.

Dummy resistors must be sized to provide the minimum safe load at all operating levels. The worst case is with fully loaded digital (5- and 3.3-V) outputs with no load on the 13- or –13-V outputs. Once the circuit is integrated with an application, the dummy loads can be removed. If an output isn't used, the rectifier diode for that circuit can be removed.

FINAL DESIGN

The final design (see Photo 1) uses EFD20 magnetics and provides 5 V at 1 A, 3.3 V at 1 A, 13 V at 0.25 A, and –13 V at 0.25 A. Total output power is limited to 10 W. The input voltage range is 8 to 35 V. Total output power can be increased by changing the cutoff zener diode to 12 V. A jumper allows direct regulation of either the 5-V or 3.3-V output.

Costs can be reduced through several changes. Expensive connectors found on the reference design can be eliminated for embedded applications. EMI chokes on the power output are not required, most processor-based circuits generate more EMI than the power converter. The input choke is important, but a choke on the transformer cord might provide the required isolation.

PWM driver ICs and MOSFET switches reduce purchased material cost and a low-resistance MOSFET eliminates the need for a heatsink. This approach allows you to change operations but requires discrete Rs and Cs, which drive up assembly cost.

The complete design is available along with production samples, and I'd be happy to answer any questions or assist with changes you might want to make.

Robert Kondner has more than 20 years of experience designing small products and embedded systems. Many applications have used ARM processors with LCD graphic panels in portable or vehicle environments. You may reach him at sales@index designs.com.

SOFTWARE

A design package including the schematic, parts list, PCB gerber files, and printouts of artwork layers is available for download.

SOURCE

Magnetic components, completed circuit boards
Index Designs
www.indexdesigns.com

Building a RISC System in an FPGA

FEATURE ARTICLE

Jan Gray

Part 3: System-on-a-Chip Design

Now that the xr16 RISC processor is complete, it's time to tie everything together and wrap up this series. In this final part, Jan designs a demo system that includes an on-chip bus, memory controller, video controller, and peripherals.

The xr16 RISC processor is designed, now it's time to design the rest of the System-on-a-Chip (SoC). Besides the CPU, the FPGA hosts an on-chip bus, bus controller, parallel port, RAM, video controller, and an external SRAM controller.

This month, I'll show how simple interfaces can make SoC design as straightforward as classic CPU, glue logic, memory, peripherals, and PCB design used to be.

XS40 BOARD

The project targets the XESS XS40-005XL V.1.2 FPGA board in Photo 1, which includes a Xilinx XC4005XL, 12-MHz oscillator (see Figure 1), 32-KB SRAM, 8031 MCU, 7-segment LED, voltage regulators, and parallel port and VGA port connectors. It's simple, inexpensive, and is featured in *The Practical Xilinx Designer Lab Book* included with Xilinx Student Edition.

I chose this board because it is well supported with documentation and tools, and because it can be used for both the XSE exercises and this project.

A SYSTEM-ON-A-CHIP

I'll build an integrated system from the resources at hand—the FPGA, RAM, the video and parallel ports, and the 12-MHz oscillator.

I used the RAM for program, data, and video memory. The byte-wide, asynchronous SRAM isn't ideal, but it is fast enough for you to read and latch a byte on each clock edge, thereby fetching a 16-bit instruction during each cycle.

By displaying all 32 KB of RAM, you can fashion a bitmapped 576 × 455 monochrome video display at VGA-compatible sync frequencies. How quaint, to watch every bit on screen!

Refer also to Figure 4, the FPGA top-level schematic. It includes the processor (P), the system memory/bus controller (MEMCTRL), the on-chip 16-bit data bus ($D_{15:0}$), on-chip peripherals (PARIN, PAROUT, and IRAM), the external SRAM interface, and the VGA video controller.

DECISIONS, DECISIONS

Before examining the design, let's briefly explore the on-chip bus design space. (This is not the sort of thing you worry about when designing to someone else's microprocessor, but in an FPGA SoC, you have a little more freedom.)

Bus design issues include how many bus masters are permitted, how is the bus clocked and pipelined, how wide is it, does it provide byte addressing, and is it split or unified with the processor core RESULT bus.

For XSOC, the pipelined on-chip 16-bit data bus $D_{15:0}$ is single-mas-

Address	Resource
0000-7FFF	external 32-KB RAM, video frame buffer
0000	reset handler
0010	interrupt handler
FF00-FFFF	I/O control registers, 8 peripherals × 32 bytes
FF00-FF1F	0: 16-word on-chip IRAM
FF21	1: parallel port input byte
FF41	2: parallel port output byte
FF60-FF7F	3: *unused*
…	…
FFE0-FFFF	7: *unused*

Table 1—*The system memory map includes eight decoded peripheral control register address blocks.*

tered (but recall the CPU also performs DMA transfers), the bus clock is the CPU clock, and the on-chip data bus is unified with the processor's $RESULT_{15:0}$ data bus. All of these design decisions help to keep this project simple.

BUS CONTROLS

MEMCTRL, the system bus/memory controller, interfaces the processor to the on-chip and off-chip peripherals. It receives the pipelined "next transaction" memory request signals $AN_{15:0}$, WORDN, READN, DBUSN, and ACE from the CPU. Then, it decodes the address, enables some peripheral or memory, and later asserts RDY in the clock cycle in which the memory cycle completes. I/O registers are memory mapped (see Table 1).

There are eight transaction types: (external RAM or I/O) × (read or write) × (byte or word), all decoded from $AN_{15:0}$, WORDN, and READN.

MEMCTRL manages transfers on the on-chip data bus $D_{15:0}$ and the external data bus $XD_{7:0}$ by asserting various tri-state output enables (*x*T) and control register clock enables (*x*CE). These enable signals are asserted according to the transaction type (see Table 3).

For example, during `sw r0, 0xFF00`, MEMCTRL decodes an I/O write word request. It asserts LDT and UDT, driving the store data onto $D_{15:0}$, and asserts IRAM/LCE and IRAM/UCE, writing $D_{15:0}$ into IRAM's SRAMs:

$$IRAM/D_{15:0} := D_{15:0} \leftarrow DOUT_{15:0}$$

Next, consider a store to external RAM: `sw r0,0x0100`. Because the external data bus is only eight bits wide, first store the least significant byte, then the most significant byte. First, MEMCTRL asserts LDT and XDOUTT:

$$XD_{7:0} \neg D_{7:0} \neg DOUT_{7:0}$$

Later, it asserts UDLDT and XDOUTT:

$$XD_{7:0} \leftarrow D_{7:0} \leftarrow DOUT_{15:8}$$

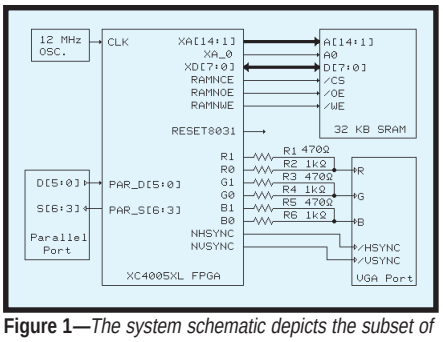


Figure 1—*The system schematic depicts the subset of the XS40 needed for our project. The 8031 (not shown) is held in reset.*

BUS INTERFACE

Now, let's design an on-chip bus peripheral interface to enable robust and easy reuse of peripheral cores and to prepare for an ecology of interoperable cores to come.

It helps to distinguish between core users and core designers. The former are more numerous, while the latter are more experienced. Therefore, I make ease-of-use tradeoffs in favor of core users.

Because FPGAs are malleable and FPGA SoC design is so new, I wanted an interface that can evolve to address new requirements without invalidating existing designs.

With these two considerations in mind, I borrowed a few ideas from the software world and defined an abstract control signal bus with all of the common control signals collected into an opaque bus $CTRL_{15:0}$.

MEMCTRL drives CTRL and also does I/O address decoding, driving the eight I/O selects $SEL_{7:0}$.

Now, you need only instantiate the core, attach CLK, CTRL, D, some SEL_i, any core-specific inputs and outputs, and you're done!

Contrast this with interfacing to a traditional peripheral IC. Each IC has its own idiosyncratic set of control signals, I/O register addresses, chip selects, byte read and write strobes, ready, interrupt request, and such. They don't call it glue logic for nothing.

Of course, we can't just sweep all the complexity under the rug. Each core must decode CTRL and recover the relevant control signals. This is done with the DCTRL (CTRL decoder) macro (see Figure 5). DCTRL inputs SEL_i, $CTRL_{15:0}$, and CLK and outputs local I/O register address, upper and lower byte output enables (read strobes), and clock enables (write strobes).

Within each DCTRL instance, you do final address decoding for the specific peripheral, combining its SEL_i signal with the I/O select within $CTRL_{15:0}$. Here XIN8 only uses LDT (the LSB output enable). The other DCTRL outputs are unloaded and automatically eliminated by the FPGA implementation tools.

Using DCTRL and the on-chip tri-state bus, the typical overhead per peripheral is only one or two CLBs, and perhaps a column of TBUFs.

Control signal abstraction can also make bus interface evolution easy. If you revise MEMCTRL and DCTRL together, arbitrary changes to $CTRL_{15:0}$ can be made without invalidating any existing designs. And, to add new bus features, simply design a new decoder DCTRL_v2, causing no changes to existing DCTRL clients.

EXTERNAL I/O INTERFACE?

There isn't one. If it were necessary to attach external peripherals, perhaps to the $XD_{7:0}$ bus, you might design some on-chip external peripheral adapter macros. Just like an on-chip peripheral, each adapter would take CTRL and some SEL_i, but its job would be to use additional I/O pins to control its peripheral IC's chip selects and so forth. Of course, as a $CTRL_{15:0}$ client, it would be able to raise interrupts, insert wait states, and so forth.

EXTERNAL RAM

The external RAM is a classic 32-KB fast asynchronous SRAM with a 15-ns access time (t_{AA}). Its pins in-

Enable	Effect
LDT	$D_{7:0} \leftarrow DOUT_{7:0}$
UDT	$D_{15:8} \leftarrow DOUT_{15:8}$
UDLDT	$D_{7:0} \leftarrow DOUT_{15:8}$
XDOUTT	$XD_{7:0} \leftarrow D_{7:0}$
LXDT	$D_{7:0} \leftarrow XDIN_{7:0}$
UXDT	$D_{15:8} \leftarrow XDIN_{15:8}$
p/LDT	$D_{7:0} \leftarrow p/D_{7:0}$
p/UDT	$D_{15:8} \leftarrow p/D_{15:8}$
p/LCE	$p/D_{7:0} := D_{7:0}$
p/UCE	$p/D_{15:8} := D_{15:8}$

Table 2—*There are a set of enables* p/* *within each peripheral.* $DOUT_{15:0}$ *is the CPU store data output register (see Part 1,* Circuit Cellar *116).*

clude A14:0 (address), D7:0 (data in/out), /CS (chip select), /WE (write enable), and /OE (output enable).

Refer to Figure 2 and the external bus and SRAM interface block of Figure 5.

$XA_{14:1}$ is 14 IOBs configured as OFDXs (output flip-flops with clock enables). $XA_{14:1}$ captures the next address $AN_{14:1}$ at the start of each new memory transaction. XA_0 (XA_0) is the least significant bit of the external address. It is a logic output and can change on either CLK edge.

$XD_{7:0}$ is eight IOBs configured as eight sets of simultaneous OBUFTs (tri-state output buffers), IBUFs (input buffers), and IFDs (input flip-flops).

During a RAM write, XDOUTT is asserted, RAMNOE is deasserted, and the OBUFTs drive $D_{7:0}$ out onto $XD_{7:0}$.

During a RAM read, XDOUTT is deasserted, RAMNOE is asserted, and

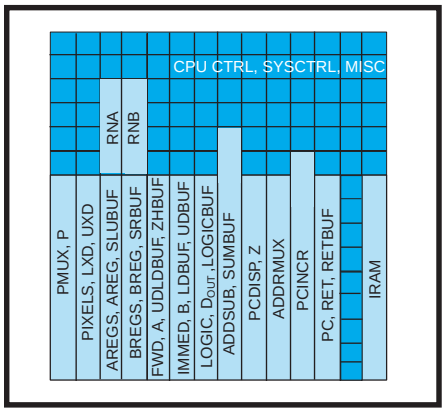


Figure 3—*The rest of the device contains the automatically placed processor control unit and other logic.*

the RAM drives its output data onto $XD_{7:0}$. The data is input through the IBUFs and latched in the XDIN IFDs (on each falling CLK edge).

To keep the CPU busy with fresh new instructions, the system reads both bytes of a 16-bit word in one cycle. In the first half cycle, it sets XA_0=0, reading the MSB, and latches it in XDIN. In the second half cycle, the system sets XA_0=1, reading the LSB, and reads it through IBUFs. The catenation of these two bytes, $XDIN_{15:0}$, feeds the CPU's INSN port, the video controller's PIX port, and $D_{15:0}$ via the byte-wide tri-state buffers LXD and UXD.

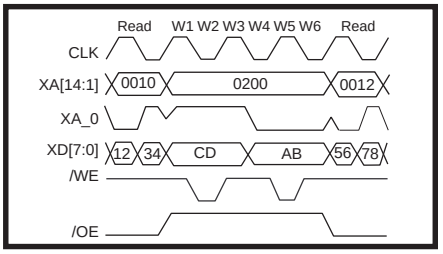


Figure 2—*The RAM interface signals for three memory transactions are: read 1234 from address 0010, write ABCD to address 0200, and read 5678 from address 0012.*

Writes to asynchronous SRAM require careful design. Let's see if we can safely write one byte per clock cycle. The key constraints are:

- address must be valid before asserting /WE
- data must be valid before deasserting /WE
- /WE must be deasserted briefly
- no adddress/data hold time after /WE

I required a fully synchronous design to be able to slow or stop the clock and was unwilling to employ any asynchronous delay tricks.

Accomplishing this requires one half clock to settle the write address, one half clock to assert /WE, and one half clock to deassert it. Therefore, byte writes take two full cycles, and word writes take three (e.g., a word write takes six half cycles W1–W6):

- W1: assert $XA_{14:1}$, data LSB, XA_0=1
- W2: assert /WE
- W3: deassert /WE, hold XA and data
- W4: assert data MSB, XA_1=0
- W5: assert /WE
- W6: deassert /WE, hold XA and data

MEMCTRL DESIGN

I've discussed the responsibilities of MEMCTRL design: address decoding, on-chip bus control, and external RAM control. Now, let's review its implementation (see Figure 6).

In address decoding, if the next access is a load/store to address FFxx, the access is to memory-mapped I/O, and SELIO is asserted. Otherwise, it's a RAM access.

Within each peripheral's DCTRL instance, its SEL_i (decoded from $AN_{7:5}$) and $CTRL_{SELIO}$ combine to develop that peripheral's output and clock enables.

For bus control, the current state of the memory transaction finite state machine determines which controls are asserted. The CPU asserts ACE (address clock enable) to request the next transaction and awaits RDY. MEMCTRL decodes the request, and the FSM enters the IO, RAMRD, or RAMWR state. The latter has three sub-states—W12, W34, and W56—corresponding to pairs of the W1–W6 half-states described previously.

In the IO state, RDY is asserted unless the selected peripheral deasserts $CTRL_0$, the I/O ready line, thereby inserting a wait state.

In the RAMRD state, RDY is asserted immediately because all RAM reads require only one clock cycle. In the RAMWR state, RDY is asserted on W34 for byte stores and on W56 for word stores.

The write controller uses flip-flops W23_45 and W45, which are clocked on CLK falling edges. So, W34 is true during W3 and W4, while W45 is true during W4 and W5. From the W* signals you derive glitch-free control signals XA_0, /WE, /OE, and so on.

The rest of MEMCTRL is straightforward. Note how E encodes (renames) the various peripheral control signals to $CTRL_{15:0}$.

I technology-mapped some logic using FMAPs. Timing analysis had revealed poor automatic mapping of this logic. This change shaved a few nanoseconds off the critical path.

Now that we've covered the implementation of MEMCTRL, let's turn our attention to peripherals.

Transaction	Cycles	Enables
RAM read byte	1	LXDT
RAM read word	1	LXDT, UXDT
RAM write byte	2	LDT, XDOUTT
RAM write word	3	LDT or UDLDT, XDOUTT
I/O read byte	1+	*p*/LDT
I/O read word	1+	*p*/LDT, *p*/UDT
I/O write byte	1+	LDT, *p*/LCE
I/O write word	1+	LDT, UDT *p*/LCE, *p*/UCE

Table 3—*Depending on the memory transaction, different bus output enables and register clock enables are asserted.*

PARALLEL PORT I/O

I provided parallel port I/O to communicate with the host. The XS40 board provides eight parallel port data inputs and five status outputs. Reserving a few for debug I/Os, I used six inputs and four outputs.

During `lb rd,FF41`, the PARIN input peripheral is selected, driving the inputs 00 || $PAR\_D_{5:0}$ onto $D_{7:0}$ (see Figure 5).

During `sb r1,FF21`, the PAROUT output peripheral is selected, capturing the store data $D_{3:0}$ in flip-flops, which drive the $PC\_S_{6:3}$ status outputs.

XOUT4 is as simple as XIN8. It has a DCTRL decoder, of course, and clocks $D_{3:0}$ on LCE (LSB clock enable). This parallel port requires only three CLBs, eight TBUFs, and 10 IOBs!

ON-CHIP RAM

XSOC also includes a 16 × 16-bit RAM peripheral. It uses all of the DCTRL outputs: $A_{4:1}$ to select the word to read or write, LCE and UCE as lower and upper byte write strobes, and LDT and UDT as lower and upper byte output enables.

VIDEO CONTROLLER

The bit-mapped video controller, based on ideas from [1], displays all 32 KB of external SRAM at 576 × 455 resolution, monochrome.

It runs autonomously from the CPU, and so is not a peripheral on the on-chip bus. It uses DMA to fetch video data, which consumes about 10% of memory bandwidth.

A video signal is a series of frames; each frame is a series of lines, and each line is a series of pixels. The video controller fetches 16-pixel words of video memory, shifts the pixels out serially, and uses horizontal and vertical sync pulses to format the pixels into frames and lines for the monitor.

Generating VGA-compatible horizontal and vertical sync timings, VGA shifts pixels out at 24 MHz, twice the system clock rate, shifting one out when CLK is high and a second when it is low. The horizontal and vertical sync pulses are advanced a few clocks (lines) to center the display in the frame (see Table 5).

The VGA ports are described in Table 6. The first five ports request new pixel data via the DMA controller. The rest are the VGA video outputs. The red, green, and blue intensities R1, R0, G1, G0, B1, and B0 drive resistor-based 2-bit D/A converters, providing up to 64 colors (4 × 4 × 4). However, at this resolution, with 32 KB of RAM, you can only support a monochrome (1-bit/pixel) display. So, each pixel bit drives all six outputs, drawing black or white pixels.

To generate horizontal and vertical syncs and a video blanking signal, you need a 9-bit horizontal cycle counter and a 10-bit vertical line counter.

After 288 clocks, it's time to blank the video. Assert horizontal sync after 308 clocks, deassert it after 353, and reset the counter and re-enable video after 381 clocks (one line).

In the vertical direction, the VGA controller must blank video after 455 lines, assert vertical sync after 486 lines, deassert it after 488 lines, and reset the counter, re-enable video, and reset the video DMA address counter after 528 lines.

The simplest way to build each counter is with a Xilinx library binary counter, such as a CC16RE. But because I had just about filled the FPGA, and because they're cool, I designed a more compact 10-bit linear feedback shift register (LFSR) counter. This uses a 10-bit serial shift register which

Photo 1—*Here's the XS40 board, with the project design loaded into the FPGA and running a demo program that's drawing graphics on the monitor.*

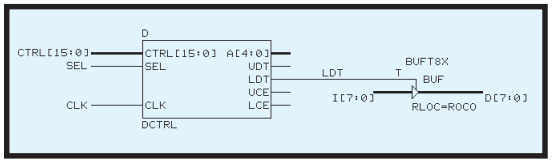


Figure 5—*The XIN8 (PARIN) implementation shows the CTRL decoder output LDT that enables the input byte to be driven onto the data bus.*

has an input that is the XOR of certain shift register output taps.

An n-bit LFSR repeats every 2^n-1 cycles, but you can make an arbitrary m-cycle counter by complementing the LFSR input bit, thereby short-circuiting the full sequence when a particular bit pattern is recognized. My LFSR counter design program can be downloaded from the *Circuit Cellar* web site.

Referring to Figure 7, note the video controller contains two LFSR counters, H and V. Each has four comparators to compare the LFSR bit patterns to the count patterns output by my program.

