Using CAN Arbitration for Electrical Layer Testing

Sam Broyles and Steve Corrigan, Texas Instruments, Inc.

The Controller Area Network (CAN) protocol incorporates a powerful means of seamlessly preventing data corruption during message collision. This arbitration process and its relationship to the electrical layer variables are explained. Techniques to force message collision and test arbitration are demonstrated with strategies to leverage arbitration as a quantitative benchmark in safety-critical systems. The benchmark is then applied to several example systems and results provided for comparison.

Introduction

The ability of a Controller Area Network to manage message collision provides a unique proving ground for protocol compliance in any application. A means of determining a benchmark for a system's performance by measuring a network's ability to execute proper arbitration is developed in this example. It is demonstrated that while a CAN bus appears to be functioning normally, many arbitration errors may be unnoticed by system operators.

Arbitration Basics

Since any CAN node may begin to transmit when the bus is free, two or more nodes may begin to transmit simultaneously. Arbitration is the process by which these nodes battle for control of the bus. Proper arbitration is critical to CAN performance because this is the mechanism that guarantees that message collisions do not reduce bandwidth or cause messages to be lost.

Each data or remote frame begins with an identifier, which assigns the priority and content of the message. As the identifier is broadcast, each transmitting node compares the value received on the bus to the value being broadcast. The higher priority message during a collision has a dominant bit earlier in the identifier. Therefore, if a transmitting node senses a dominant bit on the bus in place of the recessive bit it transmitted, it interprets this as another message with higher priority transmitting simultaneously. This node suspends transmission before the next bit and automatically retransmits when the bus is idle.

The result of proper arbitration is that a high-priority message transmitted without interruption is followed immediately by a low-priority message, unless of course, another high-priority message attempts to broadcast immediately following the same message. Since no messages are lost or corrupted in the collision, data and bandwidth are not compromised.

Electrical-Layer Variables (bit timing requirements)

Each CAN bit is divided into four segments (see Figure 1). The first segment, the synchronization segment (SYNC_SEG), is the time that a recessive-to-dominant or dominant-to-recessive transition is expected to occur. The second segment, the propagation time segment (PROP_SEG), is designed to compensate for the physical delay times of the network as shown in Figure 2, and should be twice the sum of the propagation delay of the bus, the input comparator delay, and the output driver delay. The third and fourth segments, both phase buffer segments (PHASE_SEG1 & PHASE_SEG2), are used for resynchronization. The bit value is sampled immediately following PHASE_SEG1.

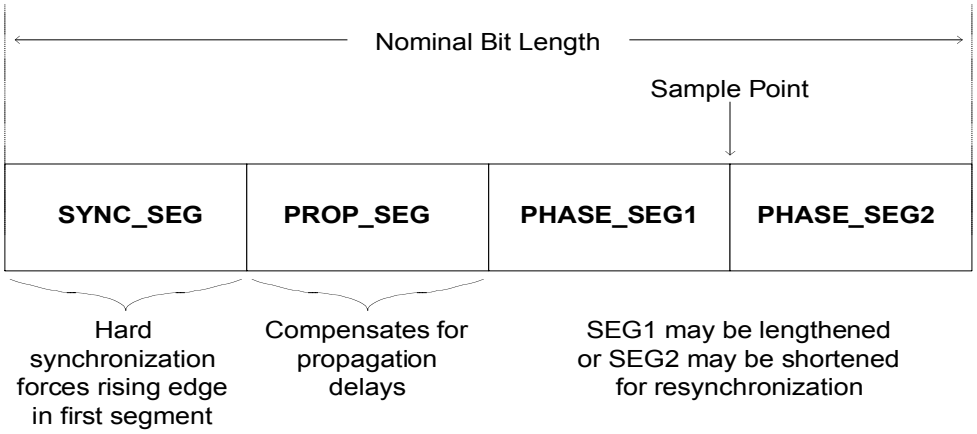

Figure 1. Partitioning of the Bit Timing Segments

The bit rate may be changed by either changing the oscillator frequency, which is usually restricted by the processor requirements, or by specifying the length of the bit segments in "time quantum" and the prescaler value. The prescaler value is multiplied by the minimum time quantum, which is the reciprocal of the system clock frequency, $1/f_{sys}$, to determine the length of a working time quantum. Bit time may then be calculated as the sum of each bit segment, and the bit rate may be calculated as the reciprocal of this sum.

