"Enhancing Automotive and Industrial Control Design Performance Using CAN"

by Keith Pazul, Product Marketing Manager
Memory & Specialty Products Division
Microchip Technology Inc.
2355 West Chandler Blvd.
Chandler, AZ 85224
(480) 786-7200

Requirements for complex electro-mechanical systems like automobiles, industrial machinery, semiconductor manufacturing equipment, medical equipment and appliances are driving the need for more intelligent, fault-tolerant and reliable designs. To accomplish this, subsystems are being networked together in an effort to share information, reduce wiring between subsystems, and improve overall cost and reliability. Controller Area Network, or CAN as it is commonly referred to, is gaining market acceptance as the serial communication protocol-of-choice for many of these types of applications.

The reasons for this market acceptance are numerous. First, CAN is a Carrier Sense Multiple Access protocol with built-in Collision Detection (CSMA/CD). This allows all devices on the network to have an equal chance to gain control of the bus to send messages. If two messages try to transmit at the same time, collisions occur and they will be resolved without being destroyed by the collision. Second, messages are not sent to other nodes on the network via addresses. This enables different message broadcast types and message configurations to be handled without changing the message format. Thirdly, CAN is a fast, robust protocol with complex error detection and recovery capability.

CSMA/CD

CSMA stands for Carrier Sense Multiple Access. What this means is that every node on the network must monitor the bus for a period of no activity before trying to send a message on the bus (Carrier Sense). Also, once this period of no activity occurs, every node on the bus has an equal opportunity to transmit a message (Multiple Access). The CD stands for Collision Detection. If two nodes on the network start transmitting at the same time, the nodes will detect the 'collision' and take the appropriate action. The message with higher priority will continue and the one with the lower priority will stop transmitting. In CAN protocol, a non-destructive bitwise arbitration method is utilized. This means that messages remain intact after arbitration is completed even if collisions are detected. All of this arbitration takes place without corruption or delay of the higher priority message.

There are a couple of things that are required to support non-destructive bitwise arbitration. First, logic states need to be defined as dominant or recessive. Second, the transmitting node must monitor the state of the bus to see if the logic state it is trying to send actually appears on the bus. CAN defines a logic bit 0 as a dominant bit and a logic bit 1 as a recessive bit.

A dominant bit state will always win arbitration over a recessive bit state, therefore the lower the value in the Message Identifier (the field used in the message arbitration process), the higher the priority of the message. As an example, suppose two nodes are trying to transmit a message at the same time. Each node will monitor the bus to make sure the bit that it is trying to send actually appears on the bus. The lower priority message will at some point try to send a recessive bit and the monitored state on the bus will be a dominant. At that point this node loses arbitration and immediately stops transmitting. The higher priority message will continue until completion and the node that lost arbitration will wait for the next period of no activity on the bus and try to transmit its message again.

Message-Based Communication

CAN protocol is a message-based protocol, not an address based protocol. This means that messages are not transmitted from one node to another node based on addresses. Embedded in the CAN message itself is the priority and the contents of the data being transmitted. All nodes in the system receive every message transmitted on the bus (and will acknowledge if the message was properly received). It is up to each node in the system to decide whether the message received should be immediately discarded or kept to be processed. A single message can be destined for one particular node to receive, or many nodes based on the way the network and system are designed.

For example, an automotive airbag sensor can be connected via CAN to a safety system router node only due to the importance of this information in the system definition. Then this router node is connected in a daisy-chain configuration to the rest of the safety system nodes. The router node takes airbag information and routes it to all the other daisy-chained nodes on the safety system network. The safety system network nodes will receive the latest airbag sensor information from the router at the same time, acknowledge if the message was received properly, and decide whether to utilize this information or discard it.

Another useful feature built into the CAN protocol is the ability for a node to request information from other nodes. This is called a Remote Transmit Request (RTR). This is different from the example in the previous paragraph because instead of waiting for information to be sent by a particular node, this node specifically requests data to be sent to it.

For example, a safety system in a car gets frequent updates from critical sensors like the airbags, but it may not receive frequent updates from other sensors like the oil pressure sensor or the low battery sensor to make sure they are functioning properly. Periodically,

the safety system can request data from these other sensors and perform a thorough safety system check. The system designer can utilize this feature to minimize network traffic while still maintaining the integrity of the network.