Each of the J-K flip-flops HENN, NHSYNC, VEN, and NVSYNC are set on reaching one counter value and reset on reaching another.

NHSYNC is asserted low during clocks 308–353, and NVSYNC during lines 486–488. HEN is the pipelined horizontal video enable, and VEN is the vertical video enable. When both are true, you fetch and shift out video data.

In the video datapath, each clock shifts out two bits of video data. Every eight clocks, WORD goes true, and it requests a new 16-bit word of video data from memory. REQ is asserted, registering a pending DMA transfer with the CPU.

Five or fewer clocks later, the CPU performs the DMA load, asserting ACK. The video data word is latched in the PIXELS staging register. On the eighth clock, this word is loaded into the PMUX 8 × 2 parallel-load serial-out shift register.

Two bits shift out of PMUX during each clock, and feed a 2–1 mux that drives the 1-bit pixel each half clock.

SYSTEM BRING-UP

After designing the CPU, I designed a simple test-fixture using on-chip ROM and ran my test programs in the Foundation simulator.

After simulating test programs for hundreds of cycles, I compiled the design using the Xilinx tools and tested it on my XS40 board. Using a parallel port output for CLK, I wrote shell scripts to single-step the processor and observe $PC_{7:1}$ on the LEDs. Later, I ran the CPU at up to 20 MHz.

Starting from a core set of working instructions, it was easy to test the rest, one at a time. If something went awry, I could do a binary search for the problem, insert a `stop: goto stop;` breakpoint into my test, recompile, and download. A real remote debugger would be nice!

Armed with a working CPU, it is easy to add and test new features, one by one. I added double-cycled reads from external RAM, then MEMCTRL, then LED output registers. Writing text messages to the seven-segment LED was a big milestone. RAM writes were next. And, late in the project I added DMA, the video controller, and interrupts.

I want to emphasize the importance of thorough testing. You have your work cut out for you when properly testing a pipelined processor and an SoC.

This has been a proof-of-concept project, and I have focused on design issues. To ship something like this, you would need to budget as much or more time for validation as for the design and implementation.

The final system floorplan, as placed on our 14 × 14 CLB FPGA, is shown in Figure 3.

SERIES WRAP-UP

In this three-part series, I have presented the complete design and implementation of a real, full-featured, pipelined microprocessor and an integrated System-on-a-Chip. I designed a new instruction set, ported

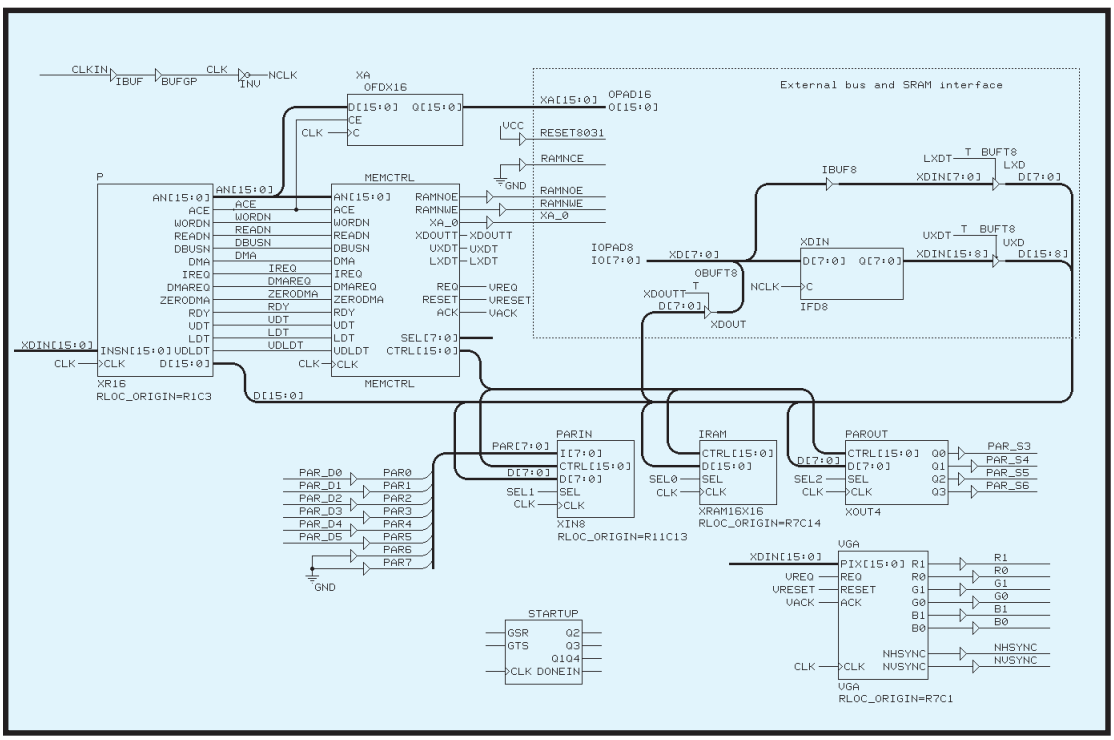


Figure 4—*The processor (P) issues requests to MEMCTRL, accessing instruction and data via the on-chip bus $D_{15:0}$ or external SRAM. Integrated peripherals provide parallel port I/O and on-chip RAM. The VGA controller fetches pixel data via DMA.*

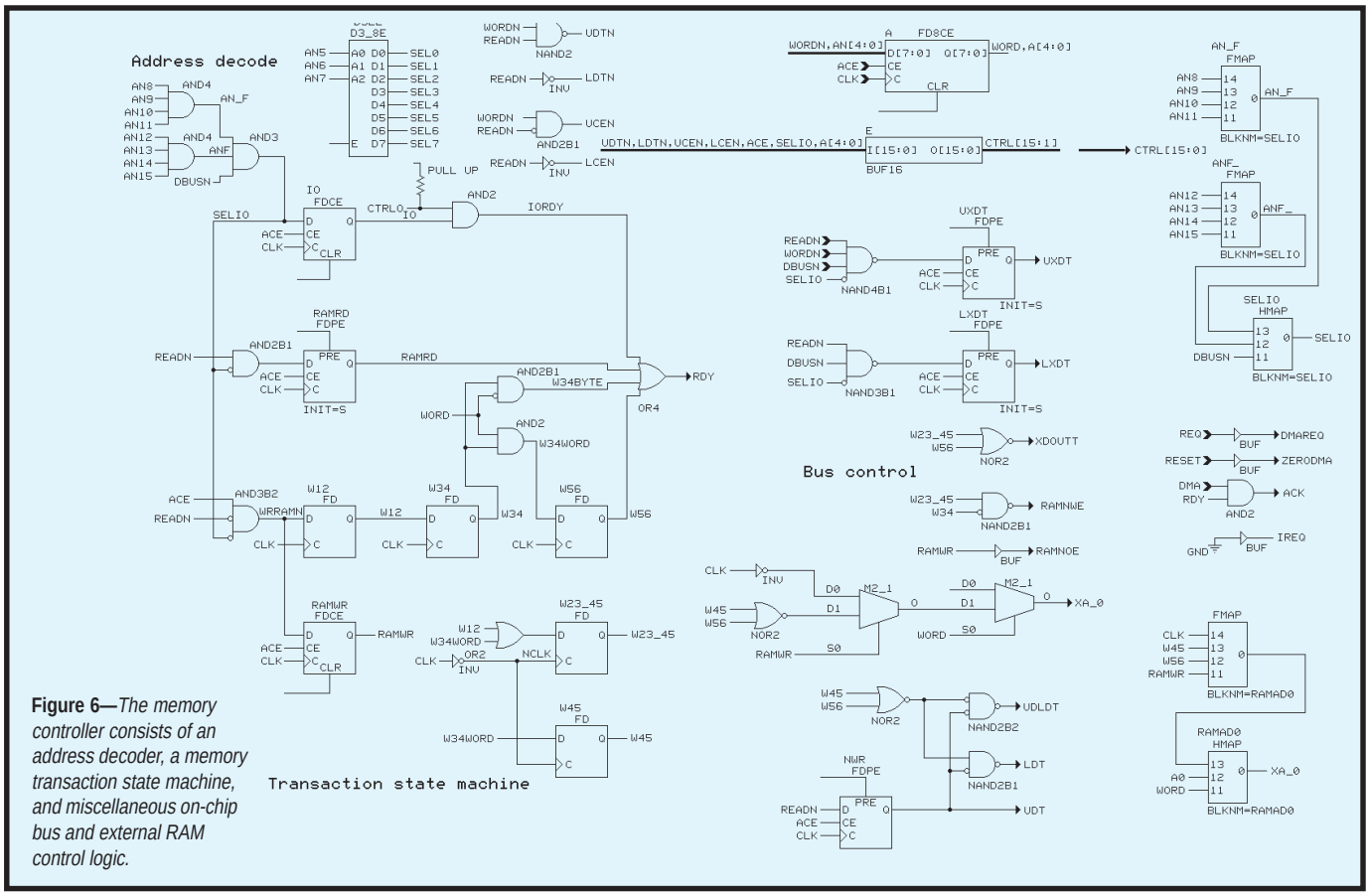


Figure 6—*The memory controller consists of an address decoder, a memory transaction state machine, and miscellaneous on-chip bus and external RAM control logic.*

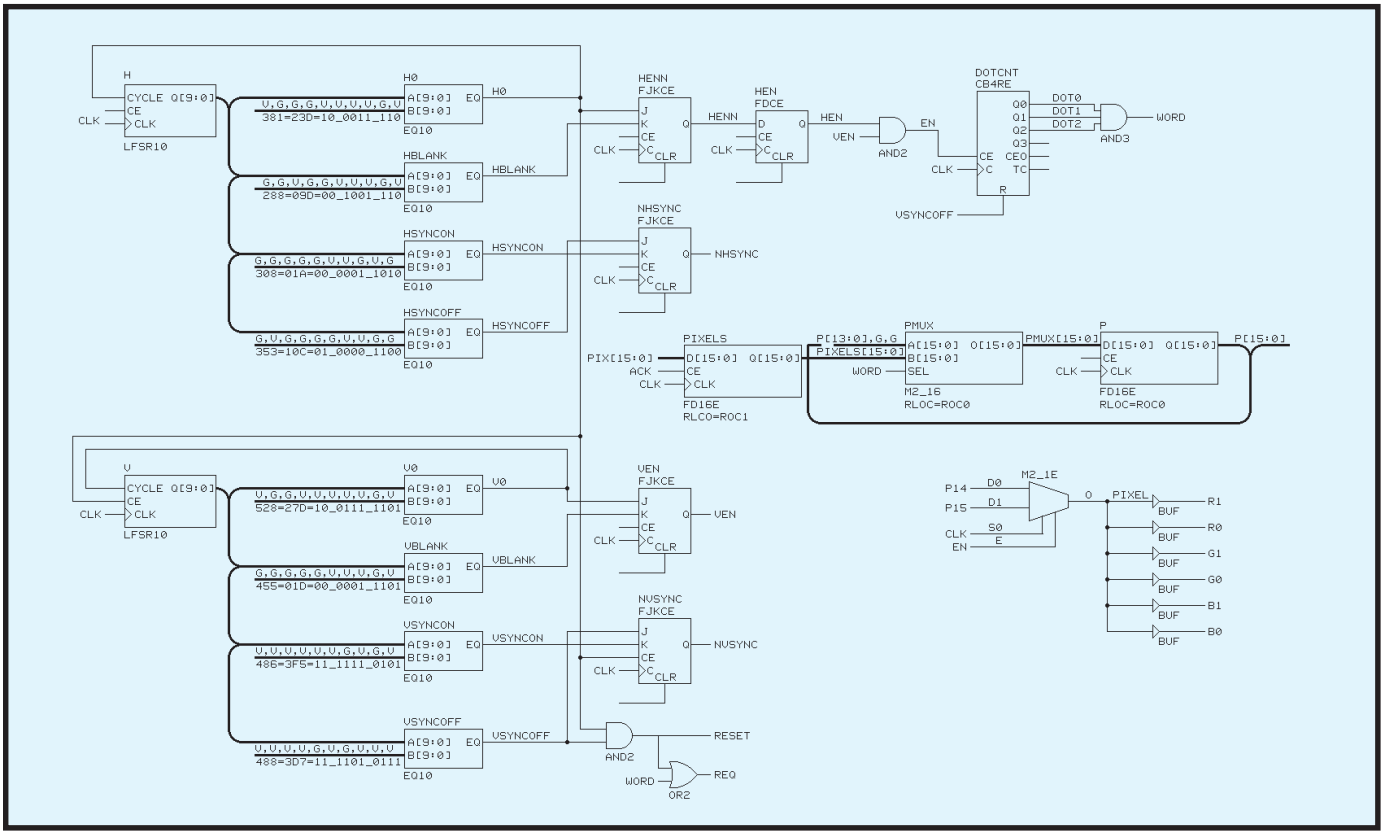


Figure 7—*As you can see, the video controller contains two LFSR counters that each have four comparators for comparing the LFSR bit patterns to the count patterns that are output by the program that I wrote.*

a C compiler, and discussed how to build practical CPUs in inexpensive FPGAs. I also explored pipelined processor design.

In this article, you have seen how to build an integrated system with reusable cores. The *Circuit Cellar* web site provides the full set of schematics, tools, links, and such necessary for you to download and build this project, with the help of the Student Edition tools and the XESS XS40-005XL proto board. Have fun!

Ashok Patel inspired these articles long ago when he and his friends kindly taught a teenager digital design. Later, he challenged me to share in kind what I have learned.

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

Port	Description	Quantity	Value
$PIX_{15:0}$	next 16-bit pixel word	two-pixel clock	83.3 ns
REQ	request DMA of next word	one-pixel half-clock	41.7 ns
RESET	reset DMA address counter	visible pixels/line	576
ACK	DMA acknowledge input	visible clocks/line	288
CLK	system clock	horizontal sync "on" clock	308
R1,R0	2-bit red intensity	horizontal sync "off" clock	353
G1,G0	2-bit green intensity	line total clocks	381
B1,B0	2-bit blue intensity	line time	31.8 ms
NHSYNC	active-low horizontal sync	visible lines/frame	455
NVSYNC	active-low vertical sync	vertical sync "on" line	486
		vertical sync "off" line	488
		frame total lines	528
		frame time	16.8 ms

Tables 5 & 6—*The 12-MHz clock and 24-MHz pixel shift frequency determines the pixels per line and lines per frame, as well as the horizontal and vertical counter values for sync and blanking events.*

Please note that I do not warrant that you have the right to build something based upon the ideas discussed in this series of articles under the relevent intellectual property laws in your jurisdiction.

SOFTWARE

You may download more information, including specifications, source code, schematics, and links to related sites from the *Circuit Cellar* web site.

REFERENCE

[1] *VGA Signal Generation with the XS Board*, XESS App Note www.xess.com/fpga/vga.pdf

SOURCES

XESS XS40-005XL
www.xess.com/fpga

FPGAs, Student Edition tools
Xilinx, Inc.
(408) 559-7778
Fax: (408) 559-7114
www.xilinx.com

NOUVEAU PC

Edited by Harv Weiner

SINGLE BOARD COMPUTER

The **6225** is an industrial single-board computer that integrates the most often used I/O, four serial parallel, FDD, HDD, 24 digital I/O, programmable watchdog timer, and 10BaseT with a low-cost-per-function. It is ideal for distributed control, POS, ticketing, operator interface, weighing equipment, and other applications where a number of computers are networked through Ethernet. The 6225 can be expanded through the digital I/O. Other features include a real-time clock, watchdog timer, and support for two floppy drives, keyboard, and speaker. The board is compatible with DOS, QNX, and Linux.

The 6225 operates from –40°C to 85°C and can withstand high shock and vibration. Low power consumption allows the 6225 to be placed in sealed enclosures without the use of a fan. The card has an ambient temperature sensor that is accurate to ±3°C over a range from –40°C to 85°C and can be read remotely. The simple operator interface of a keypad and four-line LCD display is fully supported with driver software. The card is also available depopulated for the most cost-effective OEM solutions.

The 6225 sells for **$545**.

Octagon Systems Corp.
(303) 430-1500
Fax: (303) 426-8126
www.octagonsystem

DIGITAL I/O BOARD

DATEL has introduced the **CPCI-560** digital input/output board for the Compact PCI (CPCI) bus. Featuring 24 I/O lines, the board is suited for process control applications, instrumentation that needs logic-level lines and external devices, apparatus status, and control of special test fixtures.

The I/O lines form three separate byte-wide ports and may be fully programmed as either inputs or outputs in three 8-bit groups. The lines may be software configured as a dual-byte buffered strobed digital interface with full handshakes and latch controls. Each line is capable of driving one TTL load.

A separate gatable input port may be programmed to generate a maskable interrupt to the CPCl bus for alarm conditions or status monitoring. Standard interrupts, A through D, are selectable. The user may assign a front panel LED lamp for any purpose (out of limit, system indicator, etc.). Each port may be accessed at up to 2 MHz if host-side software is fast enough to respond.

All input/output connections use a standard 37-pin D-connector mounted on the front panel. The CPCI bus supplies all power and spare fused 12-V and 5-V DC power is brought out. Full technical datasheets are available on Datel's web site.

Digital I/O and control panel software executables are included with the board for Windows 95/98 and Windows NT operating systems. Programmers developing their own code can obtain professionally commented source code that is written in Visual C++. The source code offers a DLL and device driver library and includes ActiveX controls, l.abVIEW and Visual Basic examples.

The CPCI-560 sells for **$295** in single quantity. The source code model CPCl-560WINS sells for **$395**.

Datel, Inc.
(508) 339-3000
Fax: (508) 339-6356
www.datel.com/boards.htm

EPC
REAL-TIME PCs

Ingo Cyliax

Real-Time Executive for Multiprocessor System

Part 1: Introduction to RTEMS

Ingo sets out to explain how the RTEMS open-source licensed real-time kernel has evolved from its days at U.S. Missile Command. You just might consider using the current C version to launch your next application.

The title Real-Time Executive for Multiprocessor Systems (RTEMS) sounds like it has something to do with the military. Well, it does.

RTEMS is an open-source licensed real-time kernel originally developed by the On-Line Applications Research Corporation (OAR) for the U.S. Missile Command. It was designed as a portable standards-based real-time executive for which source code was available and royalties were free. It was written before freely available source was known as open source.

The original name for RTEMS was Real-Time Executive for Missile Systems. Another current version, written in ADA, is known as Real-Time Executive for Military Systems. I will write about the version written in C in my next few articles. The sources and support are available for free. The company also offers commercial services like training, support, and custom development.

OVERVIEW

Like any open-source project, the idea is to provide a common software platform to write your application against. By feeding enhancements from the platform back to the free source code pool, the platform is dynamic and adaptable. Also, open-source works on an honor system. If users benefit from the work that was donated and develop something cool, they should consider donating it to the effort. Users are not expected to donate the application itself if it's a proprietary or custom application.

Companies dealing with open-source generate money by providing consulting or custom application development services for open-source code. For many projects, open source systems are a cost-effective solution.

In addition to all of the basic real-time features, RTEMS provides bells and whistles that are not available in other free real-time kernels.

It includes POSIX threads, real-time extensions, and a large portion of the POSIX 1003.1b-1996 API. Generally, this means that many of the applications written for this API will compile and work similarly between systems that support POSIX.

There is a full TCP/IP. This protocol stack uses the Berkeley socket interface, which is used on many platforms. The TCP/IP stack was ported from the FreeBSD project. Although it has a different open-source license than the rest of the system (which is GNU General Public Licensed), it is not intended to place any restriction on the application development that uses it. Networking support is optional in RTEMS.

Homogeneous and heterogeneous multiprocessor support is in RTEMS. In this model, global intertask communication is shared among processors. This makes the boundaries of running applications on multiple processors transparent. Because it also supports heterogeneous processor systems, you can do things like run the user interface on one type of system (which may have all of the user I/O) and crunch numbers on an array of processors that are better suited for computations.

One great feature of RTEMS is its portability. It's been ported to many CPU architectures and boards, thanks to the development tools. RTEMS for most architecture is built with GNU GCC, which can be configured for

many 32-bit architectures. Currently, RTEMS can be built for CPUs (see Table 1).

I haven't built all of these. I played with the ix86 version, but I like architecture flexibility. Occasionally, I work with the Coldfire and MC68*xxx* processors.

Also, RTEMS introduces board support packages (BSPs), a modular set of libraries that makes file templates and board-specific device drivers for specific board-level systems. By using standard devices like standard I/O drivers (the console), a clock driver, and a network interface driver, it's possible to move applications from one architecture/board to another by recompiling and linking.