Each node must perform a hard synchronization upon every recessive-to-dominant edge after a bus idle or received start of frame. Hard synchronization is a restarting of the internal bit timing to force the edge into the SYNC_SEG, where edges are expected to occur. Resynchronization is performed on all other recessive-to-dominant edges of other received bits by lengthening or shortening the PHASE_SEG1 or PHASE_SEG2 by one to four time quanta as specified by the resynchronization jump width. If the difference between the edge causing resynchronization and the SYNC_SEG exceeds the resynchronization jump width, the effective result is the same as a hard synchronization.

CAN Network Errors

CAN protocol specifies five different types of network errors. A transmitting node detects a bit error when it monitors a bit value different than it is transmitting; the reaction to this condition varies with the nature of the error. A stuff error occurs when the bit-stuffing rule is violated – a bit of opposite value must be inserted immediately following any series of five consecutive bits of the same value in a message. A cyclic redundancy check (CRC) error occurs when a receiving node receives a different CRC sequence than anticipated. (Note that all nodes independently calculate the CRC sequence from the data field). A form error occurs when a field contains an illegal bit value. Finally, an acknowledgement (ACK) error occurs when the transmitter does not monitor a dominant bit in the ACK slot to signify that the message had been received properly by another node as shown in Figure 2.

When a node detects a bus error, it transmits an error frame consisting of six dominant bits followed by eight recessive bits. Multiple nodes transmitting an error frame will not cause a problem because the first recessive bits will be overwritten. The result will remain six dominant bits followed by eight recessive bits, and cause the bus to be safely reset before normal communications recommence.

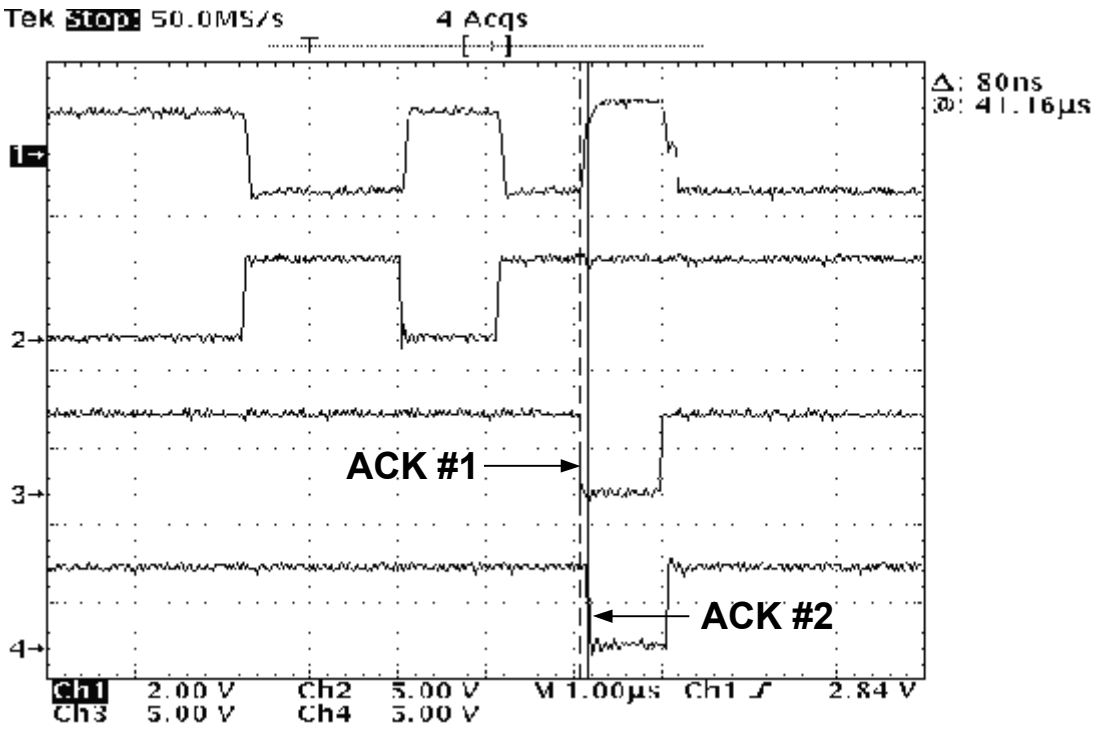

Figure 2.
Round trip propagation delay of network measured as the delta of the ACK bit response time between two nodes on opposite ends of the network.