One additional benefit of this message-based protocol is that new nodes can be added to the system without the necessity to reprogram all other nodes to recognize this addition. This new node will start receiving messages from the network and, based on the message ID, decide whether to process or discard the received information.

CAN protocol defines four different types of messages (or Frames) for communication on the bus. The first and most common type of frame is a Data Frame. This is used when a node transmits information to any or all other nodes in the system. Second is a Remote Frame, which is basically a Data Frame with an RTR bit set to signify it is a Remote Transmit Request. A Remote Frame is used by a node to request information to be sent to it by another node or nodes. The other two frame types are for handling errors. One is called an Error Frame and one is called an Overload Frame. Error Frames are generated by nodes that detect any one of the many protocol errors defined by CAN. Overload errors are generated by nodes that require more time to process messages already received.

Fast, Robust Network Communication

Because CAN was initially designed for use in automobiles, a protocol that efficiently handled errors was critical if it was to gain market acceptance. With the release of version 2.0B of the CAN specification, the maximum communication rate was increased 8x over the version 1.0 specification to 1Mbit/sec. At this rate, even the most time-critical parameters can be transmitted serially without latency concerns. In addition to this, the CAN protocol has a comprehensive list of errors it can detect that ensures the integrity of messages.

CAN nodes have the ability to determine many different fault conditions and act on them accordingly: Transmission Errors are detected via a Cyclic Reductancy Check (CRC) value; General receipt of message errors are detected via Acknowledgement Errors; CAN message format errors are detected via Form Errors; CAN bus signal errors are detected via Bit Errors; and Synchronization and timing errors are detected via Stuff Errors. Each CAN node keeps track of errors in error counters. These error counters determine if a node needs to transition to a degraded mode based on the severity of problems being encountered. CAN nodes can transition from functioning normally (being able to transmit and receive messages and Error Frames), to a passive mode where a node can only control the bus if not already being utilized, to shutting down completely (bus-off) based on the severity and quantity of the errors detected. CAN nodes also have the ability to detect short disturbances from permanent failures and modify their functionality accordingly. This feature is called Fault Confinement. No faulty CAN node or nodes will be able to monopolize all of the bandwidth on the network because faults will be confined to the faulty nodes and these faulty nodes will shut off before bringing the network down. This is very powerful because Fault Confinement guarantees bandwidth for critical system information.

Conclusion

The CAN protocol was optimized for systems that need to transmit and receive relatively small amounts of information (as compared to other protocols like Ethernet or USB, which are designed to move much larger blocks of data) reliably to any or all other nodes on the network. It has good bus-sharing and arbitration properties, unique message transmission properties, enables fast, robust communication of messages and contains fault-confinement properties that make attractive for even latency-critical applications. Semiconductor suppliers are also realizing that CAN is gaining market acceptance. Suppliers who already offer products supporting CAN are broadening their product portfolios. Also, new suppliers are entering the market by offering innovative products supporting CAN. All of these benefits are making it more attractive for system designers to utilize CAN for implementing networking technology into their designs.

####

SIDEBAR ARTICLE:

"Integrate Immediate CAN Capability into Your Existing Design"

An innovative new interface controller suppors the Controller Area Network (CAN) Specification 2.0B and features an industry standard SPI™ serial interface, enabling an easy connection to virtually any microcontroller-based system for immediate CAN capability.

With the low-cost MCP2510 18-pin stand-alone CAN controller, system designers can select from a much broader range of microcontrollers for their designs because the CAN peripheral is not required to be integrated on the device. With minor software modifications for sending data to and from the MCP2510 SPI port, CAN communication can be added to existing systems through the simple SPI interface without requiring an update to the microcontroller. This enables faster design update cycles and time to market.

The MCP2510 contains on-board features, such as interrupt capability, message masking and filtering, message prioritization, multi-purpose I/O pins, and multiple transmit/receive buffers which significantly offload the microcontroller overhead required to handle CAN message traffic. This allows simpler, lower-cost embedded solutions to be created by using the MCP2510 with a simple microcontroller. Until now, CAN support was limited to high pin count integrated solutions.

####