There are several BSPs available for the x86 architecture (see Table 2).

Family	CPUs
Motorola MC68*xxx*	mc680[01234]0, mc683[03]2, mc68360
Motorola ColdFire	mcf5200
Hitachi SH	sh7032
Intel i386	i[345]86, i386ex
Intel i960	i960[ch]a, i960rp
MIPS	idt46[05]0
PowerPC	ppc403, 603e, 604, mpc750, mpc821, 823,860
SPARC	erc32
Hewlett-Packard PA-RISC	hppa7100

Table 1—*One of the virtues of RTEMS is that it runs on several architectures and chips. The GNU tool set creates a cross-compiler environment that makes the portability possible.*

GETTING RTEMS

In order to get RTEMS from the 'Net, you must download the pieces, build the tools needed to compile and link the system, and then build the system.

First, decide what to get. When I wrote this, there were two options. One was the last official release 4.0.0. This release is bundled up nicely and the instructions are well written concerning how to download and install it. This is where I started. I downloaded documentation, the RTEMS system sources, and the patches necessary to build an RTEMS-aware version of the GNU tools.

One problem is that patches for the GNU tools are based on older versions of the tools. You need to apply some context-based changes using the automatic tool called "patch." This works best if you apply the patches against the exact version that is specified. If you don't match up the versions, you have to apply the patches by hand.

Version 4.5.0 will be released soon and likely will be based on the current set of GCC tools. In addition, it has new features and targets. Until then, there are snapshot releases. These are bundles of the systems and tools for whenever things are stable enough to warrant a snapshot.

I use Linux as my primary working platform to write my columns, do most of my initial software development, and layout my PCBs. There is a commercial schematic capture/PCB layout package available for Linux. Best of all, it gives Unix workstation performance and flexibility on cheap Intel hardware. My laptop dual boots into Windows. Still, there are CAD tools not yet available for Linux.

Building RTEMS snapshots under Linux is easy. Red Hat pioneered binary tool kits are available in Red Hat Package Manager (RPM) form. RPMs tell you the dependencies needed before you install a package and warn about conflicts with an already installed component.

RPMs for the tools needed to build RTEMS are in the snapshot release. I pick the target architecture I want to build RTEMs for, download the RPMs, and install them. To build RTEMs for the i386 architecture, I need the i386 target tools.

This may seem odd because I'm running Linux on Intel, and Intel (i386) is the target of the system I want to build. Other development tools are available for other targets (e.g., it makes more sense to think of a cross compiler for m68*xxx* on an Intel Linux host than an i386 cross compiler on an Intel Linux host). Download the RPMs necessary to build an i386 target and install it.

As a side note, the GNU tool set can be built on most operating systems. You can visit www.cygnus.com to get and install the Win32 GCC tool set. Choose this if you want to use Windows to build RTEMS. But, for the snapshots, this may be more complicated than using the Linux-based tools. Most of the development work for RTEMS probably is hosted under Linux.

After installing the build tools, download the sources and documentation. For the snapshots, there are two kits. I downloaded `RTEMS-20000118a.tgz` for the source kit and an older documentation kit `rtmsdoc-20000105.tgz`.

The `tgz` extension means gzip'ed tar format. Tar is an archive format popular in the Unix world. Gzip is a GNU compression tool/algorithm that does not use licensed compression algorithms. The GNU version of the tar utility has the gzip compression built into it and has similar features to the zip in the DOS/Windows world. A stand-alone binary version of GNU tar is available for Windows from Cygnus' web site.

The documentation kits have manuals in an HTML-based directory in PDF and DVI versions.

CONFIGURING AND BUILDING THE SYSTEM

The source kit is designed so a build environment can remain separate. So, you can put the source kit on a read-only network file system or CD-ROM. When configuring the system to make a build directory, all configuration dependent files are copied into the build directory. The directory allows different build trees for different architectures.

The configuration process is done with the GNU autoconf system. There is a script configure, which is

BSP	Board	Supports
i386ex	Generic i386EX	timer interrupt, com1,com2
ts_386ex	Technologic systems TS-132	timer, RTC, NE2000, com1,com2
pc386	Generic PC/AT board	timer, com1, com2, VGA, ne2000, wd8003

Table 2—*Here are the BSPs for the Intel processor architecture (i386). RTEMS will run on a variety of i386(ex), 486, and Pentium chips. The pc386 BSP supports the standard PC/AT environment.*

text	data	bss	dec	hex filename
94365	3623	11980	109968	1ad90 base_sp.exe
88541	3591	11980	104112	196b0 hello.exe
71576	3544	11968	87088	15430 minimum.exe
139517	4039	12620	156176	26210 paranoia.exe
95997	3623	12012	111632	1b410 ticker.exe
100093	3655	15980	119728	1d3b0 unlimited.exe

Table 3—*Here are the program sizes for the test programs build for the pc386, with the latest GNU tool set and RTEMS snapshot. Note that these are the sizes of the complete system, including the core thread package and libraries, as well as the device drivers necessary to run this application on a PC/AT motherboard.*

called at the top level of the source kit. The script guesses what you are doing and configures specific files to match. I ensured the GNU tools for the architecture were in my shell's search path and typed:

```
cd build/RTEMS/rtems-
20000118a/confgure--
target=i386-rtems
```

The target flag specifies the target I want to build, `i386-rtems`. When `autoconf` is finished, there will be files and directories in the build directory. Now, you're ready to build a system:

```
make RTEMS_BSP=pc386
```

`make` tells it only to build a system for the pc386 BSP. By default, it would try to build all of the BSPs for that particular architecture. `make` is a dependency-based tool. You specify, via macros and variables, what the dependencies for targets are, and it will build the targets. Once targets are built, rerunning make causes it to check the timestamps on the files, determining if it needs to rebuild anything. Although many versions exist, RTEMS expects GNU `make`.

If everything goes well, `make` will build all of the libraries necessary and build and link test programs for the target BSP. The tests include a Hello World program and programs that test and time some of the subsystems. If the BSP supports multiprocessor systems, it will also build MP-based test programs. The RTEMS pc386 BSP does not support MP.

My initial test build did not work because I ran out of disk space on my laptop. Although I have a 6.4-GB drive on my laptop, I have about 4 GB allocated to my Windows partitions for standard tools and CAD packages. The 2-GB Linux partition usually has space, but because I have other large projects, it wasn't enough for the RTEMS tools and system. You'll need about 64 MB for the tools, 28 MB for the documentation kit, and 210 MB for the source kit and build directory.

I fired up my hardware test system, which has an older version of Linux installed on it, copied the archives there, and tried again. This time it succeeded. That's how I calculated the memory requirements. The memory requirement is for that particular snapshot. Who knows what it will be for another snapshot or the 4.5.0 release. I'm sure if the document for the 4.5.0 is as complete as for 4.0.0, it will provide the sizes needed to build the system (see Table 3).

TEST PROGRAMS

The following is a description (taken from the RTEMS `Readme` file) of test programs that are build.

`[Base-MP]` is a simple two-node multiprocessor application that consists of a single initialization task on each node that prints out their respective node numbers and task IDs. This can be used to test a new MPCI layer because it minimizes the number of packets sent by RTEMS. The test is intended as a starting point for custom developed multiprocessor applications.

`Base-SP` is a single processor application, an initialization task that creates another task. The application is a starting point for custom developed single processor applications.

Listing 1—*Here's the Hello World Program application program source. This is one of the smallest applications. It requires the core libraries of RTEMS and a console drive in the BSP.*

```
/*  Init
 *
 *  This routine is the initialization task for this test program.
 *  It is called from init_exec and is the responsible for creating
 *  and starting the tasks that make up the test. If the time of day
 *  clock is required for the test, it should also be set to a known
 *  value by this function.
 *
 *  Input parameters:  NONE
 *
 *  Output parameters:  NONE
 *
 *  COPYRIGHT (c) 1989-1999.
 *  On-Line Applications Research Corporation (OAR).
 *
 *  The license and distribution terms for this file may be
 *  found in the file LICENSE in this distribution or at
 *   http://www.OARcorp.com/rtems/license.html.
 *
 *  $Id: rtems1.txt,v 1.8 2000/02/09 05:05:46 cyliax Exp cyliax $
 */

#define TEST_INIT
#include "system.h"
#include <stdio.h>

rtems_task Init(
  rtems_task_argument ignored
)
{
  printf( "\n\n*** HELLO WORLD TEST ***\n" );
  printf( "Hello World\n" );
  printf( "*** END OF HELLO WORLD TEST ***\n" );
  exit( 0 );
}
```

CDTEST is a simple C++ application that demonstrates how it is possible to use C++ constructors and destructors in an RTEMS application. Also, it performs a perfunctory I/O stream test.

Hello is the RTEMS version of the classic Hello World program. The build test programs consist of a single initialization task, which prints a few messages. Hello test does't include a clock tick device driver and can be used to test the start-up code of the board-support package and the console output.

Paranoia is a public domain test of the floating-point and math library capabilities of a toolset. It reports discrepancies between actual and expected results.

Ticker is a test of the user's clock tick device driver. This test has an initialization task to create three application tasks that sleep, periodically wake up, and print the time.

Listing 1 shows the Hello World program. The build sizes for the test program are stated in Table 3. The test programs are built as ELF binaries. You can configure the tools to generate COFF file format, but it may not work with tools like BDMs or ICEs, which expect COFF format.

Next month's column will look at what it takes to run one of these programs on the pc386 platform.

MORE

Some of the features in RTEMS 4.5.0 are support for the GoAhead Webserver, microwindows GUI, and ATT omniOrb packages. I'll tell you when this release is available.

The difficult part was building and obtaining the tools necessary to build the RTEMS release. A CD-ROM that includes a Linux and Windows-based run-time environment for the tools would have been helpful to me. Considering the time it takes to hunt down the pieces, I'm sure whatever the cost for producing such a distribution is worth it.

In the next few articles, I will look at how to get code down into a target system to run, as well as some of my own applications under RTEMS. I'm itching to bring up a networked environment running on a PC/104 stack, but I'll have to find my PC/104 Ethernet card first.

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

SOURCES

GNU Tools and cygwin Windows-based GNU environment
Red Hat, Inc.
(800) 454-5502
www.redhat.com
sourceware.cygnus.com

Snapshot releases
On-Line Applications Research Corp.
(256) 722-9985
Fax: (256) 722-9985
www.oarcorp.com

Fred Eady

Under the Covers

Part 2: Applications via NT Embedded 4.0

If last month's introduction to Windows NT Embedded 4.0 got you wondering what kind of applications are possible, you're not the only one. Fred took the advice of some readers and considered Internet device control via emWare.

I'm not mad at Bill Gates anymore. A Microsoft Embedded Windows NT 4.0 product manager called me last week, and we talked about the good things in Windows NT 4.0 Embedded.

I explained what I needed and why. The Microsoft product manager explained that shipping me the raw Embedded NT 4.0 code would "open Microsoft's kimono" as far as the product is concerned. Apparently, there's no way to ship an abbreviated Embedded NT 4.0 package without shipping the entire set of binaries. I wasn't prepared to pay the price for a full license, so I took his word for it.

Take a look at Figure 1. Windows NT 4.0 Embedded is the Windows NT 4.0 Workstation and Server code broken down into individual and selectable objects. An object is like an icon on the desktop with the program behind it. For instance, the calculator program is considered an object. Each object is a working chunk of code. In the case of the calculator, the code chunk is the calculator application. These code chunks are kept as binary images (under the kimono) in the binary repository.

Each code chunk can be individually selected to build a unique image of Windows NT 4.0 Embedded. That means you can build an NT image that only contains the kernel and notepad, if that's all your application requires. The image generator makes sure that the combination of objects you select will work as an NT OS before it presents an image. The Microsoft product manager asked me to make it clear to readers that this object selectability is the real strength of the Windows NT 4.0 Embedded product.

The Microsoft product manager also told me that Microsoft DOS, Win3.x/95/98 can be binary components. Microsoft calls it something else, but he said that it's a lot like the Embedded NT process. That makes sense. Microsoft always said that any Windows product is a set of programs and utilities tied together to make an operating system.

Our conversation left me with better feelings for Microsoft. There are real people working for Bill Gates, and I chewed the fat with one of them.

PROMISES

In my last article, I promised I would do something with the embedded OS. I decided to revive a subject that I received many e-mails about—Internet device control with emWare. And, this time around, no web browsers and no Java-based browser programs. I'll present a part of emWare that you may not know about yet, but will appreciate.

Did you know that you don't have to use a Java web browser to control an EMIT-enabled device? Did you know that you don't have to be a C coder to work without a browser? Can you program in Microsoft Visual Ba-

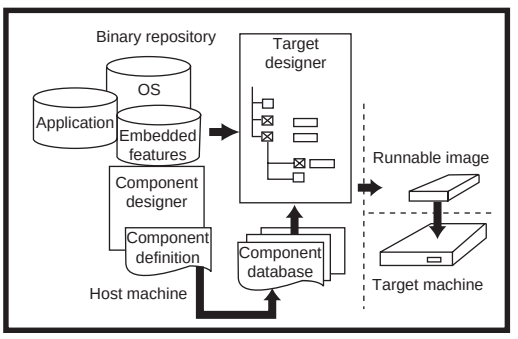


Figure 1—*The process of generating a unique Windows NT 4.0 Embedded image is much like working with Visual C++ and Visual Basic.*

sic, Delphi, or Borland C++ Builder? Then, you can write a custom application to interface with the EMIT-enabled device code you're already putting into microcontrollers. This view of emWare opens a new world of control over the Internet.

DARK SIDE OF THE MOON

Before astronauts orbited the moon, people wondered what was on the dark side of the moon. Although people never saw it, we thought the dark side existed. It was an unknown until we investigated it. That's how I felt about the application interface of emWare. I always knew it was there, but what would I find if I entered its domain?

emWare uses EMIT (Embedded Micro Internetworking Technology) to turn devices into intelligent, interactive object servers. As you can see in Figure 3, emMicro is the heart of the server, with emNet taking charge in the physical network layer. emMicro is a small piece of code that is blended into the host device's main application. emMicro uses variable tables and functions to export and import data used and generated by the main device application. Using Microtags, this information is moved in and out of the EMIT-enabled device. Microtags are condensed data packets that keep the bandwidth utilization low between emNet in the server device and emNet at the gateway.

emNet is the physical entity of emMicro and is responsible for getting the data packets into and out of the server device. Currently, RS-232, RS-485, and Ethernet are supported by emNet.

The Microtags enter and leave the server through emGateway. em-Gateway can reside at the server device level, but more likely, on a larger, more powerful computing platform. Figure 4 shows that emGateway is made up of emNet, a DLM (Device Link Module), DAS (Device Access Service), and an HTTP server, which is augmented by emObjects if necessary. For this discussion, forget about the HTTP server because this is not about browsing.

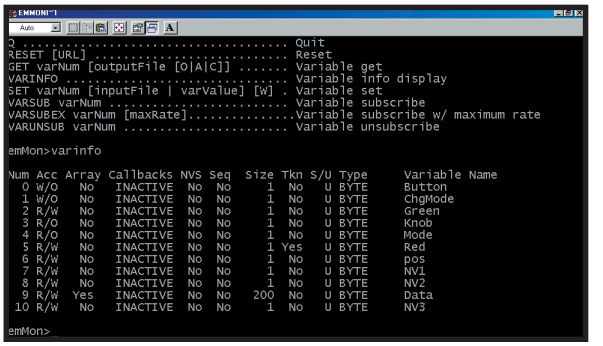


Photo 1—*You can use the emWare doc to sort this out, but note that this call gives you the number of variables, their indexes, and names.*

Here's how it all fits together. Assume emGateway is loaded on your personal computer, and you are using RS-232 connected to the sdkboard that comes with the EMIT Software Development Kit. When the OS starts, so does emGateway. A connection is requested by emClient, which in this case is residing on the emGateway machine. DAS invokes an RS-232 DLM, and a connection is established between the DLM and the sdkboard (the server device) via emNet. Once the logical connection is made, data flows between the external application and the sdkboard via emGateway.

HAL OR EAL?

It's EAL or EMIT Access Library. HAL is Windows NT for Hardware Abstraction Layer. I'm interested in using the functionality of EAL to make the application talk to the EMIT sdkboard. EAL is a C communications library that allows any C-based application that can call a Windows DLL to interface with an EMIT-enabled server device. EAL is a Windows DLL that can operate under the Windows platform. To use the functions within EAL, include the header files and linker definitions in the C project. Listing 1 is a code snippet example of how to apply EAL using C.

After compilation and linking, the rest of the code you don't see in Listing 1 becomes a program called emMonitor, which runs in a DOS window. It uses calls to EAL to demonstrate how an EMIT-enabled server device can be accessed using EAL services with a homegrown C application. Photo 1 shows an emMonitor talking with the emWare sdkboard.

OH SAY CAN YOU C

If you can't get C, get Xemit. Xemit is the ActiveX counterpart of EAL. Within Xemit, Visual Basic (VB), Delphi, and Borland C++, coders will find all of the functionality of EAL without the aftertaste of C. I'll use this to embed a VB application in Windows NT 4.0 Embedded.

Xemit provides connection, information, variable, function, event, file, and error services.

Connection service includes the connect, connected, and disconnect methods, as well as the host, Devicepath, and port properties. Connection service methods and properties are responsible for enabling communication with an EMIT-enabled device, like the sdkboard.

Information service pulls device IDs and the names and types of variables from the EMIT-enabled server device. And, you can get information about the functions, events, and files residing on the EMIT-enabled device you query with the GETXXX methods of information services.

Variable service is where the data gets manipulated. You can get, set, and monitor variables on the EMIT-

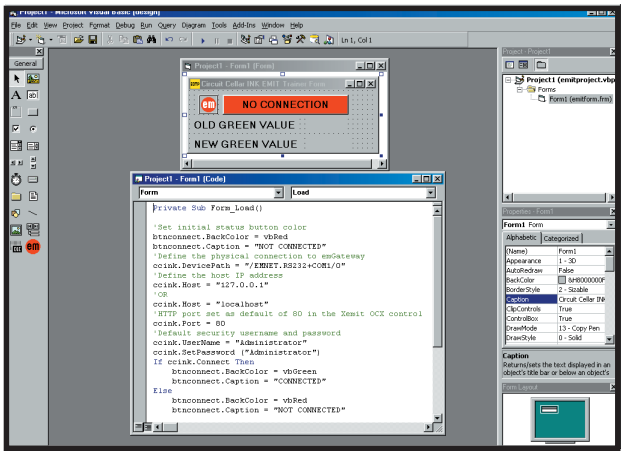


Photo 2—*I changed the intensity of the green LED on the sdkboard with these lines of code.*

enabled device using calls in variable service.

Function service invokes a function on an EMIT-enabled device and gets a return value.

Event service deals with clearing, enabling, disabling, and monitoring device events.

File service lets you store stuff in serial EEPROM on the EMIT-enabled server device or use the EEPROM like a floppy or hard disk. You can read, write, delete, get, and put files with the routines in file services.

Errors happen, and it's good to be able to cope with them. emWare's error service is not fancy, but it's adequate enough to help you figure out what you did wrong.

That's what you have to work with. There are too many methods and properties within the confines of the services to list here.

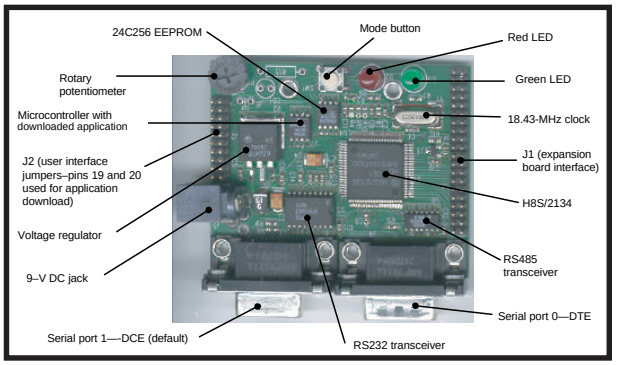


Figure 2—*This picture speaks for itself.*

CLOSE ENCOUNTERS

Before you can contact the EMIT sdkboard, you have to know where it is. Locate and identify the sdkboard to emGateway by defining the Devicepath property. If security is invoked, you must supply a username and a password, using the UserName property and SetPassword method.

Photo 2 shows a code I put together using Visual Basic 6 to show how easy it is to get the emWare sdkboard to work. You'll find examples of some of the service calls in this code.

