Implementation of a Simulated Washing Machine with an Integrated Control Panel and Intra-Appliance Local Interconnect Network Bus

David VanZoest
Duraswitch Industries, Inc.
234 South Extension Road
Section 103
Mesa, AZ 85210
Telephone: 480-586-3300
Telefax: 480-586-3375
E-mail:david@duraswitch.com

David Otten
Microchip Technology Inc.
2355 West Chandler Blvd
Chandler, AZ 85224-6199
Telephone: 480-786-7200
Telefax: 480-899-9210
E-mail: david.otten@microchip.com
IMPLEMENTATION OF A SIMULATED WASHING MACHINE WITH AN INTEGRATED CONTROL PANEL AND INTRA-APPLIANCE LOCAL INTERCONNECT NETWORK BUS

ABSTRACT

A simulated washing machine with an integrated appliance control panel and intra-appliance Local Interconnect Network (LIN) bus capability has been designed and demonstrated. The control panel allows the integration of push-button and rotary controls into thin, low profile assembly with superior graphics and electrical circuit design flexibility. LIN is used as an intra-appliance network to replace bulky wire harnesses, improve manufacturability and reliability, and reduce costs. Only three wires are necessary for communication inside of the appliance. The network can be used to create appliances that are capable of communicating via a home network to manage peak energy usage and add a new level of troubleshooting and field upgrades.

INTRODUCTION

Home networks are becoming commonplace in new home construction. Prospective homeowners find networks an increasingly available option. Current homeowners are also getting their hands dirty drilling, cutting and patching drywall in an effort to add their own network. The interest in these networks is mainly driven by a desire to network multiple PCs together to share resources such as printers, an Internet connection or some type of data storage. It is estimated that there could be as many as 10 million U.S. home networks by 2005. Although most consumers probably do not yet consider networking beyond computing, it is not difficult to see how this trend offers opportunities to extend home networks into other areas such as environment control, security, lighting or appliances.

Internet-ready appliances, under development or available today, offer energy management, flexible features, custom cycles, and even pay-for-usage. Refrigerators may have integrated Web browsing, inventory and expiration date tracking; microwaves could download custom cooking cycles or recipes that use particular ingredients; And washing machines with built-in diagnostics might automatically schedule repairs with the local service department. These are exciting prospects and offer insight into the direction of future appliance capabilities that expand well beyond their traditional roles.

After a long period of technological evolution, it seems that home appliances are on the precipice of a connectivity revolution. How will appliance manufacturers respond to the opportunity? Manufacturers must still reconcile competing demands of feature content, cost, manufacturability and flexibility, but the delicate balance is undergoing a change, especially when deciding upon an appliance electronic
control architecture. Material and manufacturing costs used to drive appliance control selection, but flexibility is getting more emphasis than before.

Although the use of a high performance microcontroller within the appliance control isn't new, the use of multiple microcontrollers was typically viewed as an unnecessary expense. As it turns out, using a decentralized, multiple-microcontroller appliance control is an effective and cost-efficient solution that offers new levels of flexibility for both consumers and manufacturers. The availability of low cost microcontrollers has made the choice of distributing the control of various functions within an appliance a viable option. Logic and power electronics can be co-located with the associated electromechanical components such as the user control panel, valves, heaters and motors within the appliance to create a modular approach to design and assembly. An innovative integrated user control panel with push-button and rotary controls in a single thin, low profile assembly further enhances design flexibility. Communication between components is accomplished through a serial data network, which greatly reduces the size and quantity of wiring required. Most components would require only three wires for the serial data bus and two or three additional wires for the high power circuits, if needed. The Local Interconnect Network (LIN) protocol is a simple, cost effective choice for the communication bus. The use of this network is an enabler for communication to other appliances in the home or to the world via the Internet. A simulated washing machine was successfully designed and demonstrated using these concepts.