The CAN protocol provides a means of fault confinement by requiring each node to maintain separate receive and transmit error counters. Either counter will be incremented by 1 or 8, depending on the type of error and conditions surrounding the error. The receive error counter is incremented for errors during message reception, and the transmit error counter is incremented for errors during message transmission (for further details, see reference 1). When either of these counters exceed 127, the node is declared "error-passive," which limits the node from sending any further dominant error frames. When the transmitted error count exceeds 255, the node is declared "bus-off," which restricts the node from sending any further transmissions. The receive and transmit error counters are also decremented by 1 each time a message is received or transmitted without error, respectively. This allows a node to return from error-passive mode to error-active mode (normal transmission mode) when both counters are less than 128. The node may also return to error-active mode from bus-off mode after having received 128 occurrences of 11 consecutive recessive bits. Overall, a network maintains constant transmit and receive error counters if it averages eight properly transmitted or received messages for each error that occurs during transmission or reception, respectively.

Analysis of Network Errors

As shown in Figure 2, the oscilloscope is an invaluable tool for observing bus status. For these experiments, a Tektronix 784D oscilloscope with Tektronix P6243 1 GHz single-ended probes are used. With careful choice of message identifiers and data fields, messages can be visually associated with the transmitting node. This guarantees observation of the participation of each node during an experiment. Additionally, the transmission lines from the controllers to the transceivers are useful for monitoring the

participation of each node during arbitration and ACK.

A more detailed record of bus activity may be found in the status and control registers of the processors that are reviewed after each experiment. Though the error counters are incremented and decremented through a complex series of rules, it is sufficient to note error status and types of errors that occur to assess bus performance under a set of experimental conditions.

To ensure that every message collision results in proper arbitration, it is required that the processor be programmed to identify the order in which each message is received. This is accomplished by checking for a receive message pending flag. For the purpose of these experiments, the processor of node A in Figure 3 is programmed to stop program execution upon the first event of improper arbitration.

Forcing Message Collision

While operating with a two-node bus, if one node enters the bus-off state, either the bus becomes silent or the other node continues to retransmit until the reception of a proper ACK bit is received. With three or more nodes, a proper ACK bit would be inserted by one of the remaining nodes and the bus simply becomes silent. Either case represents an experimental condition that would not be recommended for use in a final application.

With the three-node bus in Figure 3, message collision is forced by programming two nodes to respond immediately to a message from the third node. In these experiments, node A is programmed to send a data frame and wait for nodes B and C to respond. If nodes B and C respond in the wrong order, arbitration is not properly negotiated and node A does not retransmit. If nodes B and C do respond in the proper order, arbitration is properly negotiated and node A retransmits its message.