The first step is to install the `Xemit.OCX` control in the VB6 project. I took a wrong turn getting this included, so I'm showing you where to find it (see Photo 3a). The `Xemit.OCX` file is part of the emWare standard load and is found in the `\emWare\bin` directory. Once I loaded `Xemit.OCX` control into my project, I immediately put it on the main form (see Photo 2) and named the Xemit instance `ccink`. Adding this control opened up all of the functionality found in EAL in Visual Basic format.

First, where is the EMIT sdkboard in the TCP/IP? The Devicepath describes the physical connection the server device uses to get to emGateway. In its basic configuration, the sdkboard wants to use RS-232 and COM1. Thus, the Devicepath syntax is:

Object.Devicepath[= StringEXP]

Object is equivalent to *ccink* and *StringEXP* is defined as:

"/EMNET.RS232+ COM1/0"

To keep it simple, I'll use the Windows default IP address of 127.0.0.1. This IP address will be the host address. Substitute host for Devicepath in the syntax for the IP statement, and you have:

ccink.Host="127.0.0.1"

The host property is used by the connect method to locate the emGateway. You may use a host name that associates with the IP address. By default, the 127.0.0.1 address

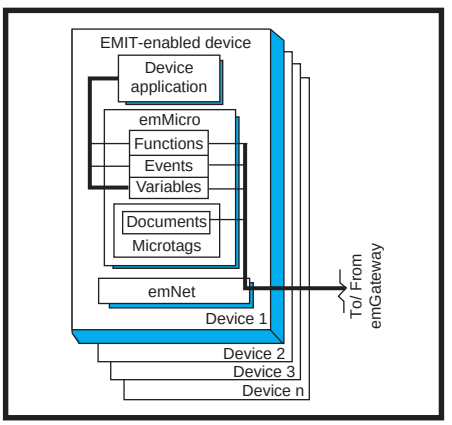


Figure 3—*The device application does a round-robin time share with the emMicro kernel. This approach makes you write tight application code.*

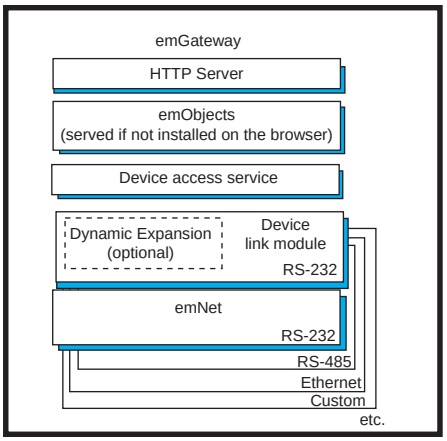


Figure 4—*Microtags are expanded in the DLM if necessary. Otherwise, the data flow is similar to the OSI model. Stuff comes in the bottom at emNet and exits at DAS or the HTTP server, and vice versa.*

is also known as localhost. I included both host statements in my code, separated by a commented OR. In this case, that works because they are the same. Don't try this in real code because you may code in the wrong IP address or misspell the host name. It retains the last one it sees.

In the host syntax explanation, it states that a port address must be specified to point to the port that the emGateway HTTP port is running on. I challenged this by eliminating the default port address of 80 in the Xemit control. The connection to the sdk-board was not made. I included the port statement for clarity here.

Photo 3b shows the security screen that appears when you run emManager. emManager is a program that lets you manually make the DLM and device connections. The username and password are defaults. Because you won't run emManager manually to establish a connection here, you must include the password and username in the VB6 process (see Photo 2). The existence of a password and username implies that the administrator user is on the EMIT access list. If an access list exists, the username and password must be defined before the connect call is made.

Now, you have satisfied all of the requirements to make the call. You defined Devicepath, host, and port. The connect method is Boolean and returns "true" if a connection is successful and "false" if the connection fails. The adventure begins with

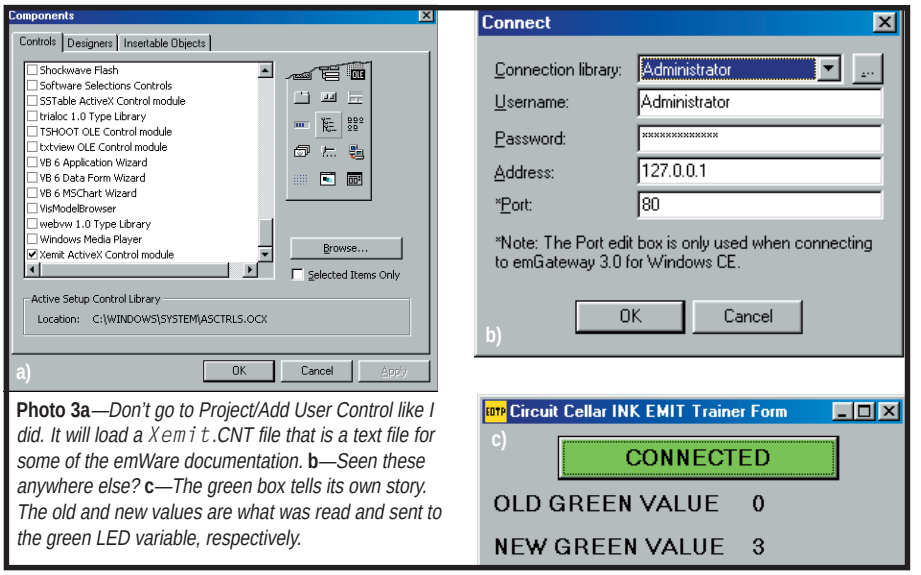


Photo 3a—*Don't go to Project/Add User Control like I did. It will load a* `Xemit.CNT` *file that is a text file for some of the emWare documentation.* **b**—*Seen these anywhere else?* **c**—*The green box tells its own story. The old and new values are what was read and sent to the green LED variable, respectively.*

`ccink.connect`. If it goes well, the red "not connected" button turns green and gives the high sign (see Photo 3c). The rest of my program plays tricks with the service calls.

This will be clearer when I tell you about the sdkboard hardware, and where I got the variable that I manipulated in the VB program.

SURFING ON THE SDKBOARD

Take a look at Figure 2. I used this emWare-supplied board because it shows you more as a general-purpose board than an EMIT-enabled board I would design for a singular purpose. Everything, including preloaded apps and variables, can be found in the board's silicon. All of the internals are completely documented and accessible, which means you can use this device for preliminary design work.

The emWare sdkboard sports a Hitachi H8S/2134 microcontroller running at 18.432 MHz, 4 KB of RAM, 32 KB of EEPROM, and 128 KB of onboard SRAM. The gold-plated pins mean expandability. There is addressable RS-485 and 115.2-kbps RS-232 available with IRDA support built in. This is the total package. All of this embedded horsepower draws 110 mA at 9 VDC.

Take a look back at Photo 1. This is a screenshot of emMonitor with the VARINFO command invoked. Note the variables red and green. These are physical LEDs on the sdkboard. Each physical LED is represented by a variable in the sdkboard application firmware. Photo 2 gives you enough detail to determine what you can and cannot do to variables on the EMIT sdkboard. There are no secrets about the Hitachi-based sdkboard. A full description of how everything works, including schemat-

ics, is included with the emWare development package.

THE GOOD PART

Imagine the sdkboard as an embedded platform that you designed. Next, imagine the VB6 example code is your application. Compile it and install it on the NT Embedded machine. Then, install emWare at the Embedded NT level. Now, you have an EMIT-enabled Windows NT 4.0 Embedded client that can interact with the EMIT-enabled sdkboard you designed.

Here's the good part. VBA stands for Visual Basic for Applications, and all Microsoft office applications can use it. VBA is used as a scripting language, allowing developers to interact with and customize functionality in applications like Excel and Word.

How does VBA fit here? EMIT-enabled servers work with variables, functions, and documents. Once you retrieve a variable from your EMIT-enabled device using Visual Basic for the application and employing VBA calls at the Microsoft app end, you can write VB code to put that EMIT-server-generated variable directly into an Excel spreadsheet. Need to add EMIT-server data to a database? Microsoft Access and Excel have the same VBA/Visual Basic functionality.

Using Windows NT 4.0 Embedded allows you to leverage the power of off-the-shelf Microsoft applications. Add emWare to the mix, and you have Microsoft application power across the Internet. A few years ago, it was difficult to design and implement a remote sensing device that could report back to a dedicated PC, assemble the data into spreadsheet format on the fly, and then add the complexity of attempting to get the data from the dedicated PC to a remote viewing point. Today's solution is EMIT enabling at the sensing device, EMIT-enabled NT Embedded loaded with Excel, custom VB code at the gateway, and a web browser at the viewing point—that's any viewing point, anywhere on any EMIT-enabled device.

Listing 1—*This is a great tool for people who need guidance using C. I wanted to show the* `includes` *here.*

```
/*****************************************************************
;                  emMonitor.c - emWare Monitor Utility
;              Copyright (C) 1998 emWare Corporation
;                        All rights reserved
;*****************************************************************/
#include "SysDep.h"
#include "EMCLIENT.H"
#include "emCon.h"
#include "ClientTypes.h"
#include "SockUtils.h"

/*****************************************************************
;  emMonitor.c - s - EMM_ATR_... Attributes
;*****************************************************************/
#define EMM_ATR_ARRAY  0x80
#define EMM_ATR_SIGNED 0x40
#define EMM_ATR        ~(EMM_ATR_ARRAY | EMM_ATR_SIGNED)
#define EMM_CMD_START  5

/*****************************************************************
;  emMonitor.c - #DEFINEs - EMM_TYP_... EMM_CMD_TABLE "emmType" values
;*****************************************************************/
#define EMM_TYP          0x00
#define EMM_TYP_ARRAY    0x80
#define EMM_TYP_DISPLAY  0x40
#define EMM_TYP_EVENT    0x20
#define EMM_TYP_FILE     0x10
#define EMM_TYP_FUNCT    0x08
#define EMM_TYP_NUM      0x04
#define EMM_TYP_QUIT     0x02
#define EMM_TYP_VAR      0x01

/*****************************************************************
;  emMonitor.c - #DEFINEs - EMM_TXT_... ASCII Text
;*****************************************************************/
#define EMM_TXT_DEV_INFO  "\
\n  Device Id ......... %d\
\n  Manufacturer Id ... %d\
\n  Product Id ........ %d\
\n  Dynamic Doc count . %d\
\n  Event count ....... %d\
\n  Function count .... %d\
\n  Static Doc count .. %d\
\n  Variable count .... %d"
#define EMM_TXT_PROMPT    "\n"EMC_TXT_PREFIX"Mon>"
```

I'm not done with Windows NT 4.0 Embedded. UPS dropped another blue box at the Florida-room door. I hope that by visiting the dark side of emWare, I turned on some LEDs in your head and again have proven that it doesn't have to be complicated (or a full-version of Embedded Windows NT) to be embedded.

Fred Eady has more than 20 years of experience as a systems engineer. He has worked with computers and communication systems large and small, simple and complex. His forte is embedded-systems design and communications. Fred may be reached at fred@edtp.com.

SOURCES

EMIT
emWare
(801) 256-3883
Fax: (801) 256-9267
www.emware.com

Windows NT 4.0 Embedded
Microsoft Corp.
(206) 882-8080
Fax: (206) 936-7329
www.microsoft.com

MICRO SERIES

Op-Amp Specifications

Getting Some Input

Joe DiBartolomeo

Part 2 of 4

In Part 2 of this Micro-series, Joe discusses specifications for input offset currents and voltages, as well as input bias current. If low-frequency and precision applications are your thing, you won't want to miss this.

This month, I'll look at common op-amp specifications that affect DC and precision applications. These specifications are bias currents, offset current and voltage, common mode rejection ratio (CMRR), and power supply rejection ratio (PSRR). I'll explain their origin and how they affect circuits.

A good place to start is the differential pair (see the "Differential Pair" sidebar). Most of the parameters stem from the differential pair, which is the basic input structure for op-amps. During the past 20 years, op-amp input has changed significantly. I will stick with the classic model, which best illustrates these DC parameters.

The ideal op-amp model is great for a first pass, however, it ignores the specification I'm dealing with. You may want to work with the three-stage op-amp model I presented last month—the differential pair, followed by the transresistance stage with frequency compensation, followed by the output stage. This illustrates the op-amp well and demonstrates how it will behave in a circuit.

BIAS CURRENTS

The ideal op-amp has no current flowing into its input terminals. However, note the differential pair shown in the sidebar, where currents flow into the op-amp input terminals. These currents are known as bias currents I_B+ and I_B-, and they flow in the external op-amp circuit.

The manufacturer's datasheet specifies the bias current as the average (magnitude) of the two input bias currents when they are tied together.

$$I_B = \frac{I_{B+} + I_{B-}}{2} \quad [1]$$

With a perfectly matched differential pair, I_B+ would equal I_B-. Then the effects of bias currents can cancel out. However, because bias currents are rarely equal, you must define another specification, input offset current, I_{OS}, which is the difference between the two bias currents. This is an important specification because it's a measure of the amount of mismatch in the differential pair. It is difficult to compensate for the effects of I_{OS}.

$$I_{OS} = I_B+ - I_B- \quad [2]$$

Like all op-amp specifications, I_B and I_{OS} depend on technology and topology. For a typical bipolar op-amp, the bias current is in the 10s of nA, while for FET op-amp's bias currents are in the 10s of pA, with some below 1 pA. However, the bias cur-

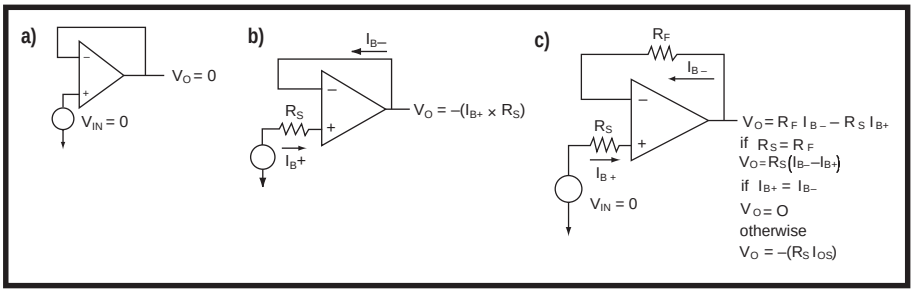


Figure 1—*This shows the effects of bias currents on a buffer amplifier,* ***(a)*** *is ideal,* ***(b)*** *considers bias current, and* ***(c)*** *shows bias current compensation.*

rents in FET op-amps double every time there is a 10°C rise in temperature. The bias currents in bipolar devices are temperature independent. Be aware of the bias currents when dealing with bipolar op-amps, and note the drift in bias currents when handling FET op-amps.

Op-amp	TLE2037I	TLE2071	TLC2201
Technology	Bipolar	BiFET	CMOS
Ib max	150 nA	5 nA	150 pA
Ios max	90 nA	1.4 nA	150 pA
Drift dIb/dT	Minimal	Double every 10°C	Doubles every 10°C

Table 1—*Comparison of op-amp technologies versus bias currents and offset current.*

Table 1 shows specifications for maximum bias and offset currents for three op-amp technologies. Design with the maximum specification, rather than the typical. For example, the typical offset current of the TLE2071 is 15 pA, while the maximum is 1.4 nA. When dealing with

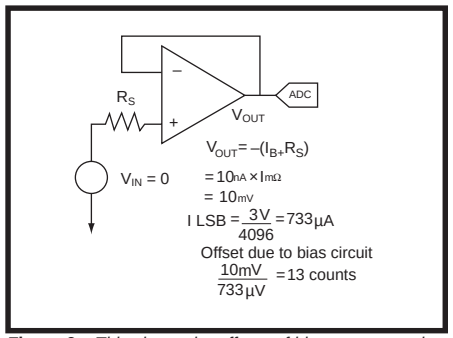


Figure 2—*This shows the effects of bias current on the output of the buffer op-amp.*

Murphy's Law, which one will your op-amp exhibit?

Generally, a faster op-amp means a higher bias current. High-speed designs are more AC than DC applications. In AC work, the input to the op-amp may be capacitively coupled. This is acceptable if a DC return path for the bias current is provided.

OK, now let's take a look at three basic op-amp configurations to understand the effects that bias currents have on op-amp circuits. I'll demonstrate with the follower, noninverting, and inverting amplifier circuits, assuming that the bias currents are the only cause of error.

In the ideal follower circuit, because V_{IN} is zero, V_{OUT} should also be zero (see Figure 1a). Redrawing the circuit and accounting for the bias currents shows that this is not true (see Figure 1b). The bias current flows in zero external resistance into the negative terminal, causing no external voltage drop. But, the positive bias current flows in the source resistance. This makes the noninverting input $-R_S I_B$, which places the inverting terminal at the same potential. Therefore, the output is:

$$V_O = -(I_{B+} \times R_S) \quad [3]$$

Suppose the source resistance is 1 MΩ and the bias currents are 10 nA (see Figure 2). When $V_{IN} = 0$, $V_{OUT} = -(R_s I_{B+})$, which, in this case, is 10 mV. If you have a 12-bit ADC with a 3-V range, 1 LSB = 733 µV. Essentially, the op-amp added an offset of 13 counts to the ADC, 10 mV/LSB.

Let's look at another example, this time using the inverting op-amp configuration (see Figure 3a). Here, I lumped the series and source resistances together and assumed that the op-amp output resistance is small compared to the feedback resistor R_F. Again, set the input to zero. In the ideal op-amp model, V_{OUT} equals zero, but as a result of bias currents, the output voltage is, in turn, not zero.

The bias current flowing into the noninverting terminal flows through zero resistance and doesn't cause a voltage drop, therefore the noninverting and inverting terminals both equal zero. This means that there is no current flowing in the series resistor R_S. Therefore, the bias current going into the inverting terminal must be supplied from the op-amp output terminal and flow in the feedback resistor. So, $V_{OUT} = I_{B-} R_F$.

The same exercise can be done for the noninverting amplifier configuration.

BIAS CURRENT COMPENSATION

You can use the bias currents to compensate for their own effects. In the buffer circuit, you can use I_B-, which flows through zero resistance to help cancel the effect of I_B+ (see Figure 1a). Place a resistor equal to R_S in the feedback path (see Figure 1c). If $I_B+ = I_B-$ and $R_F = R_S$ when $V_{IN} = 0$, $V_{out} = 0$. As I stated earlier, I_B+ rarely equals I_B-, so the output voltage will be equal to $R_F I_{OS}$.

For the inverting op-amp, add a resistor to the noninverting terminal that is equal to the parallel resistance of R_S and R_F (see Figure 3b). Like the follower circuit, if $I_B+ = I_B-$, the bias currents cancel out. However, this is rarely true, $V_{OUT} = R_F I_{OS}$. Because I_{OS} is normally 10 to 25% of I_B, there is a significant reduction in I_B effects.

It is easy to determine the current compensating resistor needed for the follower or inverting circuits. What if the external circuits are more complex? To minimize the effects of I_B, follow this rule: the DC resistance seen by the noninverting terminal WRT ground should equal the DC resistance seen by the inverting terminal WRT ground. When doing this calculation, be sure to include the source resistance. Also, the output resistance of the op-amp usually can be ignored, so consider the output of the op-amp to be at ground potential.

Note that usually bias current compensating resistors are effective only when the bias currents are well matched and flowing in the same direction. However, some op-amps have bias currents that aren't well matched and flow in opposite directions (see Figure iv). With these, bias current compensating resistors may enhance bias current effects.

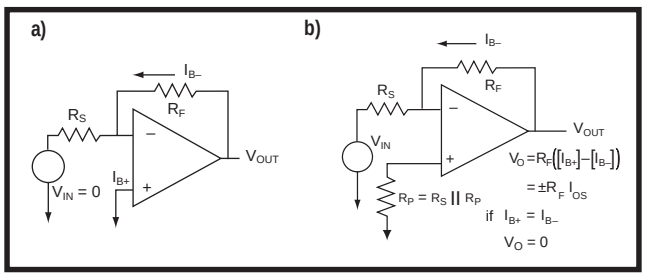


Figure 3—*Here are the effects of bias currents on an inverting amplifier,* ***(a)*** *considers bias current, and* ***(b)*** *shows bias current compensation.*

Type	V_{os}	Drift (µV/C)
BiPolar	10 – 25 µV	0.1
BiFet	0.3 – 1.5 mV	5
CMOS	80 – 200 µV	0.5

Table 2—*Comparison of op-amp technologies versus input offset voltage.*

In general, if you use FET op-amps, don't worry about bias currents, except for the temperature effects. Thus, bias current compensation resistors are not needed.

In bipolar applications, keeping the source and feedback resistors as small as possible reduces bias current effects. When a low-gain circuit is driven by a high-level signal, bias current usually won't be an issue. Conversely, in bipolar circuits, when a high degree of precision is required, bias currents are often an issue.

