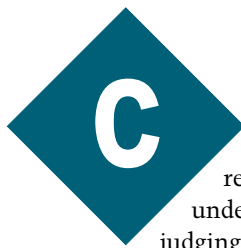


SILICON UPDATE

Tom Cantrell

RambLIN' Man

The name of the game is networking and automotive electronics is the target. In this article, Tom predicts a time when cars will be Internet-enabled, but the standards for such road warriors are in the limelight as well. Cars and chips together open up a wide array of possibilities, but when car manufacturers roll their own standards, you could find yourself tripped up on new wires.



Circuit Cellar is revered for going under the hood, and judging by the response, you take it literally. My road-warrior articles such as "Saab Story" (*Circuit Cellar* 57) and "On the Road Again" (*Circuit Cellar* 118) invariably generated a lot of comments. Clearly, automotive electronics is a popular topic. Besides the fact that it's arguably the heavyweight of embedded applications, I suspect that many engineers are into cars as well as chips and can fully imagine the possibilities offered by bringing them together.

In a nutshell, what's going on under the hood in cars reflects the overall trend—network anything and everything that's got a moving electron.

At the top level, I think it's safe to predict that cars ultimately will be Internet-enabled. But, TCP/IP and HTTP will go only so far. Under-the-hood networking remains the province of the car manufacturers who historically haven't been afraid to roll their own standards and such.

Nevertheless, recent years have seen the automakers muddling through

the process of rationalizing their networking schemes. With more than a bit of prodding from regulatory and pollution control authorities, networking standards have taken hold.

Not that the old proprietary parts counter and authorized service mentality has gone down without a fight. For instance, the historic U.S. car networking standard (SAE J1850) comes in different flavors, with each camp adding their own proprietary and incompatible tweaks.

Ford's version of J1850 runs at 41.6 kbps over a two-wire differential pair using conventional PWM modulation. By contrast, GM's version runs at 10.4 kbps on a single wire using a variable pulse width scheme. What a hassle for chips, parts, and tool suppliers!

Even as I write, the J1850 era is coming to a close. Yes, it was a kludge. Not surprising given its mishmash of proprietary roots. But, I wouldn't go so far as some people and call it a failure. The fact is, J1850 irrevocably set a course towards ubiquitous VolksComputing and established the proving ground of daily drivers needed to make it happen.

The Europeans may have trouble agreeing on the Euro, but they've done a great job prodding things along towards CAN, the emerging one-world automotive networking standard (see Figure 1).

But, networking (or multiplexing, as aficionados say) in cars is a hierarchical situation. Schemes like J1850 and CAN were conceived for serving high-level (so-called Class B or C) communication among major subsystems such as the engine, transmission, and ABS control units.

Unfortunately, based on my own experience under the hood, a major goal of networking (reducing the packaging, weight, reliability, and cost burden associated with the traditional wiring harness) remains elusive. J1850