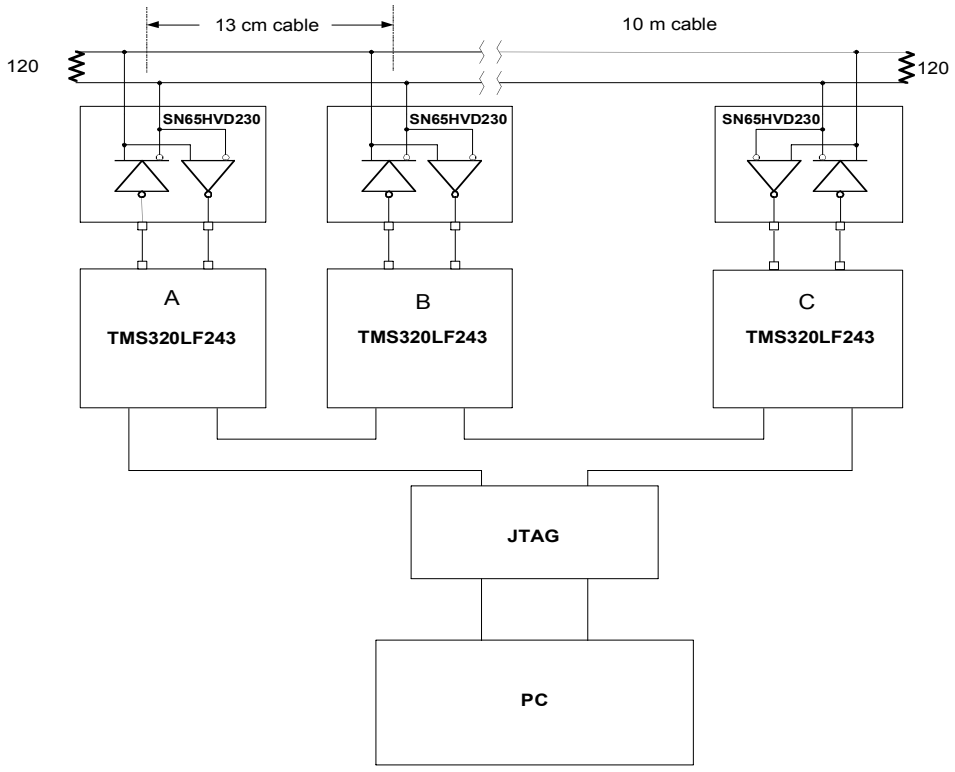

Figure 3. The Experimental 3-Node Bus

Figure 4 displays a signal capture of this system during arbitration. Channel 1 is the bus signal while channel 2 displays the output signal of the node with the highest priority, node A. Channel 3 is the output signal of node B, which is the lowest priority message on the bus. Channel 4 is the output signal of node C, which is the medium priority message. Notice that both nodes B and C participate in the ACK bit of node A and begin transmitting together. However, after just a few bits node B stops transmission until node C, the higher priority node, is finished.