INPUT OFFSET VOLTAGE

You've seen how imbalances in the op-amp input stage cause offset bias currents. These imbalances also cause an input offset voltage. If you connect an ideal op-amp, the output will be zero (see Figure 4a). However, actually the output will tend towards one of the supply rails. The voltage required to bring the output to zero is the input offset voltage V_{OS} (see Figure 4b). Like bias currents, the values of V_{OS} vary depending on the op-amp topology and technology (see Table 2).

It appears easy to compensate for V_{OS} by nulling. Nulling means injecting a current that will act in opposition to V_{OS} into the summing junction. Many op-amps have V_{OS}-nulling terminals. However, it is not simple because V_{OS} varies with temperature. The nulling will be valid only at that one temperature. And, remember that V_{OS} ages approximately 3µV per year.

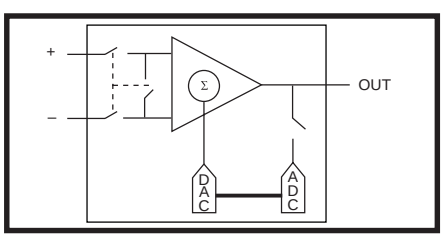


Figure 6—*In a self-calibrating op-amp, input switches allow the ADC to measure V_{OS}. V_{OS} is applied via a DAC to the summing junction to reduce its effects.*

Let's look at how V_{OS} affects a circuit. V_{OS} is no different than other differential voltages and is amplified by the gain of the circuit (see Figure 5). In high-gain circuits, the V_{OS} affects it significantly. Revisiting Figure 2, notice that V_{OS} would also cause an offset in the ADC reading.

Bipolar op-amps have the lowest V_{OS}. However, applications that are sensitive to V_{OS} are usually sensitive to I_B. CMOS is better than bipolar when considering I_B, but worse when considering V_{OS}.

There are a couple of CMOS op-amps that give low bias currents and good V_{OS}. The first option is the chopper-stabilized op-amp. A chopper converts the DC signal to AC, and then back again to DC. In the process, DC error and temperature effects are minimized. A chopper amp can get V_{OS} of less then 1 µV, with minimal temperature effect. But, choppers come with external component requirements and produce switching noise. Another solution is to use a self-calibrating op-amp, such as the TLC450*x* family of Self-Cal op-amps from Texas Instruments. This op-amp measures the V_{OS} with an ADC, and then uses a DAC to inject a nulling current (see Figure 6).

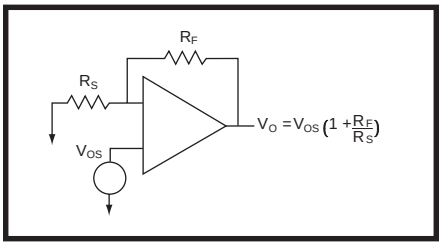


Figure 5—*V_{OS} is amplified like all differential signals. When the gain is large, it can cause significant errors.*

CMRR

Ideally, the op-amp's output voltage is a function of the differential voltage across its inputs and the amplifier's differential gain:

$$V_O = A_D V_D \quad [4]$$

where V_D is the differential voltage and A_D is the differential gain. The common mode signal is rejected. Because of the mismatches in the op-amp's differential pair, there will be a portion of the output that is a function of the common mode voltage:

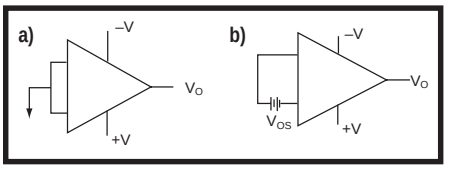


Figure 4—*This shows the effects of V_{OS}. In the ideal op-amp* ***(a),*** *the output voltage is zero. In a practical op amp* ***(b),*** *the output will tend to one of the rails. V_{OS} brings the output back to zero.*

$$V_O = A_D V_D + A_C V_C \quad [5]$$

where V_C is the common mode voltage and A_C is the common mode gain. This leads to the familiar definition for common mode rejection ratio of:

$$CMRR = \frac{A_D}{A_C} \quad [6]$$

and

$$A_C = \frac{A_D}{CMRR} \quad [7]$$

Substituting [7] for [5], you get:

$$V_O = A_D V_D \left(1 + \frac{V_C}{V_D \times CMRR}\right) \quad [8]$$

Let's take an 80 dB-CMRR op-amp (a voltage ratio of 10,000) and apply two sets of signals with the same differential voltage, but different common mode voltages. The value of the common mode voltage affects the output voltage (see Figure 7).

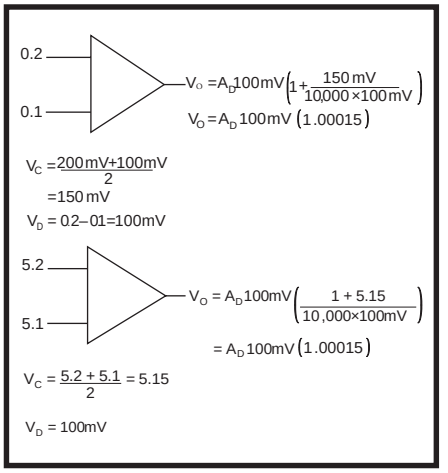


Figure 7—*Ideally, common mode voltage changes that don't affect the differential mode voltage don't affect output voltage. But, calculations prove that common mode voltage changes do affect output voltage.*

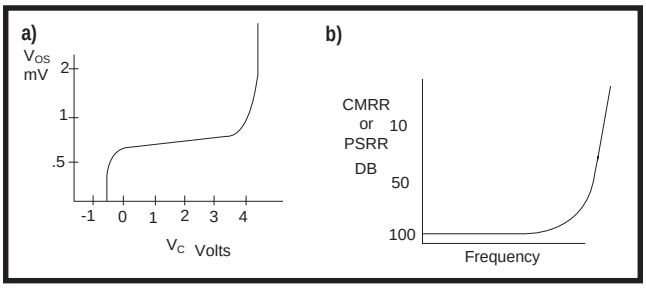


Figure 8—*The input offset voltage is a function of the common mode input voltage, because a common mode voltage change alters the differential pair's biasing point, which alters V_{OS}.*

The common mode voltage sets the operating point for the input transistors. Due to mismatches in the differential pair, different common mode voltages result in different V_{OS}. Op-amp spec sheets give curve, showing the relationship between V_{OS} and common mode voltage, V_C (see Figure 8). One way to determine CMRR, is to measure the change in V_{OS} for a change in V_C.

In addition, Equation 8 highlights the fact that as CMRR increases, the effect of the common mode signal is reduced. Looking at Figure 9, you notice that CMRR depends on frequency; as frequency increases, the op-amp's capability to reject a common mode signal decreases.

In Figure 9, V_C is the input voltage, so the output is a combination of V_{IN} and V_{IN}/CMRR. Here, there is no error due to common mode. Because both inputs are at ground, common mode voltage is zero.

PSRR

In the ideal op-amp model, if the supply voltage changes, it doesn't affect the output voltage. But, in practice, this is not true. A change in the power supply causes a change in the bias point, and therefore causes an input offset change.

$$PSRR = \frac{dV_{CC}}{dV_{OS}} \qquad [9]$$

Normally, this is expressed in decibels. For dual supply op-amps, you assume the supplies vary equally, otherwise a common mode change will occur. In the lab, you can vary the power supplies symmetrically, but this doesn't happen in practice.

What does PSSR do to your circuit? Imagine you have a battery-powered system and the rails on your op-amp change ±10% from the nominal 3 V, ranging from 2.7 to 3.3 V. The change in supply is 300 mV. If the PSRR is 100 dB, it causes a 3 µV V_{OS} change. But, if the PSRR is 80 dB, V_{OS} will change by 30 µV. Look at Figure 5 and

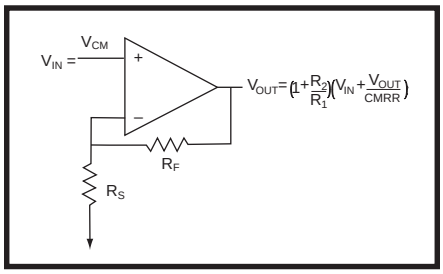


Figure 9—*In this non-inverting application, CMMR is a problem because $V_{IN} = V_C$. However, in the inverting topology CMRR is not a problem because $V_C = 0$.*

Differential Pair

The basic differential pair, which is often referred to as the long-tail pair, is shown in Figure i. The ideal differential pair will infinitely amplify any voltage that is differential to its inputs and completely reject any voltage that is common to its inputs.

The ideal differential pair exhibits all the characteristics required of an ideal op-amp's input stage:

- differential mode gain = infinite
- common mode gain = 0
- input bias current = 0
- input resistance = infinite

The differential pair circuit is powerful, yet simple. Note that when applying a signal to the inputs of the differential pair, you get the output voltage given by equation E1 (see Figure ii).

$$V_{OUT} = A_D V_{DM} + A_C V_{CM} \qquad [E1]$$

where A_D is the differential gain and V_{DM} is the differential voltage applied across the inputs. The product is referred to as the differential output voltage V_{OUTDM}. The ideal op-amp model requires this value be infinite. The practical goal is to get the highest A_D possible, while still operating linearly.

Also in E1, A_C is the common mode gain and V_{CM} is the voltage common to both inputs. The product of these two terms is referred to as the common mode output voltage V_{OUTCM}. In the ideal op-amp model, the differential pair is perfectly matched, $V_{OUTCM} = 0$. However, in operation, there is a mismatch between the two halves of the pair, causing a common mode output voltage. Think of the mismatch as activity causing a differential voltage between the inputs.

Anything that causes unequal variation in the two halves of the circuit will cause a differential voltage. Because the differential pair is symmetrical, any temperature effect will be second order, especially on the same piece of silicon.

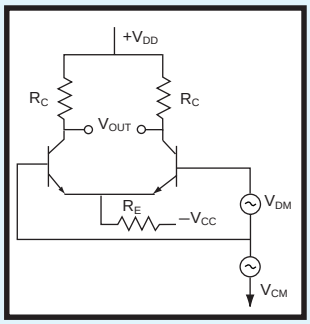


Figure ii— *Applying an input to the differential pair creates an output voltage, which is the combination of the differential voltage and common mode voltage.*

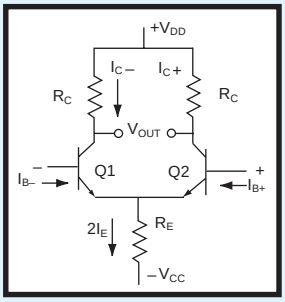


Figure i— *Ideally for use as an op-amp input stage, this infinitely amplifies a differential mode signal and completely rejects a common mode signal.*

Because the differential pair always has common mode output voltage, a figure of merit for the differential pair was established, common mode rejection ratio (CMRR). CMRR is a measure of how close a differential pair comes to the ideal of infinite differential gain and zero common mode gain.

$$CMRR = \frac{A_D}{A_C} \qquad [E2]$$

Let's apply a common mode voltage ac to the differential pair in Figure i. When the differential pair is symmetrical, the Q1 and Q2 branches are perfectly matched. This leads to base currents I_B+ and $I_B–$ being equal. Therefore, the emitter currents of Q1 and Q2 are equal, the beta of the transistors is equal, and the collector currents I_C+ and $I_C–$ are equal. Because the two collector resistors are equal and have equal current flowing through them, the collector voltages of Q1 and Q2 are at the same potential. Thus, $V_{OUTCM} = 0$. However, there are always mismatches, so V_{OUTCM} will not equal zero. In terms of circuit parameters, the common mode gain is:

$$A_C = -\frac{R_C}{2R_E} \qquad [E3]$$

A mismatch in base currents exists because of the applied differential voltage (see Figure i). This leads to a mismatch of emitter, and therefore, collector currents. Assuming the R_C resistors are closely matched, the difference in collector currents causes a differential voltage between the two collectors, V_{OUTDM}.

Ideally V_{OUTDM} would be infinite; however, the differential mode voltage is:

$$V_{OUTDM} = V_D A_D \qquad [E4]$$

where:

$$A_D = \frac{R_C}{2R_E}$$

where R_E is the familiar intrinsic emitter resistance, $26/I_E$. You can get a representation for CMRR as:

$$CMRR = \frac{A_D}{A_C} = \frac{R_E}{r_E} \quad [E5]$$

(Equations E3–E5 are simplifications of more complex equations.)

The basic differential pair was improved upon in an effort to push its performance closer to the ideal. I'll mention a couple improvements to illustrate where the representation of the differential pair in Figure iii comes from.

You want the differential gain to be large. The easiest way to do that is to increase R_C. However, V_{CC} becomes the limiting factor. If you increase R_C, you need to increase V_{CC} in order to maintain the same I_C. The solution is a current mirror active load, which has high resistance and is required to convert the signal for differential to single-ended.

According to equation E3, increasing R_C increases common mode gain.

To get the common mode gain low again, increase R_E. But, this will change the bias point. You can use a constant current source as a biasing element, which makes the R_E large. Figure iii shows the differential pair, with R_C and R_E replaced by a current mirror and a current source, respectively. These two modifications lead to high differential gain, low common mode gain, and therefore, high CMRR. These attributes are suited for op-amp input stages.

Signal conversion also is suited for use in op-amp input stages. Op-amps convert differential signals to single-ended signals. This is accomplished by taking the output from one of the collectors and feeding it to the next stage. When you make the signals single-ended, you make them easier to handle in the following op-amp stages. Some op-amp topologies keep the signal differential for at least one more stage in order to maximize a particular parameter.

As shown in Figure iii, with no differential input voltage applied, the current source splits equally between the two emitters, assuming no common mode effect. You see that no current flows into or out of the second stage because of the current mirror action.

When a differential voltage is applied, it unbalances the differential pair, thereby causing unequal emitter currents

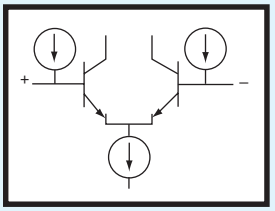


Figure iv—*Internal current sources supply bias currents. Bias currents flowing in the external circuits are reduced, but offset currents are worse.*

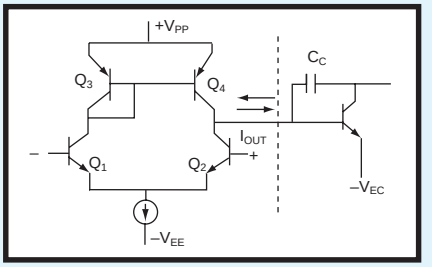


Figure iii—*Here's the differential pair with current source biasing and current mirror active load.*

in Q1 and Q2. This means that the collector currents of Q1 and Q2 are different. As a result of the current mirror, the collector currents of Q3 and Q4 must be equal. This leads to current either flowing into or out of the second stage. The direction of the current depends on which of the inputs has the higher voltage level.

Note that bias currents must flow into the base of the transistor in the differential pair (see Figure i). Because the inputs rarely match perfectly, the input bias currents are rarely the same. This leads to a specification known as input offset current (I_{OS}), which is the magnitudinous difference between the two bias currents.

There is an input offset voltage between the two inputs, V_{OS}. This offset voltage is a result of mismatched inputs and acts as if it were a differential input signal.

How does the differential pair affect slew rate? Slew rate is a measure of the drive capacity per unit of time of the op-amp. If you apply a step to the input, how long before the output reacts? Take a look at Figure iii. This shows that one contributing factor is the current the differential pair can deliver to charge the compensation capacitor in the op-amp's second stage.

Now, the direct correlation between the differential pair's and op-amp's specifications is clear. A great effort was exhausted to improve the differential pair and achieve the ideal op-amp input specification. However, designer tradeoffs complicate the effort.

Bias currents must flow into the op-amp. These same bias currents flow in the external circuits and cause errors. One solution is to supply the bias currents internally, as you see in Figure iv. This reduces the bias currents flowing externally. However, now you not only have the differential pair transistors mismatch, but also a current source mismatch, making the offset current worse than before.

imagine you have a gain of 50 in the circuit. The 3 µV V_{OS} causes an output error of 150 µV, while 30 µV V_{OS} causes a 1500-µV output error.

PSRR, like CMRR, is frequency dependent—the op-amp's ability to reject is reduced as the frequency increases. Figure 8b shows the importance of good bypassing at the op-amp power leads. If you use an op-amp in a circuit where the power comes from a switcher, bypass the power of each op-amp. At higher switching frequencies, you need one quality low-inductance capacitor per rail connected closely to the op-amp power pins.

Now you've seen several op-amp specifications and how they affect circuit performance.

Next month, I'll look at how to measure the bias current's offset voltage, CMRR, and PSRR. Also, I'll explore the output structure and input and output impedence specs.

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

REFERENCES

[1] Horowitz and Hill, *Art of Electronics*, Cambridge Press, MA, 1990.

[2] J. Karki, *Understanding Operational Amplifier Specifications*, Texas Instruments White Paper, 1998.

[3] Coughlin and Driscoll, *Operational Amplifiers and Linear Integrated Circuits*, Prentice Hall, Englewood Cliffs, NJ, 1977.

[4] Miller, *Microelectronic*, McGraw-Hill, NY, 1979.

[5] Texas Instruments, *TI Analog Seminar Handbook*, 1999.

SOURCE

TLC450*x* family of Self-Cal op-amps

Texas Instruments Inc.
(800) 477-8924, x4500
(972) 995-2011
Fax: (972) 995-4360
www.ti.com

Mike Baptiste

HCS=Heating Control System?

It may be Spring now, but one cold snowy day back in January, Mike got to pondering about his new HVAC system and how to tie it into his home-control system. Soon, he had the "H", "V", and "AC" singing in harmony with his HCS.

When I wrote this column, a snowstorm of epic proportion was hammering the East Coast. I live in North Carolina, and we received about 20" of snow during one storm in January. What better time to write about controlling a heating system with a home automation system!

I recently replaced my HVAC system with new high-efficiency units. Because I often get questions from HCS users about HVAC control, this is a great time to write about it.

I'll use the new system for my home's first floor as the basis for the examples in this column. The new system (see Photo 1) is a Trane XV90 Variable Speed/Multistage gas furnace with a high-efficiency air conditioning compressor. The system is split into three zones and has an add-on humidifier. There is a fresh air exchanger tied into the ductwork that exhausts stale air from the house, while bringing fresh air inside. It uses a heat exchanger to recover most of the heat from the outgoing air.

THE BASICS

Most HVAC Systems use 24 VAC for control. Regardless of the type, an HVAC system will feed 24 VAC to the thermostat (red wire), which connects the 24 VAC to whichever control line it wants to turn on (heat, cool, heat pump reversing valves, heat pump emergency heat, fan, etc.). Most add-on devices use the same type of 24-VAC control setup. My system has the following control lines in it: stage one heat (W1), stage two heat (W2), cooling (Y), and fan/blower (G), plus the humidifier has its own 24-VAC feed and humidifier control line controlled by a separate humidistat.

Contrary to popular belief, the fan control is not always turned on when a thermostat calls for heating or cooling. A forced hot-air furnace must reach a certain temperature before the blower is turned on or it will blow cold air. Hence, it controls the blower in heat mode by monitoring the temperature of the heat exchanger. In cooling mode, the thermostat immediately turns on the blower.

Electric heating systems have more immediate heat, and the thermostat should turn on the fan when it calls for heat. Check the thermostat documentation to determine how it operates. Thermostats like the Statnet from Enerzone allow you to select fan control mode. For cooling, the blower is turned on immediately in most systems by the cooling control via the furnace circuitry not the fan control from the thermostat.

With such a simple control setup, an HCS-II or an other home automation system can easily handle HVAC control. However, it's more complicated than wiring a few relays.

HVAC systems don't like to be short-cycled. If an air conditioner is turned off and then on again before the pressure in the compressor subsides, it will stall, potentially damaging it. The newer, high-efficiency furnaces have complex control systems that take into account combustion airflow, presence of flame, and temperature. These need time to cool down between heat calls, as well. Allow at least two minutes between consecutive calls.

In addition to the potential damage to the system, what happens if the home automation system needs to be turned off? What happens if the HVAC control code is broken while

	Standard gas/electric single stage	Standard gas/electric multistage	Heat pump single stage	Heat pump multistage
Single relay input				
2 Heating/2 cooling setpoints	300X1	700X1	450X1	600X1
Single relay input/serial ready				
2 Heating/2 cooling setpoints	300SR	700SR	450SR	600SR
Dual relay input				
4 Heating/4 cooling setpoints	300DI	700DI	450DI	600DI
Serial communicating	300SN	700SN	450SN	600SN

Table 1—*Here's a Statnet product matrix. Consult Enerzone's web site to choose the proper type for your system.*

editing code? What if it is freezing outside? The home could remain cold until the bug is found.