**INTEGRATED CONTROL PANEL**

The integrated control panel makes use of novel construction techniques and switching elements. In this demonstration both rotary and pushbutton switches are integrated into one control assembly. Many appliance users prefer more ergonomically friendly rotary selectors that simply require pointing the knob at the desired function, rather than requiring multiple button presses to select a function. Instead of adapting the bulky, high current switches used in the past, this panel is optimized to interface with a low current, low voltage microcontroller based electronic control system. The operating simplicity of older style appliances is combined with the more sophisticated operating algorithms available today.

The graphics on the panel are created using a polyester overlay utilizing a low cost, highly flexible screen printing manufacturing processes. The appearance of the panel is limited only by the imagination of the graphics designer. Usually a minimum of three colors are used with a glossy, non-surface textured, polyester. Additional colors or surface texturing printing passes will increase the costs of the panel. As new appliance features or alternative graphics are created, they
can be accommodated without retesting the mechanical and electrical content of the panel.

The construction of the panel results in a thin, low profile cross-section that provides additional design packaging flexibility. The connections route to sides of the switching components via the printed conductors on the flexible circuit substrate, thereby avoiding bulky plug-style connectors with discrete wires.

The electrical contact patterns and interconnections for the rotary and pushbutton switches are created with a screen printing process similar to that used for the graphics, except using electrically conductive and insulating inks. A variety of different circuit functions can be created with a common set of mechanical components and different print patterns. The same family of mechanical rotary components can be used to create a discrete selector, quadrature encoder or analog potentiometer. In this panel an absolute position multi-bit encoder was created.

The same flexibility applies to push-button switches and LED indicators. Both can be interconnected in either a discrete or matrix configuration. The matrix saves connector input/output pins, but sacrifices some control flexibility.

An interesting feature provided by some microcontrollers is the ability for LEDs and switch inputs and outputs to share common input/output ports. The port is time shared to provide a circuit path to ground for an LED and also monitor the switch input status, as shown in Figure 1.

![Switches and LEDs Sharing I/O Pins](image)

**Figure 1**

**INTRA-APPLIANCE NETWORKS**

There are basically two camps of appliance control, centralized and decentralized. Both perform the needed tasks of processing, sensing, indication, control, timing and external communication, but do so with different architectures. Centralized control uses a single microcontroller to manage every aspect of appliance operation: receiving input from the panel, sensing water level, changing motor speed, etc. A decentralized approach to appliance control, as the name implies, distributes control throughout the appliance. It typically employs one more powerful microcontroller and multiple smaller microcontrollers to perform the same tasks as the single, much larger controller used in the centralized scheme. Each microcontroller and associated electronic and electromechanical components
comprise a node and handle a subset of the control system tasks, which are usually organized by major components within the appliance. One microcontroller is designated as the master node for the intra-appliance network and issues commands to the slave nodes.

There are some advantages of using a centralized control system: only one microcontroller to program, fairly straightforward design, and an intra-appliance communication bus is not required. Disadvantages include: the need for a larger, faster, more expensive microcontroller to handle the entire appliance operation, and many long wires to connect needed signals. If multiple functions require simultaneous control, more precise timing is required. In this case, the microcontroller needs to not only have more I/Os, memory and peripherals, but also must run at a higher frequency. This adds to component and printed circuit board costs for the additional precision components and trace layouts that are required. A centralized control system is less flexible if additions to the appliance’s functionality are needed.

The advantages of a decentralized architecture are significant in terms of costs, manufacturability and flexibility. Costs are reduced as fewer and shorter wires are used within the appliance. Switches and sensors that each require dedicated power, ground and information wires to the central microcontroller can use a shared bus to communicate. The bus consists of the same three wires that connect each node within the appliance, power, ground and data. Manufacturability is improved; nodes can be connected in a daisy-chain fashion, simplifying installation and reducing the number of connectors. The disadvantages of using the decentralized approach is that there are multiple microcontrollers to program, a network must be designed and microcontrollers need to run an intra-appliance protocol in addition to their assigned tasks.

