Ease into the Flexible CANbus Network

By Bonnie C. Baker, Microchip Technology Inc.

ANALOG DESIGN NOTE ADN004

CANbus networks have been around for over 15 years. Initially this bus was targeted at automotive applications, requiring predictable, error-free communications. Recent falling prices of CAN (Controller Area Network) system technologies have made it a commodity item. The CANbus network has expanded past automotive applications. It is now migrating into systems like industrial networks, medical equipment, railway signaling and controlling building services (to name a few). These applications are utilizing the CANbus network, not only because of the lower cost, but because the communication that is achieved through this network is robust, at a bit rate of up to 1 Mbits/sec.

A CANbus network features a multi-master system that broadcasts transmissions to all of the nodes in the system. In this type of network, each node filters out unwanted messages. A classical client/server network (such as Ethernet) relies on network addressing to deliver data to a single node. If multiple nodes exist in this network, a star configuration implements a centralized control (**Figure 1**). Fewer microcontrollers are needed to perform the varied tasks, but the MCUs are usually more complex with higher pin counts.

In contrast, every node in a CAN system receives the same data at the same time. By default, CAN is message-based, not address-based. Multiple nodes are integrated in the system using a distributed control implementation (**Figure 1**). One of the advantages of this topology is that nodes can easily be added or removed with minimal software impact. The CAN network requires intelligence on each node, but the level of intelligence can be tailored to the task at that node. Consequently, these individual controllers are usually simpler, with lower pin counts. The CAN network also has higher reliability by using distributed intelligence and fewer wires.

Ethernet differs from CAN in that Ethernet uses collision detection at the end of the transmission. At the beginning of the transmission, CAN uses collision detection with resolution. When a collision occurs during arbitration between two or more CAN nodes that transmit at the same time, the node(s) with the lower priority message(s) will detect the collision. The lower priority node(s) will then switch to receiver mode and wait for the next bus idle to attempt transmission again.

The winning transmitter will continue to send its message as if nothing happened. Response time to collision resolution is faster because the correction occurs at the beginning of the transmission during arbitration of a message and the high priority message is not destroyed.

The CANbus network specification, written by Bosch, has been standardized by ISO and SAE. The entire CAN specification is standardized in ISO 11898-1. ISO 11898-2 contains the CAN physical layer specification. The CAN specification is not completely standardized in the SAE specification.

CANbus communication is achieved using message frames. The three types of frames are data, remote and error. Each frame has internal fields that define the type of frame that is being sent and then provides the pertinent information. For instance, a data frame is constructed with 6 fields: arbitration, control, data, CRC (Cyclic Redundancy Check), acknowledge and end-of-frame. During transmission, the arbitration field is used by every node on the network to identify and/or resolve collisions. The arbitration field is also used to identify the message type and destination. The control frame defines the data frame length. The data frame contains data and has the specified number of bytes per the control frame. The CRC frame is used to check for data errors. And finally, every transmission requires an acknowledge frame from all of the receivers on the CAN network.

In the CAN network multi-master environment, nodes can be added or removed without significant consequence to the operation and reliability of the system. An example of a single node for a CAN network is shown in **Figure 2**. In this diagram, pressure is measured using a Motorola® pressure sensor, MPX2100AP. The differential output voltage of this sensor is gained by a discrete instrumentation amplifier and filtered by a fourth order, low pass, active filter. The signal is then converted to a digital code with a 12-bit A/D converter, **MCP3201**. The receiving microcontroller sends the data to the CAN controller. The common language between the nodes is generated and maintained by the CAN controller

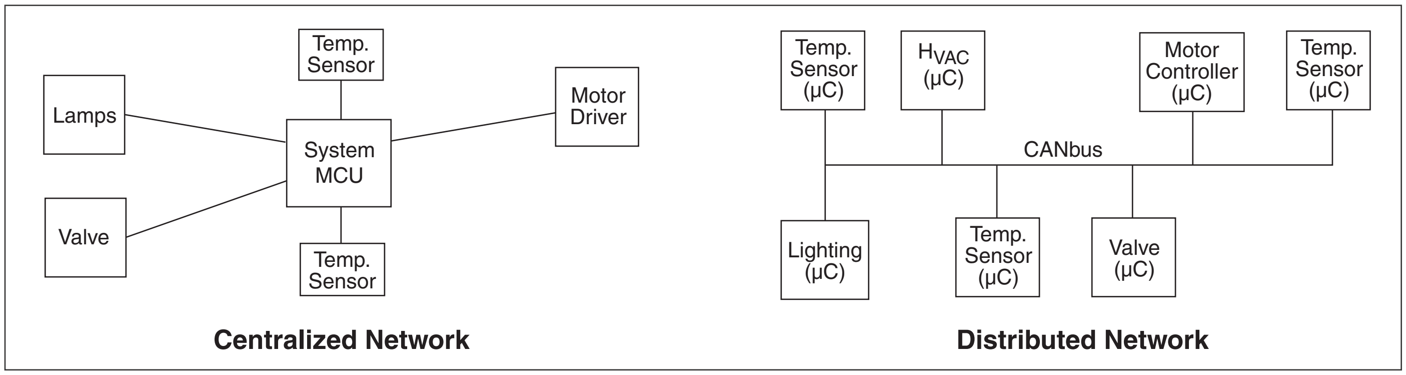

Figure 1. *For multi-task networks, a Centralized Network is usually used for Ethernet systems. If a node is added to this system, the system MCU could require significant modifications. With CAN networks, the Distributed Network is implemented. A node can easily be added or taken out of the system with minimal firmware changes.*

and the voltage compliance to the network is managed by the CAN driver.