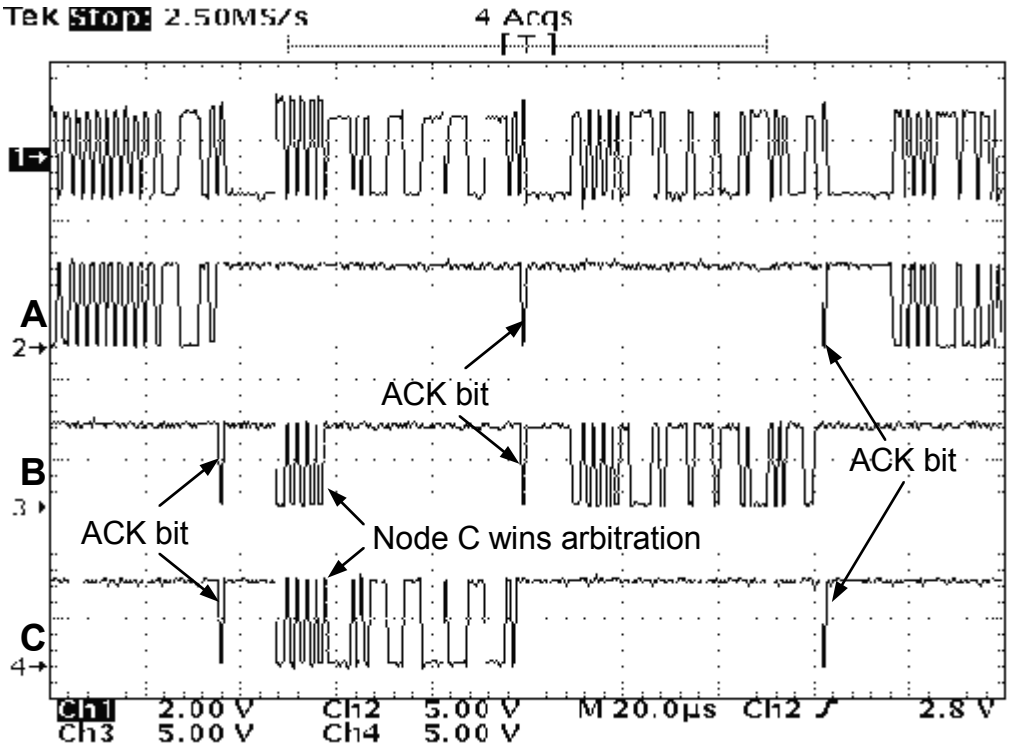

Figure 4. Arbitration Display

Experimental Network Systems

Three bus systems were tested to study arbitration and system performance. In each system, three nodes actively participate on the network. Node A sends the first data frame. Nodes B and C attempted to respond simultaneously and negotiated bus control. All messages have 29-bit identifiers and 8-byte data fields. Node A retransmits it's data frame and begins the process again if the dominant message from node C is received before the recessive message from node B. This continues for one million cycles unless node A receives the messages in the wrong order or until any of the nodes enters a bus-off state. The bus-off state is caused by exceeding the allowable transmit or receive error counts. Each network is constructed with very inexpensive 120-ohm impedance twisted-pair AWG 24 cable with grounded shielding and 120 ohm terminating resistors on either end.

The first network consists of the three active nodes with nodes A and B separated by 13-cm cables and nodes C and B separated by 10 m of cable, as shown in Figure 3. The second example in Figure 5 is a network with 27 dummy load-nodes added between node C and the termination. The cable is increased between nodes B and C to 40 meters. The dummy nodes are powered transceivers without CAN controllers or processors and serve only to load the bus as if other nodes are

present. All dummy load-nodes are mounted on a bank of test boards for ease of wiring.

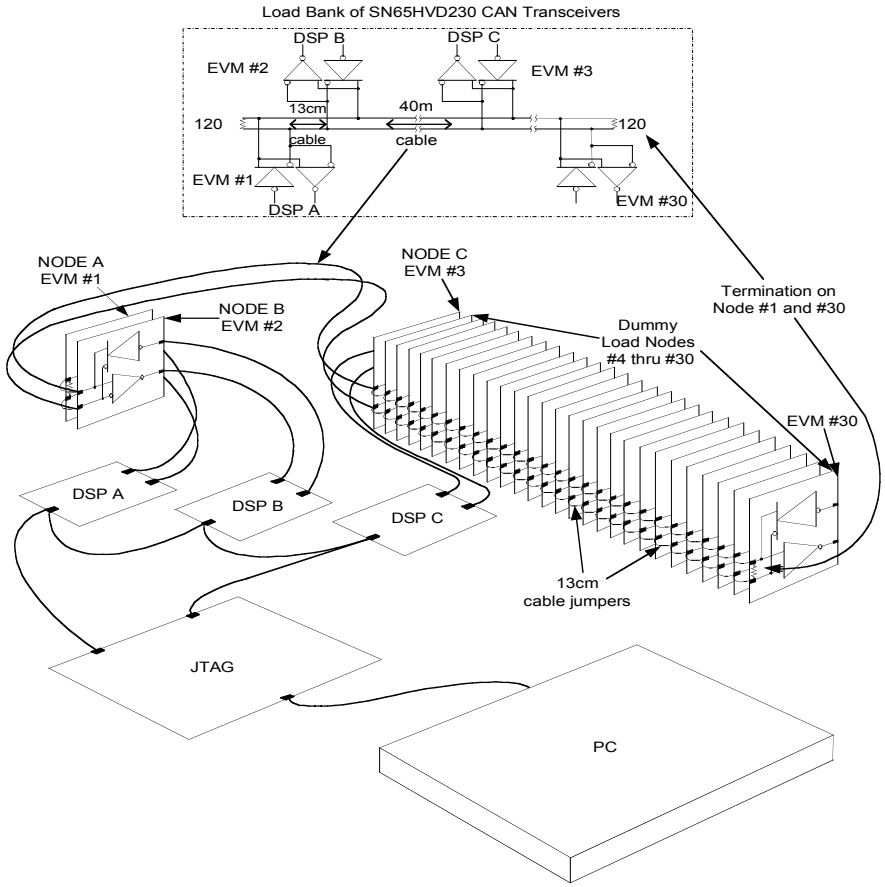

Figure 5. The 30-Node CAN Bus

The third network consists of 60 nodes, as shown in Figure 6. One bank of 30 Texas Instruments SN65HVD230 CAN transceivers and one bank of 30 Philips PCA82C250 transceivers are separated by a 200-m cable. Each bank has transceivers mounted on test boards separated by 13 cm of cable. Nodes A and B are nearest the 200-m cable on the ‘230 bank, while node C is nearest the 200-m cable on the ‘250 bank.