A LIN-based, decentralized control system is easily expandable. Nodes can be configured via firmware to perform multiple functions using the same hardware. For example, a temperature unit that uses three temperature settings on a standard model can be programmed to use 10 settings for a high-end model. Software changes can also add or optimize features, such as adding additional wash cycles, or incorporating new components such as a water heater or communication node within the control system. Leveraging multiple designs from a single design platform is being used successfully by the automobile industry and can also benefit the appliance industry. Features can be added without impacting production and reconfigurable components can be created, thereby reducing inventory.

LIN AS AN INTRA-APPLIANCE NETWORK

A consortium of automobile and semiconductor manufacturers developed the LIN protocol. It was
designed to be a low-cost, robust way of exchanging information within a vehicle in “human-time.” CAN has emerged as the standard high-speed protocol for automobiles and LIN complements it by offering a cost-effective way of extending the automobile network into components, which either do not require the speed, or can not justify the cost, of a CAN controller, such as door or lighting systems. Appliances also benefit from control systems that are flexible, robust and most importantly, inexpensive and are well suited to use the LIN protocol.

Adding LIN to a control system is inexpensive. The physical layer used by LIN is comprised of readily available components. A LIN network node can be implemented with as little as a microcontroller and a transceiver and without expensive, precision components. LIN microcontrollers with an internal transceiver are available from Microchip Technology and others, avoiding the additional board space required for a stand-alone LIN transceiver. Many of the cost saving attributes of the LIN protocol stem from the low 20K baud maximum transfer rate and slave nodes self-synchronizing to the master on every message. As a result, slave nodes can use the microcontroller’s internal oscillator option, eliminating the expense, board space and supporting components of a resonator or oscillator.

The firmware for the LIN protocol is efficient. It requires less than 200 instruction words of program memory and only five bytes of data memory. Implementing an interrupt-based version frees up processor time for other tasks until bus is active and minimizes the overhead of running the LIN protocol. With its low overhead, the protocol can typically fit within a microcontroller’s available resources.

LIN adds flexibility to the manufacturing and product definition process. New features can be added or changed via software with new or different hardware daisy-chained into the bus. Software can be developed to allow hardware to be used in multiple appliance models, thereby minimizing component part numbers. Nodes can be programmed to control multiple components. If a component changes in production, a software change allows the node to use the new component. New nodes can be easily added by programming the required commands and actions and connecting it to the shared LIN bus. No changes are required to the existing network. Although cost analysis does not factor in flexibility, using LIN simplifies manufacturing and reduces costs.

An important aspect to the consortium when defining LIN is that it be robust, survive in electrically noisy environments and not generate excessive EMI or RFI noise. To reduce emitted noise, transceivers use current supplies for slope control without additional filters. LIN transceivers are designed to operate between 8 and 18 volts. At these automotive voltages, noises on the LIN bus become a small percentage of the overall voltage range reducing their affects. Also, the voltage...
can often supply power to the entire node, minimizing the number of power supplies and further reducing costs. Because LIN is currently in use by automobile manufacturers, appliance manufacturers can leverage from the automobile industry’s experience. When compared to other potential intra-appliance protocols, LIN is easier and less expensive to implement than CAN and more robust than UART, I²C™ or SPI™ protocols. These attributes make the LIN protocol an effective solution for automobile manufacturers and an attractive option for appliance manufacturers shopping for an intra-appliance protocol.

THE LIN PROTOCOL

LIN messages include multiple levels of data verification. Messages include parity bits for the identifier byte and a checksum of the data bytes. The parity and checksum are calculated by the sending node and are verified by the receiving node. Also, LIN transceivers are able to read the bus during transmission, which allows them to confirm transmitted data is accurate. Nodes can therefore respond to bus contention or flag a damaged transceiver. The LIN specification defines the following types of errors: bit error, checksum error, parity error, no-response-from-slave, synch-field-error and no-bus-activity. How the errors are handled is not specified, which allows the designer to assign the appropriate response to each type of error. Although an appliance may not require this extent of data verification, it does allow the designer to choose the appropriate level for their application.

