Design flexibility key to simplifying the implementation of LIN

Nathan John and Franck Gauthier - November 10, 2002

Design flexibility key to simplifying the implementation of LIN

Nathan John of Cypress MicroSystems and Franck Gauthier of CrealieLogiciel Enfoui describe how PSoC microcontrollers can provide help for designers using the LIN bus.

Automotive manufacturers face many problems when implementing new designs. One problem that grows more difficult each year is accommodating the increasing number of electronic subsystems in the automobile. Systems are being added each year, such as occupancy sensors, seat position motors and rain sensors.

Simply adding wires to connect these systems increases the weight of the wiring harness, which significantly contributes to the overall weight of the vehicle. This is troublesome to auto manufacturers, who are constantly striving to reduce weight and improve fuel efficiency of the vehicle.

Automotive manufacturers have begun using networking techniques to attack this problem. Connecting many of these automotive subsystems over a single wire can reduce the number of wires. The two most prevalent digital bus standards in use today for automotive applications are Control Area Network (CAN) bus and J1850. However, an alternative standard, known as the Local Interconnect Network (LIN) bus, is gaining momentum in the automotive marketplace. This article will outline a new way to implement the emerging LIN bus standard, offering substantial advantages over traditional solutions.

Created by the LIN Consortium - a collection of automotive, software and semiconductor manufacturers - LIN is a class A (low speed) bus with a maximum speed of 20kbaud. The bus has a guaranteed latency time built into the protocol, even though it is not designed specifically for safety-critical applications. The most significant advantage offered by LIN bus is low-cost of implementation. It is estimated that the cost to implement a LIN node will be half the cost to implement a similar CAN bus node.

The LIN bus protocol is based on the common UART byte interface. Having the basic transmission follow the UART format allows many common microcontrollers to be used as nodes on the LIN bus. The LIN bus protocol specification 1.2 defines three standard baud rates: 2400 baud, 9600 baud and 19200 baud. Communication is based on a master/slave mechanism.
The bus is composed of one master node, and one or more slave nodes. All arbitration and collision management takes place in the master node, to further simplify and cost reduce the slave nodes.

The physical layer for LIN is a single wire, and the outputs of all of the nodes are logically ANDed together. The bus has two logic values: 'dominant' and 'recessive'. The bus is pulled high through a resistor, and any node that is outputs a bit value of 0 is said dominant, and actively pulls the bus down to ground level. If a node outputs a bit value of 1, it is said to be recessive, and is not affecting the bus. During each bit time, every node must check the state of the bus for fault conditions and report them to main application of the node if they are found.

All communications on the bus take the form of messages, which have a defined format known as the message frame. A diagram showing the format of the message is shown in figure 1. The message frame is composed of a header and a response. The header is further broken down into three fields:

- The synchronisation break field, composed of thirteen dominant bits (0) and at least one recessive bit, which indicates the beginning of a frame.
- The synchronisation field, which allows a slave to be synchronized on actual master baud rate.
- The identifier field, which identifies the requested message and the length of the response field.

![Diagram of the LIN Bus message frame format](image)

Only the master node can initiate a message by sending a header field that is received by all nodes. Each slave node analyses the header, and must be ready to send or receive data during the response field of the frame. The identifier field within the header informs all slave nodes in the network of the appropriate action to take. Such actions include:

- receiving bytes transmitted in the response field,
• transmitting bytes in the response field
• do nothing.

Using a microcontroller with a built-in UART, encoding and decoding the bytes that comprise the data fields are simple to implement. However, there are other aspects of a LIN bus implementation that require additional resources.

One example is using the synchronization field to determine the bit rate, which every slave node must do at the start of each message. This requires a hardware timer when using a traditional microcontroller. Additional hardware timers and counters may also be required to design slave nodes that detect several error conditions.

When you add up all of these aspects of the LIN bus protocol, it either requires many additional hardware resources, or significant software overhead to implement.