Each node in a CAN network can perform a unique function. Although **Figure 2** illustrates a pressure-sensing system, other types of systems can complement your application. Additionally, this block diagram of a CAN node can be implemented in a variety of ways. For instance, in the initial build, the microcontroller could have the CAN controller integrated on-chip. At a later date, nodes can easily be added with minimal software impact. When you are ready to add, enhance or build a small stand-alone network, the combination of an **MCP2515** with a simple microcontroller would be a good choice.

The MCP2515 stand-alone CAN controller implements version 2.0B of the CAN specification. It is capable of transmitting and receiving both standard and extended data and remote frames. The MCP2515 has two acceptance masks and six acceptance filters that are used to remove unwanted messages. The 4-wire interface between the MCP2515 and the controller is SPI™. The MCU pins used for SPI can be recovered if the MCP2515 RXnBF pins are configured as GP output and the TXnRTS pins are configured as GP input.

The MCP2515 has three main blocks:

1. The CAN module, which includes the CAN protocol engine, masks, filters, transmits and receives buffers
2. The control logic and registers that are used to configure the device and its operation
3. The SPI protocol block

Typically, each node in a CAN system must have a device to convert the digital signals generated by a CAN controller, to signals suitable for transmission over the bus cabling. The device also provides a buffer between the CAN controller and the high-voltage spikes that can be generated on the CANbus by outside sources (EMI, ESD, electrical transients, etc.). The MCP2551 high-speed CAN, fault-tolerant device provides the interface between a CAN protocol controller and the physical bus. The MCP2551 has differential transmit and receive capability for the CAN protocol controller and is fully compatible with the ISO-11898 standard, including 24V requirements. It will also operate at speeds of up to 1 Mbits/sec.

This serial communications protocol supports distributed real-time control with a sophisticated level of security. The CANbus time-proven performance ensures predictable error-free communications for safety-conscious application environments. It is able, through arbitration, to prioritize messages. The configuration is flexible at the hardware, as well as the data link layer, where many of the transmission details can be modified by the designer. This is done, while at the same time there is system- wide data consistency.

Recommended References:

AN212: *"Smart Sensor CAN Node Using the MCP2510 and PIC16F876*", Stanczyk, Mike, Diversified Engineering, Inc.

AN228: *"A Physical Layer Discussion"*, Richards, Pat, Microchip Technology Inc.

AN754: *"Understanding Microchip's CAN Module Bit Timing"*, Richards, Pat, Microchip Technology Inc.

"High-Speed CAN Transceiver", Microchip MCP2551 product data sheet, DS21667

"Stand-Alone CAN Controller with SPI™ Interface", Microchip MCP2515 product data sheet, DS21801

"Wireless CAN Yard Lamp Control", Dammeyer, John, Circuit Cellar, August 2003, page 12

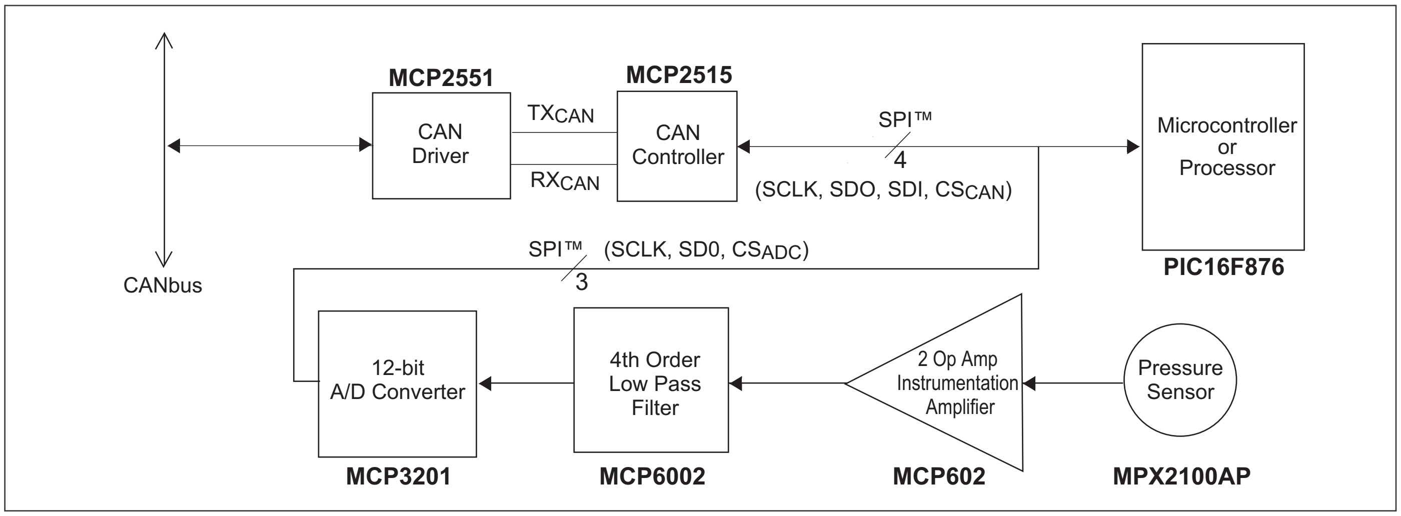

Figure 2. *This is an example of a single node for a CAN network. All of the elements for appropriate communication on the network are implemented through the CAN driver (MCP2551), CAN Controller (MCP2515) and the microcontroller.*

For more information, please visit www.microchip.com