The master node initiates a LIN message by sending a synch break, synch byte and identifier byte. A valid synch break is sent when the master holds the bus low for at least 13 bit-times to alert the slave nodes to an upcoming message. Since the master can select an optimal baud rate between 1k and 20k, the synch byte is comprised of a series of eight alternating ones and zeroes which allow the slave nodes to adjust their own internal timing to the master. Identifier bytes are sent by the master node and serve multiple purposes. The master node can request information from a slave node and, at the same time, initiate an action within other nodes. Each identifier contains a unique, four-bit message identifier, a two-bit field length and two parity bits. The identifier can be thought of as a

![Message Frame Format](Figure 2)
command sent by the master while the data bytes serve as parameters or information. A message can have two, four or eight data bytes as indicated by field length bits. The final two bits are the parity bits used to validate the identifier byte.

An interframe response delay is provided after the identifier byte is sent. Its purpose is to give the slave nodes time to decode the identifier byte, validate the identifier parity, determine whether it needs to send, receive or listen to the upcoming data bytes, and if needed, fetch new data and prepare a response. If a response is requested, the appropriate slave node sends the requested information within the number of data bytes specified in the field length and follows it with a checksum calculated on all transmitted data bytes.

**A REFERENCE DESIGN: WASHING MACHINE USING A LIN NETWORK**

The purpose of the washing machine demonstration is to show an example of an appliance using a control system based on the LIN protocol. Although it is a simplified example, it does contain a full suite of selectable cycles, agitator motor, hot and cold water valves, lid position switch, end-of-cycle buzzer, water level and temperature sensor input, and a variety of indicators. LEDs of various colors convey current water level, hot and cold water valve position, heater power, motor speed and direction. The functionality of the simulated washing machine was developed by studying a do-it-yourself washing machine repair manual. The graphics for the control panel were developed based on a generic sales demonstration panel. No attempt was made to exactly copy a current production washing machine. The sequential machine steps were broken up into relative time intervals to facilitate a working demonstration. An example operating cycle definition is shown in Table 1.

<table>
<thead>
<tr>
<th>Knob Selection: Colors</th>
<th>Knob Setting within Colors</th>
<th>Heavy</th>
<th>Normal</th>
<th>Light</th>
<th>Rinse</th>
<th>Spin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fill @ Wash Temp</td>
<td>1 to 5 1 to 5 1 to 5 1 to 5 Skip Skip</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Agitate - High</td>
<td>3 3 3 Skip Skip</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Half Drain</td>
<td>1 1 1 Skip Skip</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fill @ Wash Temp</td>
<td>1 to 5 1 to 5 1 to 5 1 to 5 Skip Skip</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Agitate - High</td>
<td>6 4 2 Skip Skip</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Agitate - Lo</td>
<td>3 3 3 Skip Skip</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Half Drain</td>
<td>1 1 1 Skip Skip</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fill @ Rinse Temp</td>
<td>1 to 5 1 to 5 1 to 5 1 to 5 1 to 5 Skip</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Agitate - High</td>
<td>1 1 1 1 Skip</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Drain</td>
<td>1 1 1 1 Skip</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spray @ Rinse Temp and Drain - Lo</td>
<td>1 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fill @ Rinse Temp</td>
<td>1 to 5 1 to 5 1 to 5 1 to 5 1 to 5 1 to 5 1 to 5 Skip</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Agitate - High</td>
<td>1 1 1 1 Skip</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Drain</td>
<td>1 1 1 1 Skip</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spray @ Rinse Temp and Drain - Lo</td>
<td>1 1 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Drain</td>
<td>1 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Example Operating Cycle Definition Table 1*