REDUNDANT CONTROL

The good news is that it's easy to control the HVAC system via the home automation controller and still ensure that it will work if the automation system or program fails. Numerous vendors make thermostats that can be controlled externally.

If the external control fails, the thermostat remains operational and takes all necessary precautions, such as preventing short cycling. Photo 2 shows a Statnet thermostat and the RS-485 interface card, which is located directly behind the thermostat or in a wiring distribution panel.

Communicating thermostats operate as a standalone thermostat, but most of the functions performed using its keys can be controlled by a remote system. Changing the system mode and temperature setpoint of a thermostat are common functions performed remotely. Some thermostats use X10 commands to select a setpoint temperature from a predetermined list. Others use dry contact inputs to switch among preprogrammed setpoints. Still others have complete serial interfaces with command sets that let users change most attributes.

A communicating thermostat makes the HVAC control setup portable. When replacing a heat pump with a gas system, simply purchase a new type of thermostat. The controls for the home automation system (heat and cooling setpoints, fan mode, etc.) won't change because they are common among all HVAC systems. Thus, the HVAC system becomes a black box with standard functions from the thermostat outward. This lets the thermostat worry about turn-on delays, heat pump reversing valves, and which control lines to enable under given conditions.

Controlling the HVAC system involves multiple control points and systems. The variety of sensors and status inputs make it easy to develop powerful control algorithms. Figure 1 outlines one of my three HVAC systems. Note that if the HCS-II control section failed, only the humidifier would stop functioning.

THE STATNET

The Enerzone Statnet is an ideal thermostat. It has an RS-485 network interface and supports a command set that lets you change setpoints, turn on the fan, read the temperature, and receive changes of state for various HVAC systems. If the serial interface isn't used, the thermostat supports dry contact inputs, which allows the current setpoint to be changed to a day or night setting depending on the state of a connected relay.

The Statnet line uses a full duplex RS-485 network that isn't compatible with the HCS-II. Next month, I'll cover a PIC-based RS-485 interface that allows the HCS-II to talk to Statnet thermostats. For now, I'll discuss the dry contact interface. The Statnet thermostat comes in different models that are designed to meet the requirements of various HVAC systems. Each model comes in four different versions that indicate the type of control interface (see Table 1).

I recommend getting an SR type, so you can use the dry contact interface now and add a serial interface card later. You also can get the SN type with the RS-485 interface. Disconnect the RS-485 interface, and it will act like an SR with dry contact capability. After I present the RS-485 HCS-II interface, you can reconnect the interface to your thermostat.

ZONING

Until recently, homes with forced-air systems usually had HVAC systems installed so there was one thermostat for the entire house, or at least for an entire floor. In the latter case, there were individual systems with their own thermostats for each floor. However, in larger homes, it is often desirable to keep different sections, served by the same system, at different temperatures. This is accomplished via zoning.

When a house with forced-air HVAC is zoned, different sections are served by dedicated ductwork fed off the furnace/heat exchanger. Each zone's supply duct can be closed off using an air damper similar to the one in Photo 3. Radiant hot water systems use a similar concept—different zones are fed hot water from a central supply, using dedicated zone valves.

Most dampers use a spring to open and a motor to close. Dampers are controlled by a zone control panel that logically decides when an HVAC system should be turned on based on which thermostats call for heating or cooling. If one zone calls for heat, the other zone dampers close so heat goes to the zone calling for it.

Discussing zone control would take pages because it is complicated (e.g., handling one zone that asks for heat while another asks for cooling, handling multi-zone calls with a multistage system, etc.). Again, it is best to use a dedicated zone control panel (see Photo 4).

Figure 1 outlines all the systems involved in my first-floor HVAC system control. The Trol-A-Temp zone panel takes care of combining each thermostat's heat or cooling calls. It also takes care of preventing short cycling, opening and closing the zone dampers, and determining when the second heating stage should be used. The zone panel has a temperature sensor mounted in the supply duct that shuts the system down tempo-

rarily if the air temperature exceeds set thresholds. This can occur when only a small zone calls for air, which reduces airflow and causes air to loop through a bypass damper between the supply and return ducts. During cooling, this can lead to freezing on the cooling coil, and during heating, it can overheat the furnace.

Another positive feature of the Trol-A-Temp panel is its unoccupied input. Connect a relay to this input and close it when the house is unoccupied. The zone panel will monitor the thermostat connected to input 1 and use it to control all zones. Thus, you can setback one thermostat, rather than all of them when the house is unoccupied.

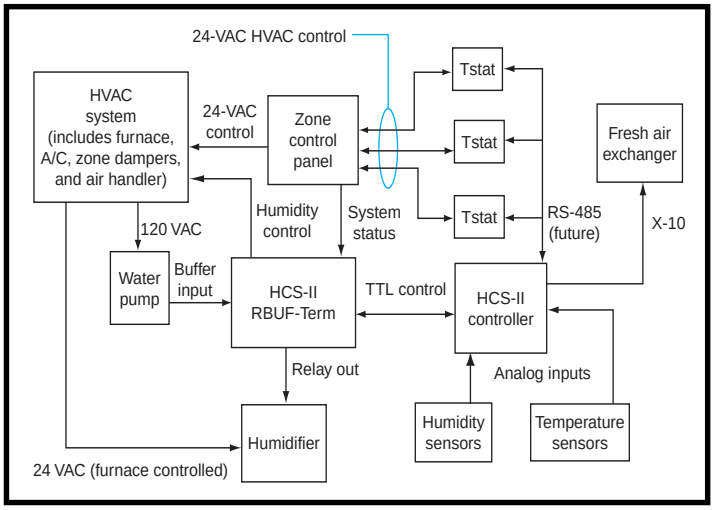


Figure 1—*Controlling this HVAC system involves multiple control points and systems. The variety of sensors and status inputs makes it easy to develop powerful control algorithms. Note that only the humidifier would stop functioning if the entire HCS-II control section quit.*

REMOTE CONTROL

HVAC systems have many systems that require intelligent control. Humidifiers should be used only under certain circumstances. You may not want to run a fresh air exchanger when it is below freezing outside. High-efficiency furnaces with variable speed blowers can be used for added dehumidification during the summer to save energy.

By connecting thermostats, humidity control, and fresh air control to your home automation system, you can develop advanced algorithms to improve the comfort level in your home and also save energy. The most obvious one is temperature setback.

Currently, there is a debate as to whether or not setbacks save energy. Setbacks involve setting thermostats back to less comfortable temperatures while the home is unoccupied or at night to save energy. Numerous factors affect the outcome, including system efficiency, system size (or tonnage) compared to the home size, home insulation, family lifestyle, and the type of lighting used.

Experimentation is the only way to determine if setbacks work for your home. Use the home automation controller to monitor when the system is on and record the duration it runs daily. Monitor the external temperature to take into account the load on the system for that day.

A good way to do this is to monitor degree-days. A degree-day is the difference between 65°F and the mean outdoor air temperature that day. A mean temperature of 80°F during the summer would be a 15° cooling day. A mean temperature of 40°F in the winter would be a 25° heating day. A home automation system with an external temperature sensor makes this an easy task.

You can compare different temperature control philosophies. Leave the temperature set at a standard setting for one month. Record the duration the system is on and the sum of the degree-day for each day. Next, switch to a control model, where you setback the temperature when you are at work or asleep.

Take the recorded data and compare the amount of time the system was on per degree-day. This will provide enough information for the best decision. You also can experiment with other scenarios as well, like using setbacks only during the day or night, but not both.

ADD TO THE MIX

Do you get the feeling temperature control is more involved than you thought? Well, it's only part of the picture. The humidity level of the house affects how it feels. If it's too humid during the summer, it feels hotter. During the winter, moderate humidity improves the feeling of warmth. However, furnaces dry out the air as it is heated.

Each of my systems has two humidistats. One controls an aftermarket humidifier and another feeds an input on the furnace. This input alters the blower speed during cooling to maximize the dehumidification of the inside air, which in turn reduces the load on the air conditioner. Each humidistat is a simple dry contact switch that closes when the humidity level reaches a specific setpoint. The inputs can't be connected in parallel because each has its own AC power source.

Add-on humidifiers should only run when the furnace is heating. In my system, the furnace controls the 24-VAC humidifier power so it is turned on only when the furnace is heating. Older furnaces usually use current sensing relays combined with the heat control line to control the humidifier power.

To replace the six humidistats in my home (two for each system), I used humidity sensors based on an old Jeff Bachiochi article ("From the Bench," *Circuit Cellar* 57). These sensors output 0 to 5 V proportional to 0 to 100% humidity and use sensors from Panametrics. To prevent short cycling, I use XPRESS timers to ensure the humidity relays (closed for higher humidity, open for less) don't change states less than two minutes apart. I also use hysteresis. If I shoot for 35% humidity, I won't open the relay until it reads 37% and won't close the relay until it reads 33%.

STOP THE FLOOD

An important piece of I/O is probably the simplest. Most central air conditioners and high-efficiency gas furnaces produce a lot of condensation. A trickle humidifier produces a

Photo 1—*Zoning an HVAC system requires more duct. Note that the gas exhaust goes through regular PVC pipe because it is cool. All of the furnace controls are located in the lower cabinet.*

lot of wastewater. In crawl spaces, this is drained outside. However, systems in basements use condensation pumps to get the water out.

If this pump fails, the water will flood. During the last cold spell, one of my drain lines froze outside and the water backed up. Most HVAC contractors use pumps with built-in safety switches that open if the water level gets too high. The most common setup is to run the furnace's 24 VAC through the switch so the system will shutdown if the pump backs up. Of course, I had not connected it yet when mine got blocked.

Blocking that occurs at night is a problem because the furnace will shut down, but won't be noticed until the home gets colder. Instead, feed the safety switch to the home automation system. The home automation system can turn the thermostats off and alert occupants.

Leave the pump switch wired into the furnace's 24-VAC supply and monitor the output if you don't want to rely on the home automation system to shut down the furnace. If the 24 VAC stops and the power is still on, either the pump died or the transformer burned out. You can still have an alarm that announces the problem.

AH, FRESH AIR

My favorite feature on my new HVAC system is the fresh air exchanger. This unit mounts on the wall and ties into the existing ductwork. It draws air from outside and injects it into the HVAC return duct. At the same time, it draws air from the house via a centrally located air vent and exhausts it outside (well away from the fresh air intake).

To save energy, the unit has a heat exchanger, which recovers up to 77% of the heat from the air being exhausted. I was skeptical, but during a recent cold spell when outside air was well below freezing, the temperature of the incoming air was warmer than I expected. I'll monitor the temperature more exactly in the future to see how well it functions.

The air exchanger has no control circuitry. It simply plugs into an outlet and runs. However, this was not what I wanted, so I plugged it into an X-10 receptacle. This allows me to control it with some intelligence. Even though it can recover a lot of energy, it is best to run it at night during the summer and during the day in the winter.

I run the exchanger in my HVAC system for 15 min. each day at a certain time depending, on the time of year. The HVAC blower must be on when the air exchanger is used, so I use a parallel relay on the furnace's fan control to turn it on as necessary. When I can communicate with the

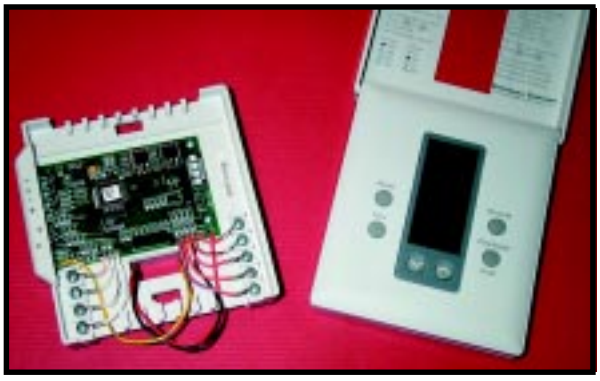


Photo 2—*Statnet thermostats are slim units. The RS-485 interface is added via an extra circuit board (left), which can be mounted behind the thermostat or in a central wiring cabinet. If you mount the thermostat on top of the RS-485 interface, it will be approximately twice as thick.*

thermostats, I'll be able to send a Fan On command instead.

To reduce the amount of wasted heat, the exchanger is not turned on when the outside temperature exceeds preset thresholds. Future plans include monitoring the range hood in the kitchen so I can turn on the exchanger when the hood fan is on.

I plan to add power dampers and duct fans to the air exchanger lines. I want to bypass the heat exchanger in the air exchanger when the outside temperature will help my indoor comfort. During the summer, the night temperature often hovers in the mid-60s. I hope to draw in cool air from outside to reduce my bill.

MORE IDEAS

Now that my HVAC system is under control, there are a few more things I plan to renovate. Most homes have attic fans to help cool the space during the summer. Even if there is excellent ceiling insulation, a 130°F attic will still increase the load on the air conditioner. However, in cooler seasons, heat that's trapped in the attic will reduce the heating load.

Photo 3—*Zone dampers use motors to slowly close the vanes inside the damper. When power is off, a spring opens the vanes. I may add leaf switches to ensure that the dampers close when powered. Right now, detecting problems is difficult because they remain open when off.*

Most fans are installed with a switch so they can be turned on and off. This switch is usually located on the roof joists. If you can reach it, replace the switch with an X-10 relay switch that can handle inductive loads. Now, you can use your home automation controller to decide when to use the fan based on the outside temperature. I don't recommend trying to bypass the built-in thermostat of the fan. Just turn its power source on and off at will.

If you control your attic fan's power, it may be a good idea to locate a temperature sensor in the attic. During a winter warm spell when fan power is off, the attic may reach high, damaging temperatures, so you may want to turn it on. Leave the rest of the heat to help at night when the temperature may drop drastically.

SYSTEM MONITORING

Controlling your HVAC system in a cost-effective manner is important. Devising a smart HVAC control strategy requires a bit of experimentation on your part. You will need to monitor the system, logging when it is on and how long it runs.

Photo 4—*These zone panels handle three thermostats. The thermostats and dampers are wired at the bottom, and the HVAC system control is wired on the left. Numerous indicators show the system's status, including the state of each damper. Monitoring panels with an HA system with buffers is easy.*

Given the standard 24-VAC control architecture, monitoring the system is not that difficult. You can easily monitor the system with a few AC-capable optoisolators fed into a digital input on the home automation system. Industrial I/O modules are ideal for this. If using a zone panel like Trol-A-Temp with numerous status LEDs, you can easily leech off these indicators to monitor the zones.

ONWARD AND UPWARD

A smart combination of standalone control tied to your existing home automation system can make controlling an HVAC system simpler and more robust. In my next article, I'll move beyond the basics and discuss what exactly it takes to get the HCS-II and Statnets talking. I'll also present more XPRESS ideas for smart HVAC control. By the time this project's done, we'll all be comfortable, regardless of the weather.

Mike Baptiste runs his own company, Creative Control Concepts, which designs and sells home automation equipment, including the HCS-II. You may reach him at baptiste@cc-concepts.com.

SOURCES

X-10
X10 Inc.
(201) 784-9700
Fax: (201) 784-9464
www.x10.com

Humidity sensors
Panametrics Inc.
(781) 899-2719
Fax: (781) 899-1552
www.panametrics.com

Statnet thermostats
Enerzone Systems
(972) 424-9808
Fax: (972) 424-8055
www.enerzone.com

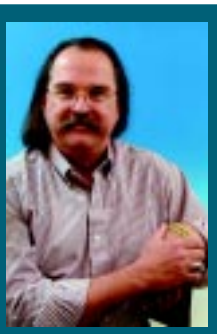
Zone panels and dampers
Trol-A-Temp
(201) 794-8004
Fax: (201) 794-1359
www.trolatemp.com

PerfectAire air exchanger
Research Products Corp.
(608) 257-8801
www.resprod.com

Jeff Bachiochi

Digital and Analog Output Control

Taking Route X-10



Call it quirky, call it temperamental, call it mysterious, but you certainly can't call X-10 useless. Jeff's project sheds some light on controlling outputs via X-10.

Those of you who have X-10 installed in your home are aware of its quirks. There are those lights and appliances that turn on and off randomly. And, there are the modules that work only when the dryer is running or the moon is full. OK, the actual reasons for this mysterious behavior are line noise and signal cross-coupling. But, as soon as you think it's functioning well, an appliance is moved or added, which affects an unrelated module.

However, there certainly isn't anything better than X-10 for the same price. *Circuit Cellar* has presented articles about home automation since its third issue, published more than 12 years ago. Even if a new system was here today, it would take years to integrate it into homes with the success of X-10.

So, my project this month is based on the X-10 control system, developed by X10 Inc. of New Jersey. I won't explain the protocol, you can find that in back issues of *Circuit Cellar*. Instead, I will show you how to take advantage of the X-10 framework to control digital and analog signals.

THE BIG TWO

The two most widely used X-10 modules are the lamp and appliance modules. The appliance module is recognized by its familiar on/off click noise. The module uses mechanical contacts that make or break the circuit path, causing the noise. Because the appliance module has contacts, it can switch large currents. But, it has only two states, on and off.

The lamp module uses a solid state triac to control the power. The triac doesn't have the current capability of the mechanical contacts of the appliance module. Instead, the triac gives control of power at each half-cycle. By controlling what percentage of each cycle the triac conducts, the power to the load is controlled, as well.

Lamp modules are aptly named. Light bulbs dim when power is limited, however, appliances with transformers or motors cease operating. Therefore, don't use lamp modules on

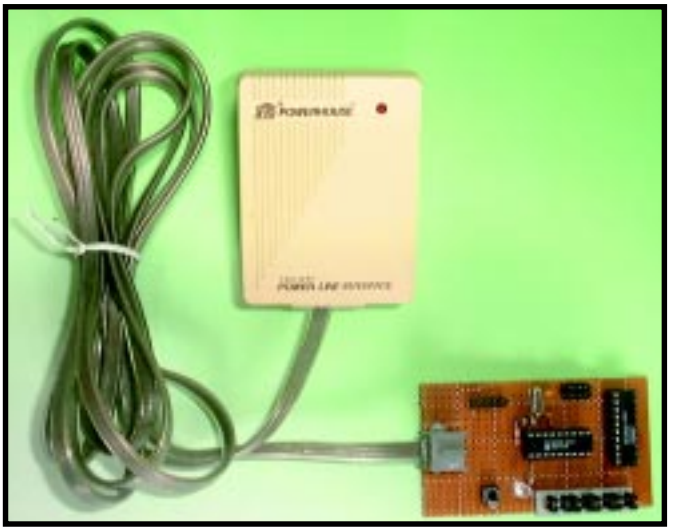


Photo 1—*X-10's isolated power line interface lets you easily talk and listen to or from existing X-10 modules. Here, I used a PIC microprocessor to add X-10–controlled outputs and a quad digital pot for analog outputs.*

objects like ceiling fans and fluorescent lights.

In addition to the on/off commands, the lamp module has two supplementary commands, dim and brighten. There are 31 steps between on and off. The lamp module will brighten from full-off to full-on via 31 brighten commands and reverse the process with 31 dim commands.

For this project, the brighten and dim commands will be the basis for analog output control, while the on and off commands will be the basis of digital output control.

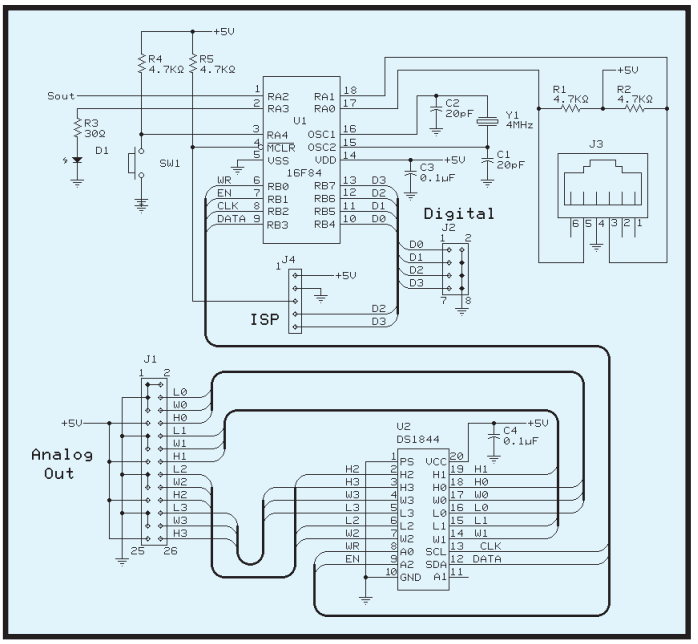


Figure 1—*A pushbutton input allows the user to easily set unit and house code addresses. Both digital and analog outputs are available at four consecutive addresses.*

PL-513 AND TW-523

X10 recognized that a personal or embedded computer would increase X-10 use in the home. The PL-513 was the first commercially available X-10 interface that allowed users to initiate X-10 commands. Fully isolated from the AC line, the PL-513 provides the zero-crossing timing necessary to be able to gate the 100-kHz modulation onto the line.