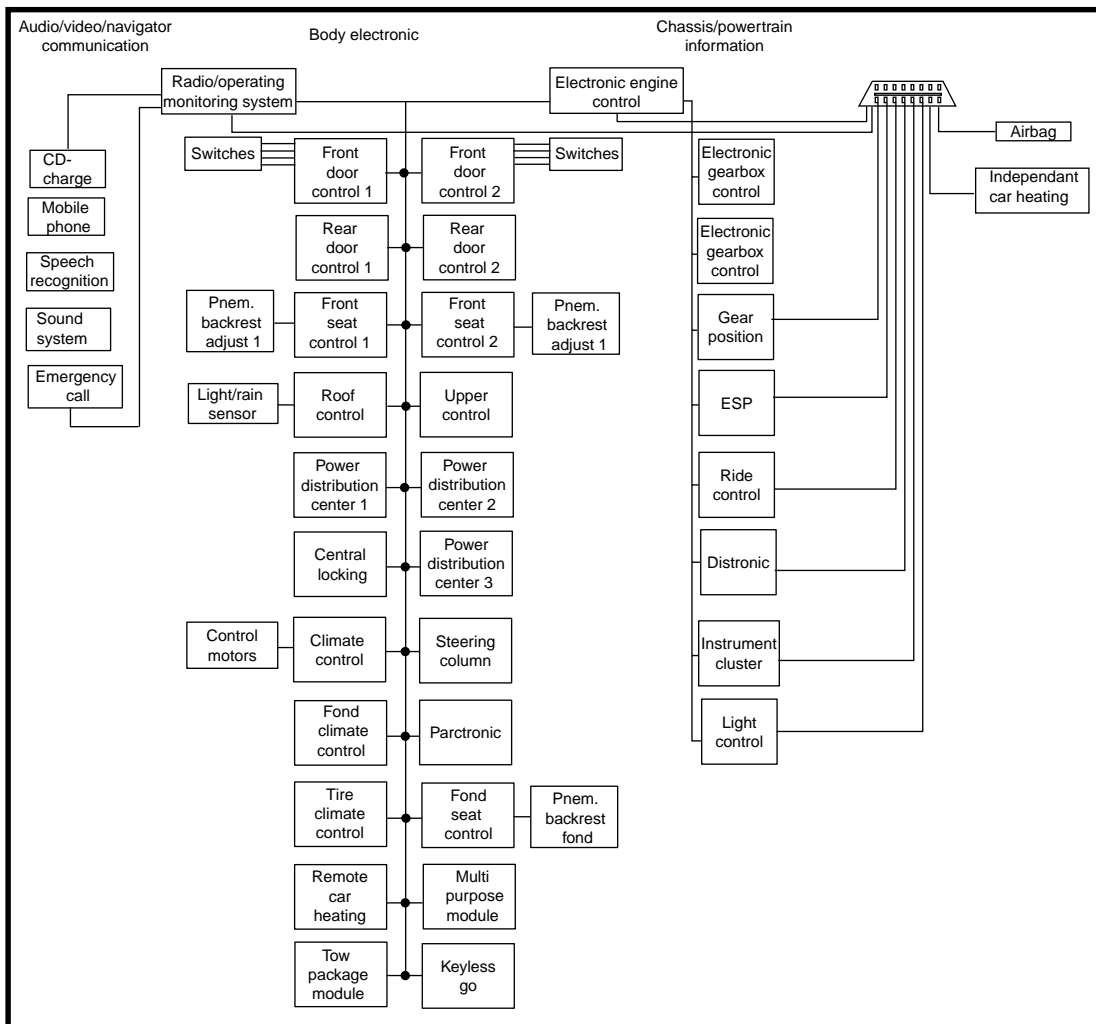


Figure 1—Mercedes S-Class automotive electronics is all about microcontrollers and networking.

and CAN may be eliminating some wires, but in many cases, they're inspiration for new features that (you guessed it) require new wires. For example, in my '99, even the stereo is connected to J1850, offering speed-sensitive volume control.

The bulk of the wiring challenge doesn't lie with such whimsically brainy features. Rather, it's the low-tech stuff like switches and lights that are still mainly wired point-to-point like in yesteryear.

Forget electronically variable valve timing, active suspension, drive-by-wire, and the rest and consider something mundane like a driver-side door. It's home to a growing litter of switches (windows, lock, and mirrors), motors (windows and mirrors), actuators (locks and folding mirrors), lights (courtesy, marker, switch, and turn signal in mirror), and so on. The situation is repeated for other major assem-

blies such as seats, climate control, and lighting. J1850 and CAN are overkill, but running dozens of wires hither and yon hardly seems like progress. What to do?

LITTLE NETWORK

Actually, LIN stands for Local Interconnect Network, not little network, but you get the idea. It's an econobox Class A network for all those proliferating lights, switches, and motors at a low cost that CAN can't match.

LIN is the proposal of a largely European consortium of automakers (Audi, BMW,

Volvo, and VW) with some U.S. participation from recently married DaimlerChrysler and U.S. automotive-electronics kingpin Motorola. The specs and first generation tools were originated by the final consortium member, Volcano Communications Technologies AB.

LIN incorporates many design features and eschews others, in a manner unique to the task at hand. It's a simple UART-based (ISO 9141 NRZ, in auto spec terms) scheme that sacrifices speed and fancy desktop network wannabe pretensions in favor of simplicity and low cost.

Electrically, LIN is a single-wire (up to 40 m), wired-AND arrangement (like

I²C), where each node is connected by a pull-up resistor to the bus and any node can transmit by pulling the line low (see Figure 2).

The LIN bus runs off the car battery (12 V), thus, eliminating many electrical and wiring hassles right up front. The downside is that the bus needs to handle power surges up to 40 V, polarity reversal, and such, but that dirty work is handled by off- and on-chip

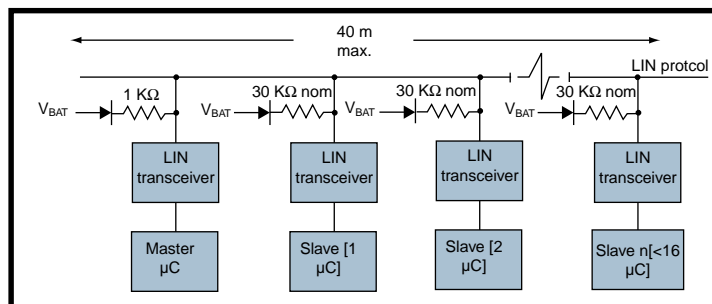


Figure 2—The basic wired-AND configuration is similar to I²C, but LIN is even simpler.