Within the washing machine is a LIN network consisting of one master node connected to the control panel and three slave nodes used to control the motor, water valves and water heater components. The master node serves three purposes; it executes the cycle sequence, masters the LIN network and
monitors the control panel. It is programmed with all of the cycle information and controls cycle timing by sending commands to the slave nodes at the appropriate times. Cycle sequences are executed using six commands: OFF, FILL, DRAIN, HALF-DRAIN, AGITATE and SPRAY. Each command is assigned a unique identifier, designated to use two, four or eight data bytes and whether to read, write or listen to the data bytes. This information is stored in look-up tables within each microcontroller’s program memory. An identifier assigned in the master as a read will be defined as a write identifier in one slave node and as a listen for the other nodes.

LIN uses a common bus to communicate; therefore, all nodes see the same information at the same time. In contrast to an address-based protocol, LIN is command-based, meaning multiple slave nodes can respond to a command in different ways. For example, in this demonstration, the FILL identifier causes the Valves node to open the appropriate hot and cold water valve, the Motor node to turn off the motor, and the Heater node to turn the heater on or off as needed. If an address-based protocol were used, the master node would have to issue multiple commands to each node to achieve the same effect.

![LIN Node Allocation](image)

**LIN Node Allocation**

*Figure 3*
Taking a closer look at the commands used within the control system’s LIN network, it can be seen how cycles are run. The OFF identifier terminates all instruction execution except DRAIN and HALF-DRAIN which are polled for completion. OFF causes the Heater node to turn off the heater, Motor node to stop the motor and Valves node to close both water valves. The FILL command passes water temperature and level information within the message’s data bytes. It causes the Heater node to turn on the heater for hot or warm water and off for cold, and the Valves node to open the appropriate hot or cold valve and increment the water level. The DRAIN identifier makes the Motor node spin the motor counterclockwise at high speed and the Valves node to decrement the water level to empty. HALF-DRAIN also spins the motor counterclockwise at high speed but this time the Valves node drains only half of the water. The AGITATE command passes a high or low motor speed. The Motor node then spins the motor clockwise at the requested speed. SPRAY specifies both motor speed and water temperature and causes all three slave nodes to respond. The Heater node turns on the heater for hot or warm water and off for cold, the Motor node spins the motor counterclockwise at the specified speed and the Valves node pulses the appropriate hot and cold water valves. LIN streamlines the task of executing a cycle sequence by allowing the designer to create an efficient and powerful instruction set.

Since filling the washing machine depends on the available water pressure, the FILL command cannot be timed. This is also true for the DRAIN and HALF DRAIN commands. Therefore, the master node queries the Valves node to determine when the appropriate water level has been achieved. Extending this concept, status identifiers were created that allow the master to request information from the three slave nodes. The master node uses the STATUS-VALVES command to periodically poll the Valves node after sending a FILL, DRAIN or HALF-DRAIN command. Status commands also include current state information. Encoded within STATUS-VALVE command is a bit to indicate whether the FILL or DRAIN is complete, as well as hot and cold valve positions and current water level. The STATUS-MOTOR command causes the Motor node to send motor power, direction and speed information. Lastly, the STATUS-HEATER identifier receives heater power and analog input values for water level and temperature from the Heater node.

**HOW TO IMPLEMENT AN APPLIANCE CONTROL SYSTEM USING LIN**

Implementing a LIN control network is greatly simplified by assessing the requirements of the control system and defining the various message formats used by the network. First determine how many cycles are needed and organize them into matrix.
Look-up tables work well for storing and retrieving large amounts of cycle information because they store the data efficiently and allow it to be easily accessed. Next, analyze the cycles and reduce them to their smallest instruction set, assign identifiers and determine which parameters will be passed via data bytes. Remember that the field length bits within the identifier define the number of data bytes assigned to each message. Finally, establish message formats and document them because they will be frequently referenced during the design process.