The next logical improvement was the TW-523. This interface provides a gate signal (referenced to the zero crossing) that shows the presence of X-10 command modulations on the line. Now, you can send X-10 commands and also see X-10 traffic on the power line. Note that only legal X-10 device commands are seen. Extended data commands aren't legal to the TW-523.

LEGALITY

X10 Inc. licenses the X-10 command protocol for free if one of its devices is part of the interface (although, the patent ran out). Using a TW-523 for this project simplifies the interface and satisfies the license agreement. I give kudos to X10 Inc. for providing reasonably priced modules and interfaces. This may explain why no other more advanced, expensive system has been commercialized.

With the zero-crossing and gate output of the TW-523, you can determine the presence of all transmitted X-10 commands. I use a PIC 16F84 to watch those signals. Figure 1 shows the circuitry involved in this project. The five bits of Port A available on the F84 are used for X-10 interface, push button input, LED output, and an optional serial output. The remaining eight bits of Port B are reserved for outputs. The lower nibble supports analog output control. The upper nibble is for digital outputs.

Although the analog outputs could have been an external DAC, I used a quad digital pot for a couple reasons—DACs are expensive, and digital pots offer an isolation factor and can be used as direct replacements for mechanical pots. A digital pot's physical output connections are upper, lower, and wiper terminals of the pot. These devices are offered in a few different ranges, so you can swap values by plugging in a different chip.

Most commercial X-10 devices use a mechanical selection for setting the module's address, that is, house and key codes. This project defaults to A1 when plugged in the first time. At that point, the first digital output and analog output default to the base address. The remaining outputs are based on this address. In this case, they become A2, A3, and A4. Notice that I'm allowing a digital output and analog output to occupy the same address. This is arbitrary, and I could have given each its own address.

To choose an alternate base address, the pushbutton must be pressed upon powerup. When this is done, the LED turns on to show that you're in the learning mode. The next X-10 transmission it sees that contains a house and unit code (key code <16) will be used to designate a new base address.

A BETTER PicBasic?

I've used PicBasic for a long time. When Micro Engineering Labs came out with its Pro version, I paid no attention to the fact that X-10 support was added. Clearly, Micro Engineering Labs made PicBasic Pro easy to work with. Although I could have written this in assembler, there are advantages to using the right tool.

If you know the F84, you know it has only one timer and no UART or SPI hardware. But, it does have the advantage of 64 bytes of internal EEPROM for nonvolatile storage.

Figure 2 shows a flow chart of the program. I like building programs on smaller successes (as opposed to debugging the total program). So, I often add an optional serial output port, which is used as a debugging tool, even when serial output isn't needed.

The first application I wrote was for monitoring and reporting X-10 traffic. When connected to a PC running a terminal program, I saw how well the compiler's X-10 routines worked. Three lines of BASIC code were all it took to convince me.

I needed to save the house and unit codes into EEPROM so the device would remember the base address, even when the power is off. If the values read from these locations are not legal values (0–15), the house code defaults to A and the unit code to 1. But, how do I know what the module thinks its base address is when there's no display? I used the LED output to flash the house code first, pause, and

then flash the unit code. During powerup, after checking the pushbutton for a programming request, the module's code is flashed for address confirmation.

The BASIC `Xin` command fetches the next X-10 transmission. A word (two bytes) is returned with the transmission's house code in the upper byte and key code in the lower byte. The house code is always part of the transmission. However, the key code can hold either the unit code (key code <16) or function code (key code >15).

When a key code value is less than 16, that value (a unit code) is saved into the present unit variable. This variable is necessary for remembering what unit code future commands (function codes) are acting on.

When an X-10 transmission contains a function code (key code >15), it looks for a match between the house code received and MYHOUSE in EEPROM (the module's base address). If there's no match, the program flows back to the `Xin` command. If there is a match, then another match is looked for among the present unit value (last unit code received) and the "MYUNIT0" in the EEPROM (the module's base address), and the next three sequential unit codes (unit codes are mod four).

A match with any of the four sequential unit codes sets the variable UNIT to the offset of the match's base address (0–3). UNIT indicates which output you would operate on. Again, non-matching codes return program flow to the `Xin` command.

Now, you're ready to look at the function code and decide what to do about it. Although there are 16 functions listed in the X-10 house and key code table, only half of them relate to module control. Along with the normal on, off, bright, and dim functions, more were added to perform special functions. One function, all units off, is like a safety command, allowing everything to be turned off at once. All lights on and all lights off can attract attention in an alarm situation. As a programmer, you don't have to follow the rules regarding functions.

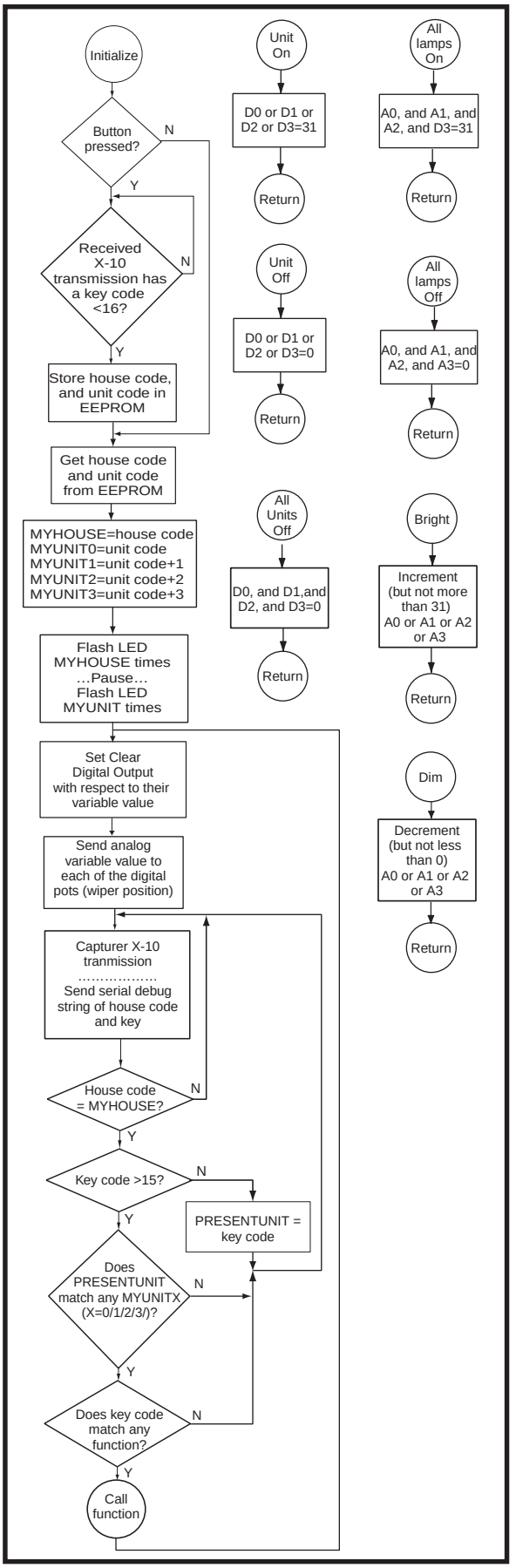


Figure 2—*The function calls (on the right) can be easily altered to give any response required to the other X-10 command.*

ACTION ITEMS

I covered seven of the function commands. When the function code matches one of these constants, the program branches to separate routines, which affects one or more of the eight outputs variables. Variables A0–A3 (analog) and D0–D3 (digital) hold values from 0–31 (all but 0 will be considered on). Brighten and dim commands increment and decrement the A*x* values, in their respective routines. Other routines clear or set (to 31) the A*x* and D*x* values, in their respective routines.

Before returning program flow to the `Xin` command and awaiting another X-10 transmission, there is one more action item. Depending on the values of the four digital variables D0–D3, the corresponding outputs of Port B (B.4–B.7) are set or cleared. Any value other than zero sets the corresponding output bit. It's a different story for the analog outputs.

ANALOG OUTPUTS

My first thought for analog outputs was an external quad serial DAC, but that's overkill here. I considered an R2R ladder, but there aren't enough pins for four analog outputs.

Then, I thought of PWM outputs with RC filters on each of the outputs. This could be cheap, however, it couldn't be accomplished using PicBasic. Because the PWMs would be background task and the F84 has one timer, there would be a problem sharing the timer with the `Xin` and `Serout` commands of PicBasic. I could have written this successfully in assembler, but that wasn't my goal. The

next solution I contemplated had all the right pros and no cons.

Interfacing to a quad digital pot solved the DAC problem while still allowing the program to be written in PicBasic. The added advantage of having the device look and act like a mechanical pot was overwhelming. The Dallas part I used has 64 positions. I needed 32 positions to replicate the 32 states of a lamp module. (I could have used all 64 positions, forcing the user to send 64 dim or brighten commands to traverse full scale.)

Controlling this digital pot can be through either SPI or I^2C. (PicBasic has commands for both SPI and I^2C.) I didn't need to control many devices tied to the same two-wire bus, so I used the SPI interface. A single-input pin on the device selects the interface mode. All four pots can be written to in a single four-byte transfer. Each byte contains the same destination pot number (0–3) as the MS bits of the transfer byte. The LS six-bits are the wiper position (0–63). Because my variables hold values of 0–31, I multiply these by two (shift left) before ORing in the pot-select MS bits.

WHAT'S NEXT

The digital outputs can interface directly with any digital input, such as a PC's printer port status inputs or a solid state relay. Although there are some sensitive 5-V DC relays, most require 50 mA or more, so a transistor driver is necessary for direct control of a mechanical relay. The same applies for a solenoid, which is appropriate for use as a door-strike release. On/off control of other devices might require an opto device. The isolated side of opto devices includes open-collector, FET, TTL, SCR, and triac.

As seen in Figure 1 and Photo 1, the analog outputs (pot terminals) are placed on 2 × 3 connectors in a way that offers connection to V_{CC} and ground. With jumpers to the end terminals of the pot, this gives ~159 mV per step based on a V_{CC} of 5 V. Here, you have an analog voltage that can be used as a voltage source with those appliances, which offers control of external inputs such as illumination, position, or speed.

A larger field of contenders may exist for appliances that already have mechanical potentiometers, which can be replaced by the digital pot. Because each of the four pots is isolated from the power supply, they can be substituted for any mechanical pot. Applications here include volume or tone controls of any audio device (although the DS1844 is only available in linear taper, other DS18*xx* products are available with a log taper). You can be creative when designing them into your own circuitry for offset, gain, tuning, or calibration adjustments.

This is a project everyone can enjoy because most of the pieces are available off-the-shelf. And, if X-10 seems mysterious to you, this project just might turn on some lights for you.

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

SOURCES

X-10
X10 Inc.
(800) 675-3044
(201) 784-9700
Fax: (201) 784-9464
www.x10.com

TW-523 Power Line Interface
Creative Control Concepts
(919) 304-3107
www.cc-concepts.com

DS1844-010/-050/-100
Dallas Semiconductor
(972) 371-4448
Fax: (972) 371-3715
www.dalsemi.com

PIC16F84
Microchip Technology Inc.
(888) 628-6247
(480) 786-7200
Fax: (480) 899-9210
www.microchip.com

PicBasic Pro
Micro Engineering Labs, Inc.
(719) 520-5323
Fax: (719) 520-1867
www.melabs.com

On The Road Again

SILICON UPDATE

Tom Cantrell

Part 1: Going Places with Silicon

This month, Tom combines two interests and takes an under-the-hood look at the chips that reside under the hood. Peel back the chrome and leather and you'll see how "smart" today's cars really are.

I consider myself lucky to get to write about chips, the silicon wonders that have changed our lives in many ways.

Sometimes I contemplate what I would have done without the computers and chips. Sell insurance? I don't think so.

Instead, I could see myself writing for a car magazine. Chips and cars both expand our reach, and I am fascinated by both. They take us places we couldn't go otherwise, whether with moving electrons or moving parts.

Today, those of us who like computers, chips, and cars have it made because they are all coming together. At this point, cars arguably represent the single largest application for chips and incorporate as much, if not more, computing power than computers!

If you share my feeling that getting under the hood of chips like we do here at *Circuit Cellar* is exciting, what could be better than getting under the hood of the chips that are under the hood?

Buckle up, and let's go for a ride.

SILICON CARRIAGE

Mechanically (at a macro block-diagram level), today's cars are little changed from those of yore. Your car probably has an internal combustion engine, four tires, and a steering wheel, just like your parents' car had.

It's the chrome and tailfins—the accessories and doodads like automatic transmission and air conditioning—that differentiate today's cars from the Model-T.

Until 20 years ago, progress was more or less limited by the pace of mechanical advancement. When the IC entered the picture, car evolution started to speed up, reflecting the increasing electronic content. The pace of innovation is becoming more and more measured in Silicon Valley chip time rather than Detroit pounding steel time.

As a chip kind of guy, it's interesting for me to watch the auto suppliers migrate down the path of computerization. What you see is the car folks following the trail markers laid down by the computer types. For instance, in the late '70s and early '80s, you'd typically find a single 8- or 16-bit

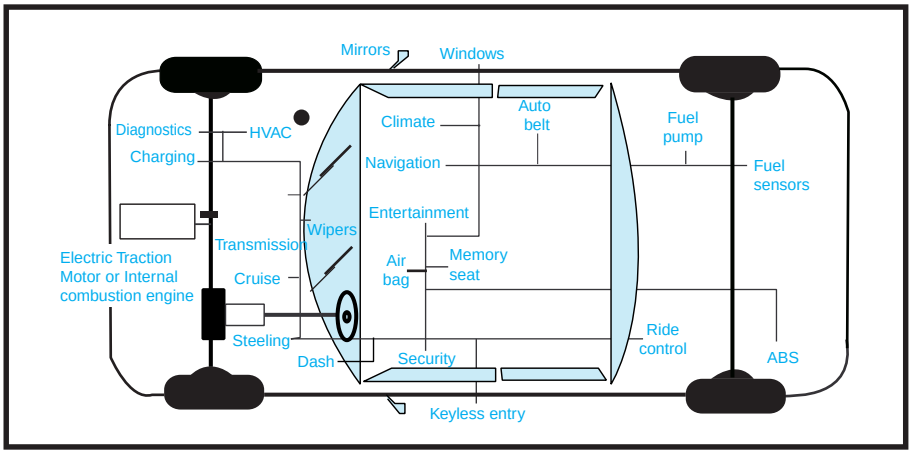


Figure 1—*WWW = Wheels Wide Web? Automotive LANs, such as J1850 and CAN, are required to support the MCUs onboard. Communicating sub-systems cut wire cost and enable advanced total vehicle control strategies.*

MCU working as the centralized mainframe of the car, handling the few most critical functions associated with engine control and such.

Today, it's not surprising to notice subsequent movement to more bits (32-bit MCUs), more memory (e.g., half-megabyte of flash memory is typical for high-end controllers), and more megahertz.

Naturally, the car designers, like the software people, never met a MIPS or MHz they didn't like and were more than happy to hang more car features and accessories on the chip. Just as with the mainframes of old, eventually the single central brain starts wheezing under the load of trying to juggle everything at once.

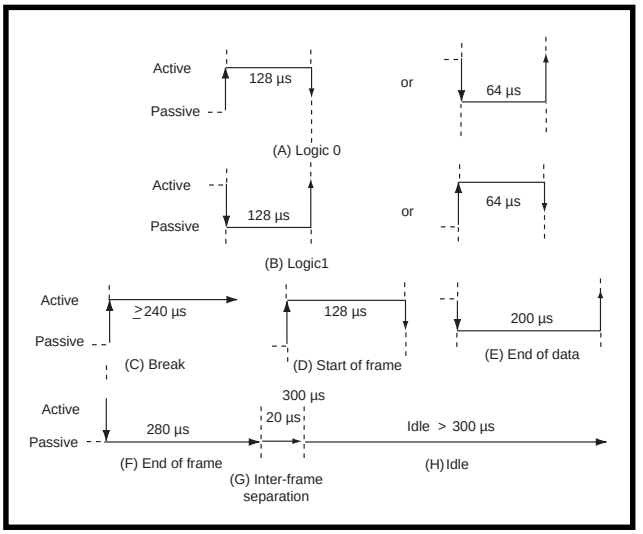


Figure 2—*At this point, J1850 VPW (Variable Pulse Width) encoding, notably used by GM, probably has the largest installed base. Unlike J1850 PWM or pre-J1850 UARTs, it uses each edge to deliver information.*

Following the roadmap, the next move was to begin to supplement the single engine controller by adding various sub-system processors. Now, instead of just an ECU, there are controllers for brakes, transmission, lighting, and so on. I've heard that today, a luxury auto may have as many as 50 micros doing their thing. More micros deliver the processing power to handle more functions, but that isn't the end of the story.

Originally, the function of each controller was carried out in isolation (i.e., the ECU handled the engine, BCU the brakes, TCU the transmission, and so on).

But, car designers wanted more, notably the ability for the specialized controllers to communicate with each other and gain mutual access to each others' data and the myriad of sensors from which it comes. For instance, both the engine and transmission controllers think they can do a better job if they know the actual wheel speed, data that was traditionally only needed for the brake controller.

In the old days, the computers accomplished such sharing with point-to-point networks. Of course, there is a problem in getting everything to talk point-to-point, and that's the fact that you need a lot of wire.

The wire problem shouldn't be underestimated. Anyone who's poked around the innards of a car knows that today's vehicles bring new meaning to the term rat's nest. According to an app note from ST Micro, a typical wiring harness includes one mile of wire, weighs more than 75 lbs, and costs $1000! [1]

Okay computer people, here's a quiz. What do you do if you want to connect a lot of computers and let them share data, but you don't want to string miles of wire?

LIFE IN THE FAST LAN

The automotive LAN concept, called multiplexing in car-speak, is the answer, and the past five years has seen widespread adoption (see Figure 1). The concept is the same as in the computer world, with engines and transmissions taking the place of workstations and file servers.

You've probably heard of CAN (Controller Area Network), which is widely accepted by automakers as a next generation standard. CAN is also used in non-automotive industrial applications, where it is the basis for a variety of Fieldbus schemes. There have been a number of articles covering CAN in *Circuit Cellar* (58, 59, 69, 70, 71, 114), and the major MCU suppliers are quickly jumping on the bandwagon and introducing CAN-enabled controllers.

Although CAN is already on the road in Europe, if you're one of the hundred million or so people driving a recent vintage domestic (from the last 10 years), what's actually under your hood isn't CAN but likely some variant of the current US automotive LAN standard, SAE J1850.

Keep in mind that J1850 only defines what I would consider the physical and link layer specs, such as media type and electrical characteristics, data rates, arbitration, priority, and so on.

In other words, J1850 only deals with getting a piece of data from here to there, rather than what the data means. There are other standards associated with what would be considered the application layer (important ones include J1979 and J2178) that define higher-level message format and interpretation.

Before you run over to www.sae.org and start filling your shopping cart, here's a tip. Instead of trying to figure out which of the dozens of multiplexing-related standards you need, ask for a document known as HS-3000. This 399-page tome contains the latest version of J1850 (make sure you get the 1999 edition) plus all the other related standards, and at just $89, is much less expensive than buying the individual standards separately.

Now, back to the wire. J1850 reminds me of a cross between Ethernet and I²C. It's a multimaster bus topology in which nodes (up to 32) contend for access to a single wire (up to 40 meters). Like Ethernet, nodes are free to attempt transmission any time the bus is idle, and they detect collisions by monitoring whether or not what's on the wire matches what they are trying to send.