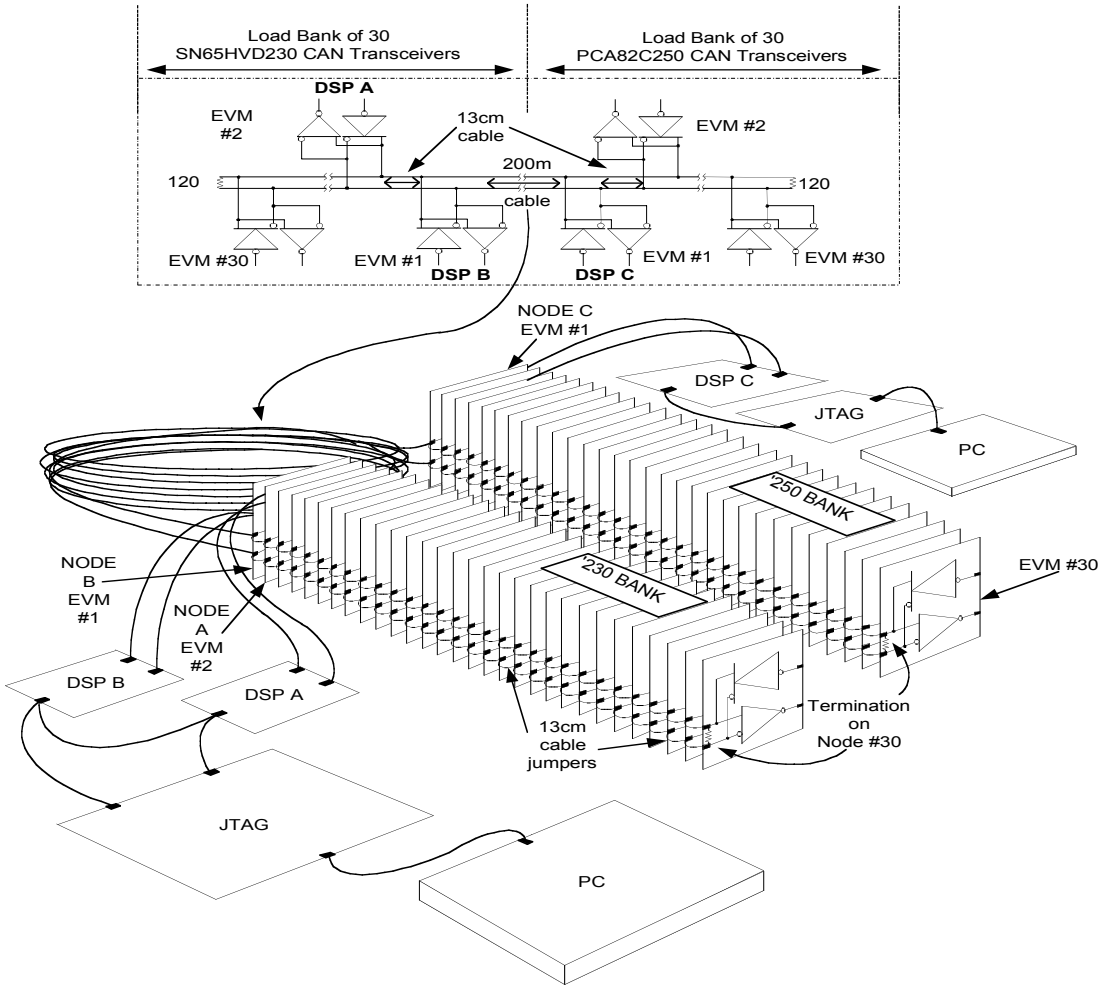

Figure 6. 60-Node CAN Bus

RESULTS

Analysis of Experimental Network Systems

The first example network (3 nodes) is tested at 500 kbps, 625 kbps, 800 kbps, 1 Mbps, 1.25 Mbps, and 2 Mbps. It performs flawlessly at 500 kbps and 625 kbps, but continually fails arbitration at signaling rates of 800 kbps, 1 Mbps, and 1.250 Mbps, and is inoperable at 2 Mbps due to a form error. The test is performed again by checking that each message is received without checking the receive order. Now the network functions without error at signaling rates of 800 kbps, 1 Mbps, and 1.250 Mbps. Messages from nodes B and C are being received, but in the wrong order – an arbitration compliance failure.