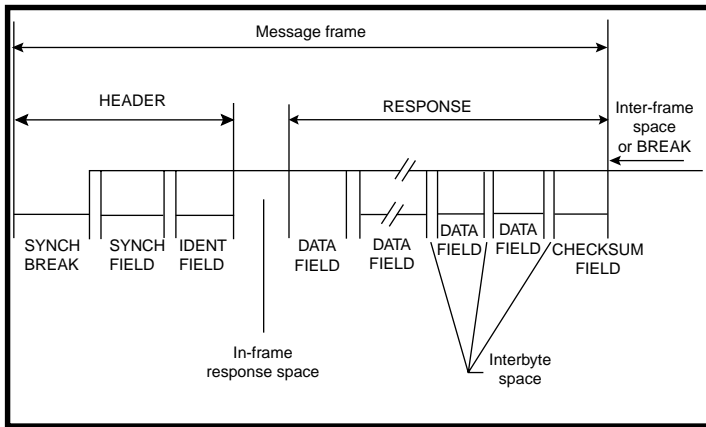


Figure 3—The master issues the header, and the appropriate responder (i.e., the master or a particular slave) gives back the data in due course.

transceivers.

The single-wire configuration and large voltage swing dictate a limited slew rate (typically 2 V/ μ s) to minimize EMI concerns. That, in turn, limits the maximum data rate to 20 kbps, which is just as well. It's fast enough to handle lights-and-switches duty, but unlike J1850/CAN, can easily be bit-banged by any 8-bit micro.

It's more than the speed limit that differentiates LIN. After all, it's roughly similar to current J1850 nets. Rather than raw bandwidth, it's the way the LIN bandwidth is used that sets it apart.

The major distinction is that LIN is a polled, rather than a contention, network. In LIN, slaves speak only when spoken to. Yes, LIN does require a master node to run things (i.e., poll), which represents a potential single point of failure. However, that's also true of many other electrical components in modern cars, and silicon is more reliable than most. If that's not good enough, a combination of explicit design redundancy and limp-home backup can be applied.

In recompense, the master-slave approach has some practical benefits. The master controls all communication. There is no arbitration or automatic retry. The message duration is either bound by the LIN specification or under the software's control. Put it all together and it's possible to achieve accurate timing and guarantee the worst-case service interval for each node.

In fact, what timing uncertainty LIN does have is largely a by-product of the data rate being allowed to drift quite a bit ($\pm 15\%$). Indeed, all nodes re-

synchronize their timing to the master on every message. The benefit is that, unlike the typical fixed data rate setup, the nodes don't need expensive, fragile crystals, but can instead use simple (albeit sloppy) RC clocks.

MINI-MESSAGE

Let's look at the overall way a LIN setup works. As shown in Figure 3, the basic unit of activity is a message frame. The master initiates a message transfer by issuing a header that consists of three parts.

First, the master holds the bus low for at least 13-bit times to get all the slave nodes' attention with the *SYNCH BREAK*. At the end of the *SYNCH BREAK*, the master sends the character 0x55, which (with alternating ones and zeros) acts as a *SYNCH FIELD*. Each node times the bit transitions in the *SYNCH FIELD* to adjust their own data rate. With everything in sync, the master sends the *IDENT* field, which includes a 4-bit identifier, a 2-bit designator that specifies the amount of data to transfer (1, 2, or 4 bytes), and two parity bits. The parity scheme seems like a bit of a head-scratcher but, besides simply checking the integrity of the *IDENT* field, also ensures that patterns 0x00 and 0xFF never appear on the bus.

At this point, the master has requested the appearance of a particular data item defined by the 4-bit identifier and length (1-, 2-, or 4-byte) designator. Now, a responder puts the data on the bus, protected by a simple checksum rather than a complicated CRC.

Typically, a slave node provides the response. However, it's important to

understand that the response isn't to the master, but to the bus. In fact, the master may supply the data in response to its own header, which allows it to send its own data on the bus. Furthermore, everything on the bus, the master and all slaves, has the opportunity to look at the response. For example, a door lock solenoid and window motor could both go into action upon detecting a door key lock/unlock message.

How does LIN deal with the possibility of collision? It doesn't. The protocol dictates only a single responder. Each node is required to monitor its own output in order to detect and log the occurrence of a collision, but that's it. There's no collision recovery (i.e., arbitration) mechanism built-in because that would compromise timing predictability. Of course, the master can request collision and other error-status reports (parity, checksum, or no response) from time to time if desired. Otherwise, the strategy is basically to try again next time (i.e., bad messages are just thrown away), which is arguably the best course of action when dealing with lights, switches, and such.