The control panel uses a five-bit absolute gray code selectors for super, regular and gentle cycles, whites, color and delicate clothes and heavy, normal, light, rinse and spin cycles, as well as water level and temperature options. By assigning numeric values to each selection, the starting operating cycle location can be calculated and found in a software look-up table. In this demonstration, the panel settings are periodically scanned and the first cycle location is calculated after the start button is pressed.

The next task is to organize the components into nodes and select a microcontroller to run each node. Microcontrollers with internal transceivers are available, but not in every configuration. If a suitable one is not found with an internal LIN transceiver, a stand-alone transceiver can be used. For this washing machine demo, nodes were created for the heater, valves, motor and panel. Due to the LIN protocol’s low bandwidth requirements, there are usually multiple candidates for the node that could be chosen as the master node. Because indicator lights and switches do not require much processor time, the control panel was made the master node. If the master microcontroller had been embedded within the panel, a three wire LIN connection could replace the high pin count connector.

When designing a slave node, every identifier is assigned a response. This includes whether it will read, write or listen to the data, how many bytes of data will be included within the message and what action, if any, is required. If information is to be transferred, data needs to be sent in a timely manner to avoid a time-out error. Therefore, it is advisable to shadow transmitted data by routinely updating and storing it on a regular basis. By doing so, the most recently fetched data can be sent without delaying the message response by gathering and formatting data. As part of the response, determine whether information needs to be fetched locally or from another node, and whether commands need to be timed by the master, slave or polled for completion. If polling is required, add status request identifiers to determine when the slave’s tasks are complete. By including current node status, diagnosis and system monitoring features can also be added.

Once all of the planning is complete, the control system may be implemented from the bottom up in two steps. Write the code to control the component, and after it is verified, add the LIN firmware. Ensure all required
tasks execute correctly, sensors can be read and indicators updated, etc. After the node is working correctly, integrate the LIN firmware into the microcontroller and assign actions to identifiers. Verify each identifier works correctly and unused identifiers are assigned into listen mode. Common implementation mistakes are to assign identifiers the wrong action, an incorrect number of data bytes, or to read data when it should send data or vice versa.

Since new nodes can be added without modifying the existing network, it would be possible to offer a communication port as an optional upgrade. It would allow the appliance to connect to an energy management system that could monitor household consumption and schedule appliance start times. Consumers could purchase new wash cycles or use it to upgrade their available cycles. Service technicians could download diagnostic information, maintenance history and component identification codes. Communications could be extended to the Internet, allowing the appliance to request a maintenance call or inform the owner of available upgrades or recalls. The trend for future appliances is improved communication that would benefit both the consumer and manufacturer.

**CONCLUSION**

Appliance manufacturers must balance functionality, implementation cost, manufacturability and flexibility. This balance changes every generation and trends toward increased flexibility and improved communication. Implementing a LIN based control system within appliances can add a new dimension of flexibility that benefits all aspects of manufacturing. A control panel with flexible graphics and electrical circuit options complements this approach.

**REFERENCES**

1. www.datadoctors.com
2. Chu, Damon, “Electronic Controls in Appliances: Central vs. Decentralized Processing”, Microchip Technology Inc.
8 U.S. Patent Numbers: 5,666,096;
6,069,552; 5,867,082; 5,990,772;
6,130,593; 6,069,545; 6,023,213;
6,137,387; China (PRC) Patent Number
ZL96110385.X, Taiwan (ROC) Patent
Number NI090979

9 www.repairclinic.com - How Things
Work: Washing Machine

10 www.duraswitch.com

11 www.microchip.com

Note: The Microchip name and logo, and PICmicro are registered trademarks of Microchip Technology
Inc. in the USA and other countries. I²C is a trademark of Philips Corporation. SPI is a registered
trademark of Motorola. All other trademarks are the property of their respective owners.