The 30-node network is tested with signaling rates of 250 kbps, 500 kbps, 625 kbps, 800 kbps, and 1 Mbps. It performs flawlessly at 250, 500, and 625 kbps, but fails arbitration at 800 kbps. At 1 Mbps, the network fails arbitration after encountering bit errors, stuff-bit errors, and cyclic- redundancy- check (CRC) errors. The network is re-tested by checking only that each message is being received without checking the receive order. This results in successful operation at 800 kbps, but messages from nodes B and C are being received in the wrong order. The network remained inoperative at 1Mbps after encountering form errors.

The 60-node network is tested with signaling rates of 31.25 kbps, 62.5 kbps, 125 kbps, and 250 kbps. It performs flawlessly at 31.25 kbps

and 62.5 kbps, but fails arbitration at 125 kbps having encountered form errors. It is inoperable at 250 kbps, encountering stuff bit errors. Again, the network is re-tested by checking only that each message was being received without checking the receive order, and the network remained inoperable at signaling rates of 125 kbps and above due to stuff bit errors.

Figure 7 displays the experiment without checking for receive order to better understand the failure mechanism of arbitration. This shows node B transmitting (CH3) its lower priority message before node C (CH4) transmits its higher priority message without node C ever having competed for bus dominance. For proper arbitration, it was required that both node B and C would attempt to transmit following the message from node A (CH2).

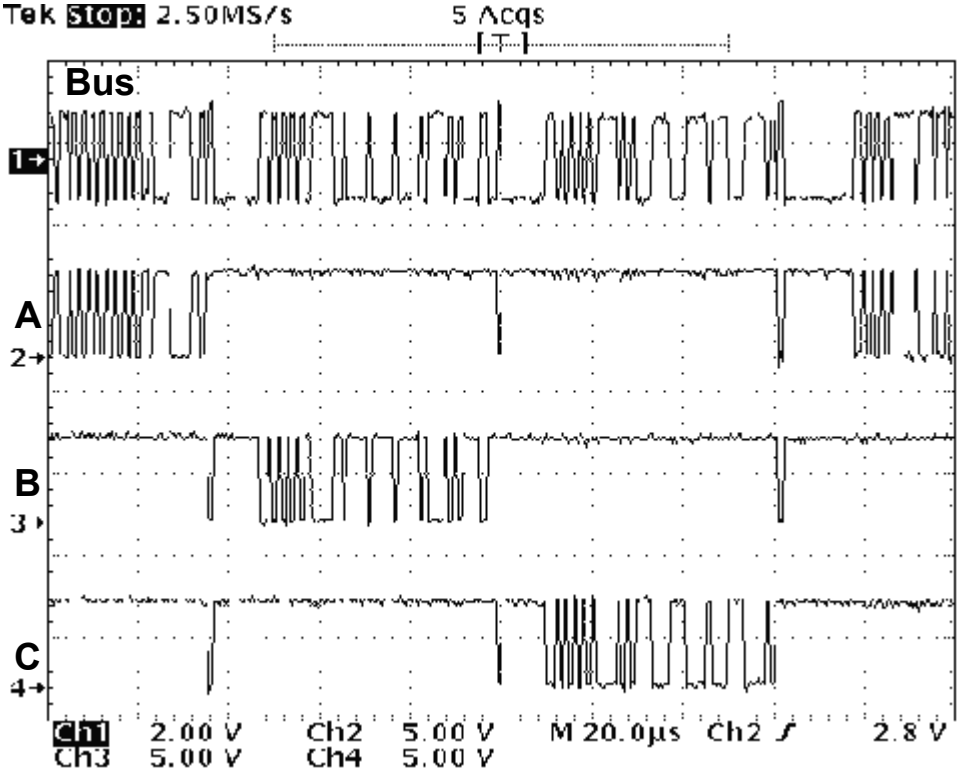

Figure 7. CAN bus signal at 1 Mbps with 11-bit identifiers and 2-byte data fields.