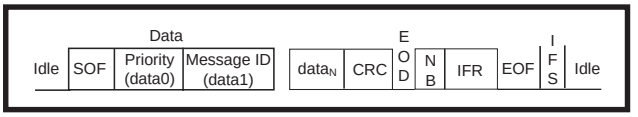


Figure 3—*The definition of a J1850 frame is intentionally broad to cover a range of applications. The length varies from 2–12 bytes, including CRC and optional In-Frame Response (IFR).*

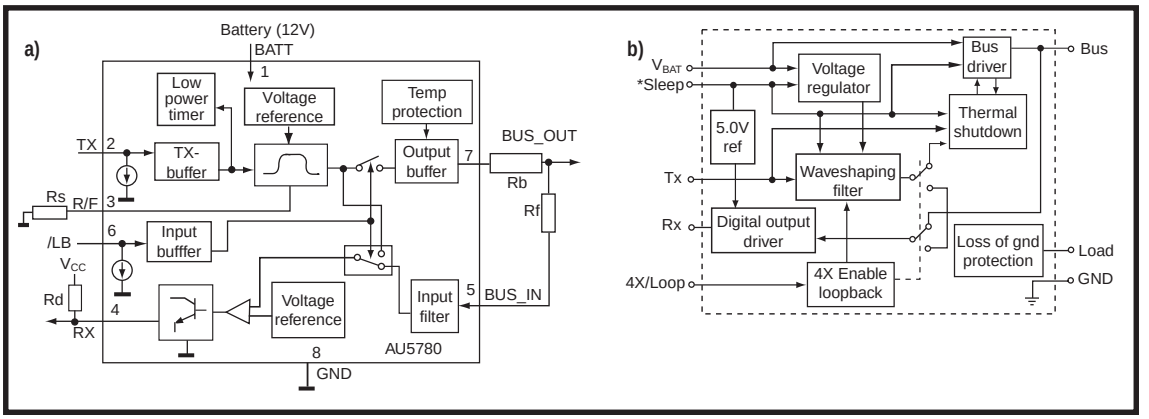


Figure 4—*An under-the-hood MCU faces all manner of electrical hazards and potholes. A transceiver, such as the Philips AU5780A **(a)** or Motorola MC33390 **(b)** acts like a good set of shock absorbers, making for a smooth and safe ride.*

When an Ethernet transmitter detects a collision with another transmitter, both transmitters back off (i.e., quit transmitting immediately), and then each will retry. Repeated collisions are avoided because each node waits for a different (random) length of time.

Instead, J1850 uses an I²C-like arbitration mechanism in which, at the instant of collision, the transmitter that happens to be transmitting a dominant level (0) continues while the other (transmitting a 1) stops. This is more efficient than Ethernet, because at least one message gets through rather than both having to be completely resent. It provides the foundation for a priority scheme in which more important messages are coded in such a manner as to be dominant over others, as well.

Just as there are various versions of Ethernet (speed, wire type, etc.), so is the case with J1850. Basically, reflecting the proprietary history and tendencies of automakers, J1850 supports what boils down to GM- and Ford-type variants.

The differentiation encompasses speed, encoding, and wire type. GM runs at 10.4 Kbps, Ford at 41.6 Kbps. Notice that 41.6 Kbps is exactly four times 10.4 Kbps, and in fact, some 10.4-Kbps applications use 41.6 Kbps for running electrical diagnostics, the so-called 4X mode.

Ford uses a PWM encoding in which the bit cells are equal, the difference between a 1 and 0 is a 1/3 or 2/3 duty-cycle respectively.

The GM uses Variable Pulse Width (see Figure 2), which is trickier. Bit time is variable, and determination of a 1 or 0 depends on the previous state and time between transitions rather than a level or duty-cycle.

Finally, GM relies on a single wire while Ford uses a differential pair. The DC characteristics are roughly similar, but only roughly, reflecting the one versus two-wire tradeoff in an electrically harsh and noisy environment. Essentially, 1-wire VPW calls for a larger voltage swing (0–8 V vs. 0–5.25 V for PWM), must tolerate more ground offset (2 V vs. 1 V), and limits edge rate (18 µs vs. 1.75 µs minimum) to reduce radiated emissions.

Note that these specs define circumstances under which communications are guaranteed to continue undisturbed, not absolute limits. In fact, J1850 requires that nodes survive such ugly (and invariably likely) failures as a bus short to battery, which is typically 12–16, but can spike up to 50 V during what's ominously referred to as a "load dump."

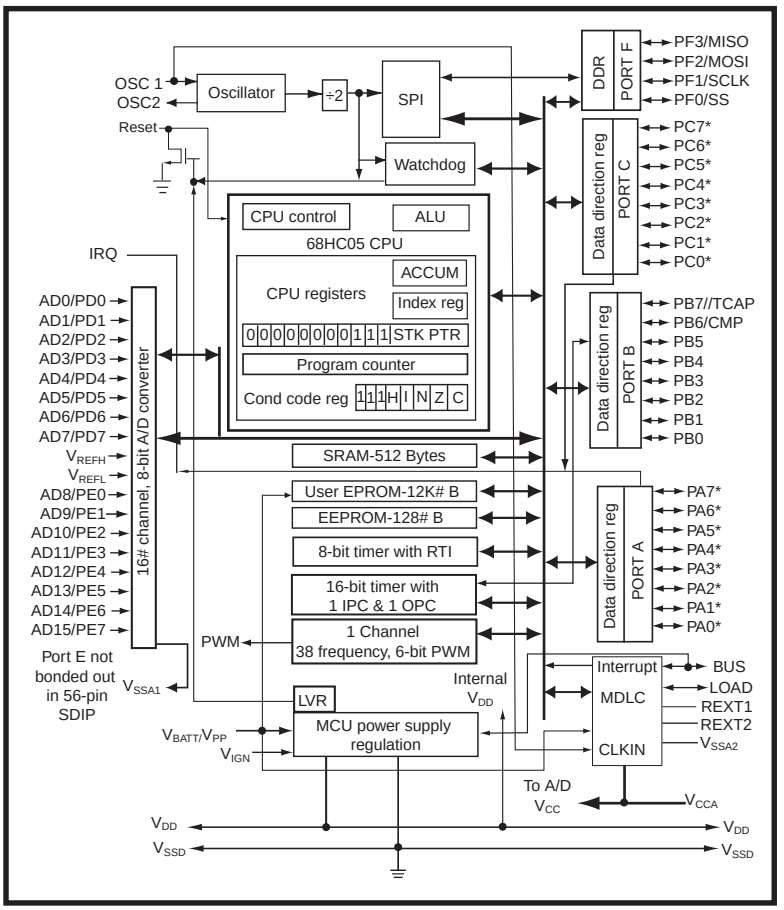


Figure 5—*The 'HC705V8 uses Motorola's MDLC, which handles J1850 traffic at the message level.*

MESSAGE IN A THROTTLE

Although Ford and GM 1s and 0s do not look exactly the same on the wire, moving up a notch on the J1850 network hierarchy, things start to come together in the form of a standard packet or frame format,

as shown in Figure 3. Additional symbols (i.e., pulses of different lengths) are defined to delineate various portions of the frame.

From an idle bus, a frame begins with an SOF (start of frame) symbol, followed by a header, data field, CRC, and then an EOD (end of data) symbol. Then, it's possible to receive an immediate in-frame response (IFR) preceded by a normalization bit (NB), and either a single byte from a single responder, multiple bytes from a single responder, or single bytes from multiple responders. The frame terminates with an EOF (end of frame) symbol. The number of header, data, and optional response bytes varies within the constraint that the maximum frame length between SOF and EOF is limited (12 bytes for VPW, 101 bit times for PWM).

I'll get deeper (much deeper) into the details of frame format later. For now, my point is that J1850 is designed to efficiently support a range of communication requirements encompassing messages that are short (1 byte data plus CRC) or long (11 bytes plus CRC), frequent or rare, broadcast, acknowledged, and so on.

For instance, both functional and physical addressing are supported. As with a typical LAN, physical addressing works with the source and destination addresses of specific nodes.

By contrast, functional addressing is a more unique concept, kind of like publish and subscribe. For example, the braking/traction controller can simply publish a single wheel speed message that can be subscribed to by a variety of interested parties such as the instrument panel, cruise controller, and transmission.

There's even extended or geographical addressing that goes so far as to map the car into 49 distinct zones (e.g., left side aft A pillar). For instance, imagine a multizone closed-loop automatic climate-control setup. If the auto switch is turned off for the passenger zone, there's no need to ask for, or receive, superfluous input from the passenger side cabin temperature sensor. Although the example is somewhat contrived, it's not difficult to imagine geographic addressing becoming increasingly useful as the number of sensors, motors, and switches continues to grow.

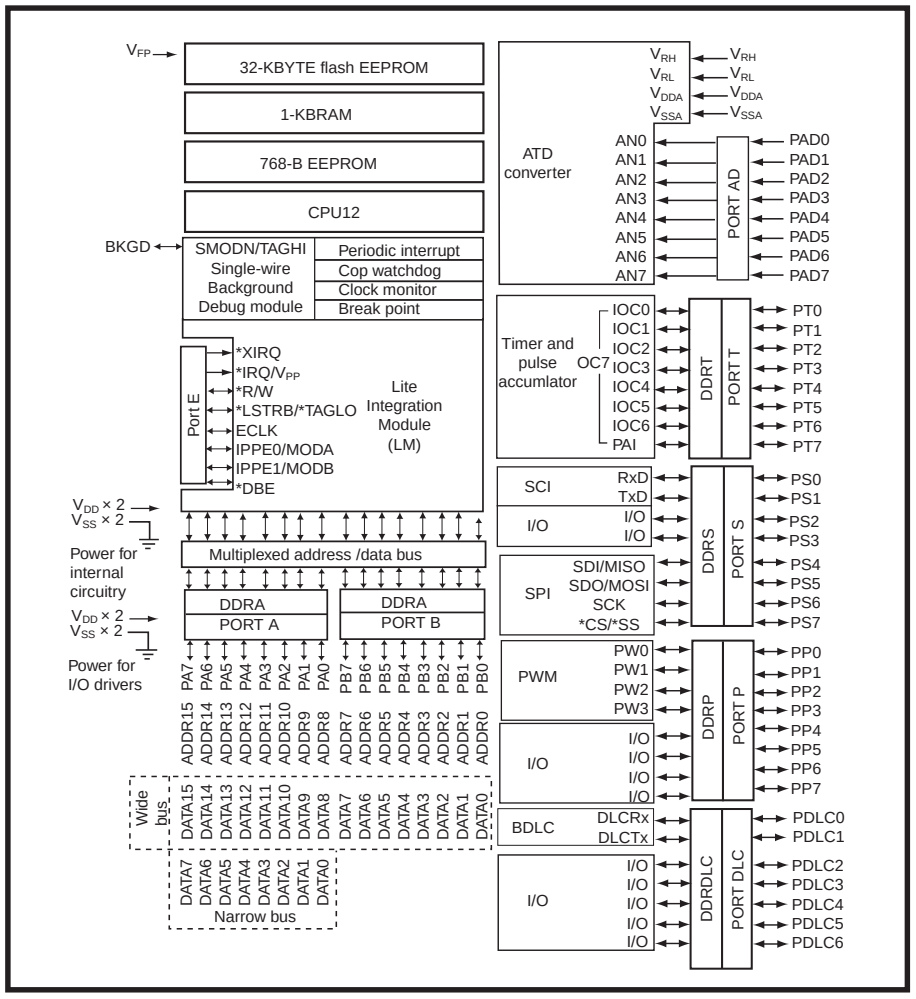


Figure 6—*The HC912B32 is available in both flash memory and ROM versions. More MIPs and a BDLC (Byte Data Link Communications) module deliver sophisticated processing and communication capability.*

SOCKET SET

Because the standard has been bouncing around since the early 1990s, there's no shortage of chips that speak J1850.

Although not trivial, the relatively low speeds mean crafting your own J1850 interface software or ASIC is certainly feasible. The most critical path is probably arbitration, which requires the loser, having detected a collision, to quit transmitting within the bit time. Calculating the CRC, address matching, and handling in-frame or multiple responses are all gotchas to be dealt with. No problem. Thanks to the march of silicon, there are plenty of MIPS and gates on the shelf to throw at the problem.

I highly recommend that you think twice before hanging jumper cables on your favorite MCU or ASIC though. Remember that cars are not an electrically friendly place, so to protect your fragile digital chip, consider a J1850 transceiver, such as the AU5780A from Philips shown in Figure 4a. Not only does it fend off load dumps (50-V spike for 1 s), but it also tolerates such sad but true glitches as a jumpstart gone awry (i.e., battery reversal), thermal overload, 9-kV ESD, and so on. It handles the waveshaping of the output (slew rate limit), something you would probably rather not hack at with your own software or gates. Notice in Figure 4b that Motorola makes a similar J1850 transceiver, the MC33390.

Frankly, rolling your own J1850 protocol MCU or ASIC is ill-advised for all but the most serious and specialized players. Most people will be better served doing less making and more buying.

I found a good list of parts at the web site of an outfit called Advanced Vehicle Technologies (www.avt-hq.com/devices.htm). While you're there, be sure to grab a copy of an excellent paper by Michael Riley, "In-Vehicle Computer Networks."

As you peruse the list, you'll quickly see that Motorola is by far the heavyweight champ when it comes to J1850-savvy chips. They've got an entire range of 'HC05s, '08s, '12s, '16s, 68ks, and PowerPCs targeting the automotive market.

At the low-end, the 'HC05V8 is a good example (see Figure 5). It combines the otherwise conventional MCU with a J1850 MDLC (Message Data Link Controller). The MDLC handles messages at a rather high level, offloading the MCU (after all, the '05 isn't exactly the Corvette of controllers) of the gory details (encoding, arbitration, CRC, etc.). On-chip buffers (one 11-byte transmit and two 11-byte receive) largely insulate the '05 from high-speed timing hassles. Built-in error checking means that the '05 doesn't need to bother cleaning up after accidents. The MCU never even sees a bad message.

However, the easy-to-use message-level approach comes at a price in terms of flexibility and the ability to deal with some J1850 features. For instance, while the MDLC can coexist with the earlier mentioned diagnostic 4X VPW mode, it can neither send nor receive such messages. Furthermore, the MDLC doesn't support the optional in-frame response feature. That's okay for GM, who doesn't use IFR, but not Chrysler, who does.

Enter a chip like the 'HC912B32 (see Figure 6), which combines a much higher performance 'HC12 16-bit core with a different J1850 module, the BDLC (Byte Data Link Communication). As the name implies, the BDLC handles communication at a lower byte, rather than on the message level. The BDLC can receive (but not transmit) 4X messages, accommodates in-frame response, and is generally more flexible.

For instance, when the MDLC loses arbitration during transmission or encounters a CRC error during reception, it simply handles the situation automatically without ever informing the MCU. However, it may be that a designer, for whatever reason, would like to know what's going on and deal with it explicitly. Thus, the BDLC issues an interrupt upon arbitration loss or CRC error, leaving it to the MCU to take the proper course of action. Sure, it takes cycles, but that's what MIPS and MHz are for.

HONEY, I CRASHED THE CAR

As part of my job, I get to hack around with all manner of chips and computing gadgets. However, I also have to write and deliver a manuscript at the end of every month—no ifs, ands, or buts.

I learned my lesson the hard way long ago. I don't use the same PC I write my columns on for my various experiments. I don't install every weird software package that crosses my desk or, heaven forbid, pop the lid and plug in any hardware of dubious or unknown pedigree.

May I humbly suggest you think twice (make that about 99 times) before toddling out into the garage, wire cutters and soldering iron in hand, and start playing brain surgeon with your daily driver. That is, unless you've got a hankering to experience your local mass transit options first hand while your mechanic has a good laugh ("You did what?").

Instead, why don't you spend some time surfing on the Internet and reading up on the subject. By the time you're up to speed, next month's issue should be arriving, and I'll show you a safer and saner way to get online under the hood.

Tom Cantrell has been working on chip, board, and systems design and marketing in Silicon Valley for many years. You may reach him by e-mail at tom.cantrell@circuitcellar.com or by telephone at (510) 657-0264.

REFERENCE

[1] L. Perier & A. Coen, *CAN-do Solutions for Car Multiplexing*, ST7/ST10/U435 ST Microelectronics Application Note AN1086

SOURCES

AV5780A
Philips Semiconductors
(408) 991-5207
Fax: (408) 991-3773
www.semiconductors.philips.com

MC33390
Motorola
(512) 328-2268
Fax: (512) 891-4465
www.mot-sps.com

CIRCUIT CELLAR Test Your EQ

Problem 1—How can you measure an unknown voltage without drawing any current whatsoever?

Problem 2—What are the timing formulas (monostable and astable) for the venerable 555 timer?

Problem 3—In the U.S., how many AM broadcast channels are there? How many FM broadcast channels?

Problem 4—When sorting items on a computer, what is the time order of complexity (relative to the number of items to be sorted N) if you are limited to comparing two items at a time? What is it if this restriction is removed?

What's *your* EQ?—The answers and 4 additional questions and answers are posted at www.circuitcellar.com.

You may contact the quizmasters at eq@circuitcellar.com.

8 more EQ questions each month in Circuit Cellar Online

see pg. 2

PRIORITY INTERRUPT

Publishing 101

It's not exactly Publishing 101, but the economic logic behind printing a magazine is pretty easy to grasp. The expenses are paper, printing, production, editorial staff, and reader fulfillment (getting and keeping readers). Revenue comes from subscriptions and advertising; if B is greater than A, you stay in business.

Unless you're talking about one of those oil-prospecting newsletters ($2k a year), subscription fees don't come close to paying for it. Typically, if the subscription income is enough to pay for the paper, postage, and printing of the magazine, you're doing well. And, unless you have other income to offset it, increasing paper and postage costs are usually the reason you'd raise subscription rates (*Circuit Cellar's* subscription price has been the same for seven years, BTW). Advertising pays for the rest of the magazine, including the staff.

If a magazine is to grow and prosper in today's economy, it either has to deal with the realities of the print publishing model and do it better than anyone else, or change its business model to capitalize on the part that it does best.

So, how is the competition fairing? Disaster is probably a sympathetic description of what is happening elsewhere. With the possible exception of EDN, trade magazine advertising is down significantly. Some magazines, like *Computer Design*, have vanished while others have significantly reduced their size. I predict that we haven't seen the end of this trend, and a few more trades will bite the dust in the next year.

At the other end of the spectrum, there has been that great sucking sound of consolidation. Two technical magazines from the same publisher suddenly became one journal in January, while another publisher with a quadruple spread of quarterly presentations combined them into a single quarterly periodical. You don't do that if you are making money.

It's easy to blame the Internet for the demise of print advertising revenue. Ten years ago, where else would an engineer go besides the trade press and technical magazines to read about the latest and greatest from Dallas or Analog Devices. Today, however, big advertisers put more money into their own web activities and less into print advertising. What is a print publisher supposed to do?

In my opinion, the decline of borderline print publications is a direct result of the one thing I learned at *Circuit Cellar*, content is king! Make the editorial content of any publication (print or web) worthwhile and there will be loyal readers. With an established readership, advertising revenues will prosper. In the past year, our circulation has increased, our website activity has doubled, and we've had the highest advertising year in the history of *Circuit Cellar*.

Are we better at publishing a magazine than Cahners or Penton? Absolutely not. But, like any small business, we have the advantage of quick action and rapid change. The thing that *Circuit Cellar* does best is its editorial. If there is a practicable business model to grow our print publication, it is simply to take what we do best and find more places to publish it. Content is still king.

In the past year, *Circuit Cellar* editorial expanded considerably. Most notably, readers can find more articles and projects each month in *Circuit Cellar Online*. Hosted by ChipCenter (www.chipcenter.com), *Circuit Cellar Online* is not a rehash of the print magazine, but entirely new and unique editorial. Tom Cantrell, Ingo Cyliax, and George Martin provide lots of technical wit and wisdom each month, along with a bunch of feature articles and projects on the subjects that you enjoy here in the print magazine. Also, there are eight more EQ questions for you to solve. Read everything online or print a PDF to read later.

Having *Circuit Cellar Online* also allows us to have some unique web-resident features. Every issue includes tidbits we call Resource Links. Covering such varied subjects as flash memory, TFTP, low-voltage differential signaling, and power measurement, these Resource Links provide an in-depth review of the different web-resident resources available on the subject.

Recently *Circuit Cellar Online* expanded even further with a new service called ASK US. ASK US is a free technical question-and-answer forum that's run by *Circuit Cellar,* and it isn't some BBS where everyone asks and no one answers. There is a team of engineers and programmers who get paid to help you. The detailed answers in the archive are definitely worth a visit.

So, I think I've figured out the prescription for making at least one print magazine grow, and it isn't to add more paper. It's to add more editorial (grin). In truth, the realities of doing business don't change. A larger magazine with many extra articles needs a lot more advertising to pay for them. I won't kid you, right now that advertiser is ChipCenter. The best way to keep *Circuit Cellar* the best magazine that hits your mailbox each month is to read and support *Circuit Cellar Online,* as well. A business school graduate might criticize the interdependence of my publishing activities, but he's only looking at how many dollars fall out as profit. I'm looking at online editorial expansion as an excellent way for us to keep doing what we do best.

steve.ciarcia@circuitcellar.com