Notice that the spec doesn't limit the time allowed between the header and data or between each data byte. Instead, it simply dictates a maximum length for the entire message frame. That's nice because it means even a slow micro can handle the protocol, yet the overall message timing remains bounded.

LO BAT

With the growing population of electronic hangers-on, a car's quiescent power consumption becomes an issue. Indeed, my '99 has a gadget that preventively disconnects the battery before the accessories drain it to the point when it can't start the engine.

It's no surprise then that the only special message defined by LIN is *SLEEP* (*IDENT* field 0x80). The master, and only the master, issues the *SLEEP* command. In return, slaves can send a *WAKEUP* signal to the bus that is a single byte 0x80 (not an entire message). Upon detecting a *WAKEUP* signal, all nodes powerup and the master resumes normal messaging. Should the

RESOURCE

Microchip Technology Inc., "LIN Protocol Specification Implementation with PICmicro Microcontrollers," AN729, www.microchip.com.

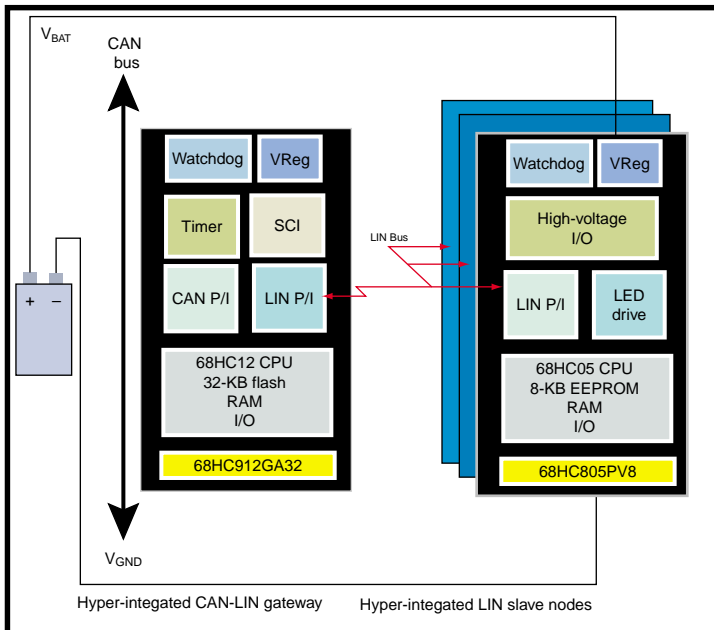


Figure 4—Full-fledged support for LIN by Motorola bodes well for the standard's acceptance.

master not respond to a *WAKEUP* call (within 128 bit times), the *WAKEUP* is issued up to three times before something is presumed seriously amiss.

CARAVAN

I can't be sure how LIN will fare, given the historically somewhat insular, proprietary, and arcane inclinations of automakers. As far as I'm concerned, everyone should use it. At the same time, the worth is as much in the concept (a simple network for simple stuff) as the particulars.

Motorola's doing its part with a whole laundry list of LIN solutions (see Figure 4). They run the gamut from higher-end 8- and 16-bit parts ('HC08s and '12s) that would work well as a LIN master and CAN gateway, to highly optimized app-specific slave nodes ('HC05s) that integrate everything, including the LIN transceiver, on-chip. I also notice that, though not a founding member of the consortium, Microchip has announced that it's getting on the LIN bandwagon. Microchip announced plans to deliver PICs with built-in LIN transceivers and a LIN protocol developers' kit next year. You can find a good app note for a PIC-based LIN driver on the company web site. LIN, along with the latest announcement of dsPIC (a 16-bit MCU with DSP features) and recent acquisition of TelCom Semiconductor, could

be just the ticket to boost Microchip's automotive biz to the heights they enjoy in other markets.

I don't know about Ford and GM, but between Motorola and Microchip, you're talking about a lot of 8-bit micros. Furthermore, there's some talk of applying LIN in other light, switch, and motor applications such as refrigerators, washers, dryers, and the like.

Whether it's your desktop, car, or refrigerator, the name of the game is the same. Let the networking begin. ☱

Tom Cantrell has been working on chip, board, and system design and marketing in Silicon Valley for more than ten years. You may reach him by e-mail at tom.cantrell@circuitcellar.com, by telephone at (510) 657-0264, or by fax at (510) 657-5441.

SOURCES

Local Interconnect Network
<http://www.lin-subbus.de/>

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

Motorola, Inc.
(602) 952-4103
Fax: (602) 952-4067
<http://www.mot-sps.com/automotive/index.html>

Circuit Cellar, the Magazine for Computer Applications. Reprinted by permission. For subscription information, call (860) 875-2199, subscribe@circuitcellar.com or www.circuitcellar.com/subscribe.htm.