When implementing the LIN bus specification, designers can choose from a plethora of microcontroller options. Traditional microcontrollers offer a fixed set of hardware peripherals that can assist in particular tasks. The goal of the manufacturer is to provide a set of peripherals that meet the needs of a set of applications.

However, a flexible new microcontroller architecture makes the implementation of LIN bus communication much easier than traditional devices.

The Programmable System on Chip (PSoC) microcontroller has blocks of programmable analog and digital blocks that can perform a variety of functions, including many standard peripheral functions. The designer can use these blocks to configure the exact peripheral set for the application.

The digital blocks can be used to perform all the hardware functions required by the LIN bus interface, which relieves the CPU from performing these functions and limits the processor usage.

In addition to limiting CPU usage by the LIN protocol, it is also beneficial to limit the number of digital blocks used. Using fewer digital blocks for LIN bus functions makes more available for other functions in the application. The PSoC MCU has a unique capability that allows the designer to maximize resource usage at any time. This capability, called dynamic reconfiguration, allows a designer to create multiple configurations that perform different functions at various times during the operation - essentially re-using digital blocks.

The information that determines the configuration is stored in Flash memory, and is transferred to RAM based registers at boot time. Because of this, the user code can change the values in these registers at any time, which will change the configuration of the PSoC blocks.

Dynamic reconfiguration is extremely beneficial during a LIN bus implementation. The LIN bus transmission has three distinct phases:
1. header transmission/reception,
2. data transmission,
3. data reception.

Each of these phases uses a different configuration to save block resources. Although the functions performed are slightly different between the master and the slave node implementation, the total number of blocks used is reduced from eight to three in both cases.

For a master node, these functions are:

- Synchronization break field transmission: the length of this field (13 dominant bits and 1 recessive bit), is not a standard eight bits wide, and therefore requires a baud rate generator, a timer and a counter to implement.
- Data byte transmission: the synchronization field, the identifier field, the data fields and the checksum field are each one byte fields, and can be implemented using a standard serial transmitter and a bit time counter.
- Data byte reception: the data fields and the checksum field are also transmitted in a byte wide manner, and can be implemented using a standard serial receiver and a bit time counter.
- For a slave node, theses functions are:
  - Synchronization break field detection and timing measurement of synchronization field: this is used to re-synchronize the clock of the master and slaves, which is done by measuring time intervals between signal edges. This requires a timer and a counter to implement.
  - Like the master node, data field and checksum field transmission is done with a serial transmitter and a bit time counter. Data reception is done with a serial receiver and a bit time counter.

For each function, a hardware configuration is defined to include hardware blocks, I/O definitions and interconnects. During a frame transmission, each node switches from one configuration to another in order to manage the exchange. For example, the slave is first configured to detect the synchronization break field and to synchronize on the master clock. If synchronization is successful, it switches configurations to receive the identifier field. If it has to answer this message, it again switches configurations to transmit data and checksum fields.

This instant reconfiguration is possible only if the time needed to switch from one configuration to another is short enough to avoid data loss. Based on the available hardware resources and an optimised design of the configuration routines, this time is below 50µs, which is less than a bit time at 19200 bauds.

There are several benefits to using the PSoC MCU for LIN bus applications. First, the code memory required is small (approximately 1.5Kbytes for a slave node and 1Kbytes for a master node), which leaves most of the program area available for the application program. The CPU load needed for the driver management is very low, at less than 10%, and most of the hardware PSoC blocks are left for the application to utilize. The user can implement either a master node, or a slave node on any of the PSoC MCU devices, making this a universal solution for LIN bus applications.

Cypress Microsystems and Crealie Logiciel Enfoui have taken the implementation of a LIN bus
network, with one master node and two slave nodes, and created a reference design board (figure 2).

Fig 2: Reference design board of LIN bus network

In this design, the individual nodes pass messages between each other indicating input conditions to the node (switch positions), and displaying outputs (lighting LEDs). This reference design is available as part of a complete design kit.

Published in Embedded Systems (Europe) November 2002