Note that node C is transmitting a valid ACK bit synchronously with node B in response to node A's message, but is unable to participate in bus arbitration properly. In each of these experiments, programs for nodes B and C are identical except for the message identifiers and data fields. If a dominant bit is received on the last bit of the intermission field in the interframe space between messages, it should be interpreted as the start of frame bit. Any other node also intending to transmit a start of frame bit immediately following intermission should begin transmitting the first bit of its identifier during the next bit so proper arbitration may commence. In other words, if one node jumps the gun on the last bit of the interframe space, all other nodes should accept this as a valid start of frame, synchronize, and transmit or receive as normal. Also, any high-priority message is delayed until the end of a lower priority message if the lower priority message began transmitting first. Therefore, node C must have attempted to transmit more than a full bit length after the dominant start of frame bit from node B, even if node B began to transmit a bit too early.

Node C's inability to negotiate arbitration properly and consistently, whether from bit error during message identifier or delayed message transmission, represents an experimental condition that exceeds the network's ability to be completely CAN protocol compliant. This condition can be remedied by slowing the signaling rate, since the network is exceeding its maximum rate.

Summary of Experimental Systems Performance

The final assessment of each experimental network reflects the highest bit rate at which the network maintains full CAN compliance with proper arbitration. The first network of 3 nodes and 10-m bus is fully compliant at 625 kbps. The second network with 30 nodes and 40-m bus is also fully compliant at 625 kbps while the third network with 60 nodes and 200-m bus is fully compliant at 125bps.

Conclusion: Evaluation of Arbitration as a Quantitative Benchmark

Clearly, testing for proper arbitration is a more stringent test than merely watching for bus errors in normal operation and suggests that an application can appear to be quite functional although it is unable to support proper arbitration. Arbitration problems are invisible to a user if no operational errors are encountered in the application. Both the first and second experimental networks operate without bus errors at much higher signaling rates than each network could support with proper arbitration.

Proper arbitration is critical to CAN performance because this is the mechanism that guarantees message collisions do not decrease bandwidth with multiple retransmission or loose messages. The quantitative benchmark produced by this method of testing a network for proper arbitration compliance is therefore defined as the maximum bus speed attainable on a safety critical network with the full data security enabled by CAN protocol.

This method is compatible with any network topology. Special care should be taken in the application of this method when selecting the location of nodes to monitor (A, B, or C) to ensure that the C monitor node is the worst-case node. Node C should be positioned to maximize the propagation delay, signal reflections, and other network conditions of the final application. This assures the greatest difference between the response of nodes B and C to node A. Therefore, networks with completely different topologies may be compared quantitatively with this arbitration benchmark.

This method is also adaptable for use during normal network operation, and offers the ability to check protocol compliance and provides confirmation that maximum data security is being enforced in the application while the network continues normal operation. If even more security is required for safety-critical systems, Time-Triggered CAN (TTCAN) has been developed by Bosch to addresses concerns about low-priority messages occupying a bus when a very high-priority message needs to be sent.

References

ISO-11898 Can Specification 2.0

Controller Area Network, Basics Protocols, Chips and Applications; Dr. Konrad Etschberger; ISBN 3-00-007376-0

CAN Systems Engineering, From Theory to Practical Applications; Wolfhard Lawrenz, ISBN 0-387-94939-9

Sam Broyles
Texas Instruments, Inc.
P.O. BOX 660199, MS 8701
Dallas, Texas 75266
sam.broyles@ti.com
phone # 214 480 3232
fax # 703 940 8113
http://www.ti.com

Steve Corrigan
Texas Instruments, Inc.
P.O. BOX 660199, MS 8710
Dallas, Texas 75266
phone # 214 480 4743
fax # 214 480 3160
s-corrigan1@ti.com
http://www.ti.com