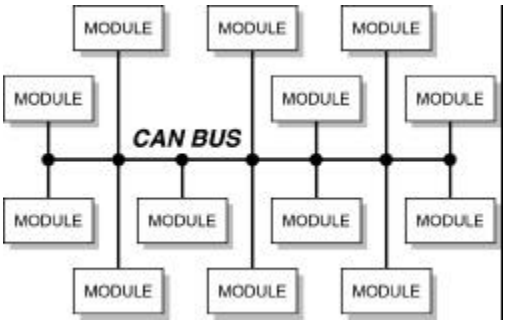
Introduction to CAN



What is CAN?

Controller Area Network (CAN) is a common, small area network solution that supports distributed product and distributed system architectures. The CAN bus is used to interconnect a network of electronic nodes or modules.

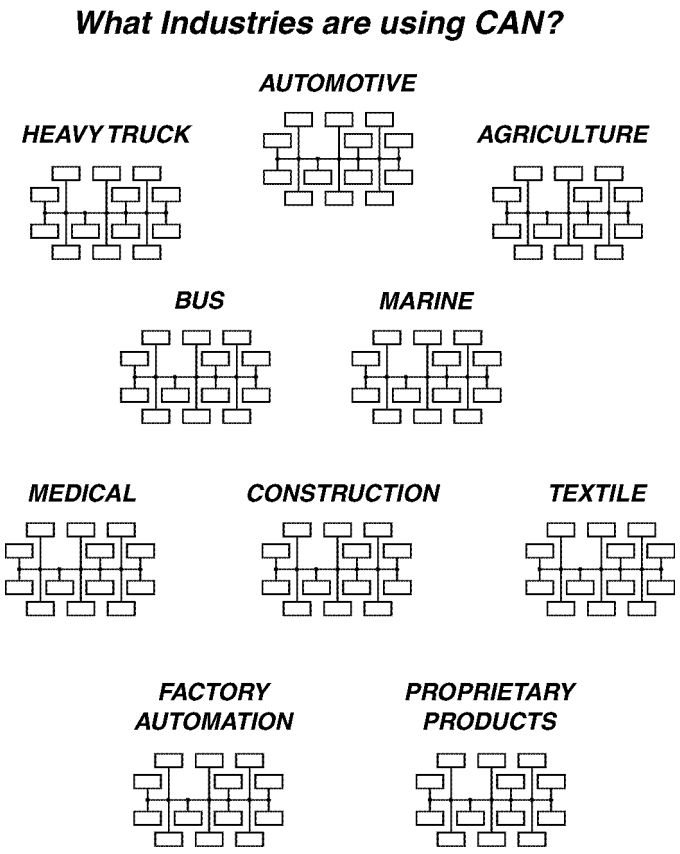
Typically, a two wire, twisted pair cable is used for the network interconnection.

The CAN protocol is a set of stringent rules, implemented in silicon, that supports the serial transfer of information between two or more nodes.

CAN is implemented by a large number of industries including automotive, truck, bus, agriculture, marine, construction, medical, factory automation, textile, and many others.

CAN is used as the basis for several major "7-layer" protocol developments such as J1939, CANopen, SDS, DeviceNet, and NMEA-2000. Each of these large protocol architectures are essentially complete industry-specific network solutions packaged to include defined requirements for the physical layer, address structure, message structure, conversation structure, data structure, and application/network interface. Pre-packaged "7-layer" protocols provide high value for vertically integrated industries like heavy truck, marine, or factory automation.

On the other end of the spectrum, many other companies choose to develop a proprietary distributed product strategy. For both business and technical reasons, these companies internally create a customized "7-layer" CAN-based protocol that is optimized to satisfy their own application-



Is CAN a Standard?

Although CAN was originally developed in Europe by Bosch several years ago primarily for automotive applications, the protocol has gained wide acceptance and has become an open, international standard.

As a result, the Bosch CAN 2.0B specification has become the de facto standard that new CAN chip designs follow. While some previous CAN chips are becoming obsolete, many semiconductor companies are rapidly developing new CAN controllers.

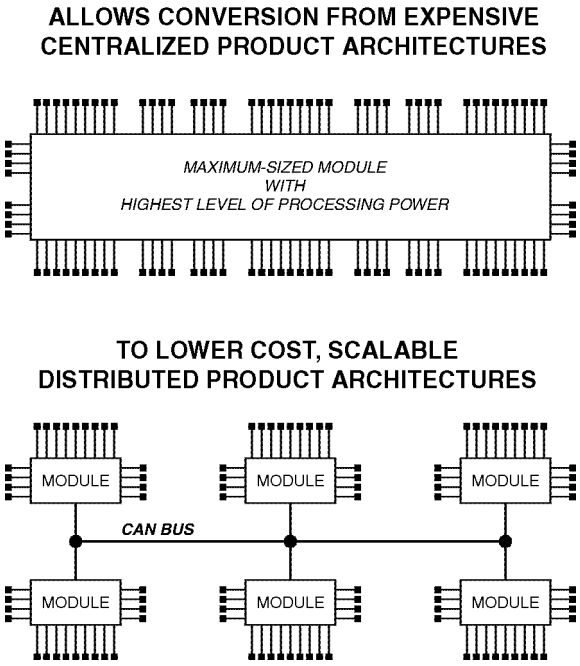
CAN is an Open International Standard

Since Bosch has fixed the definition of CAN, the protocol is essentially stable. Bosch has demonstrated no intention to make any new changes and no longer has technical staff supporting the protocol. This is one of the primary business reasons why CAN has become a standard.

What is the key business and technical reason to use CAN?

CAN is the leading type of small area network for several business reasons.

Why use CAN?



Key Reasons to Use CAN

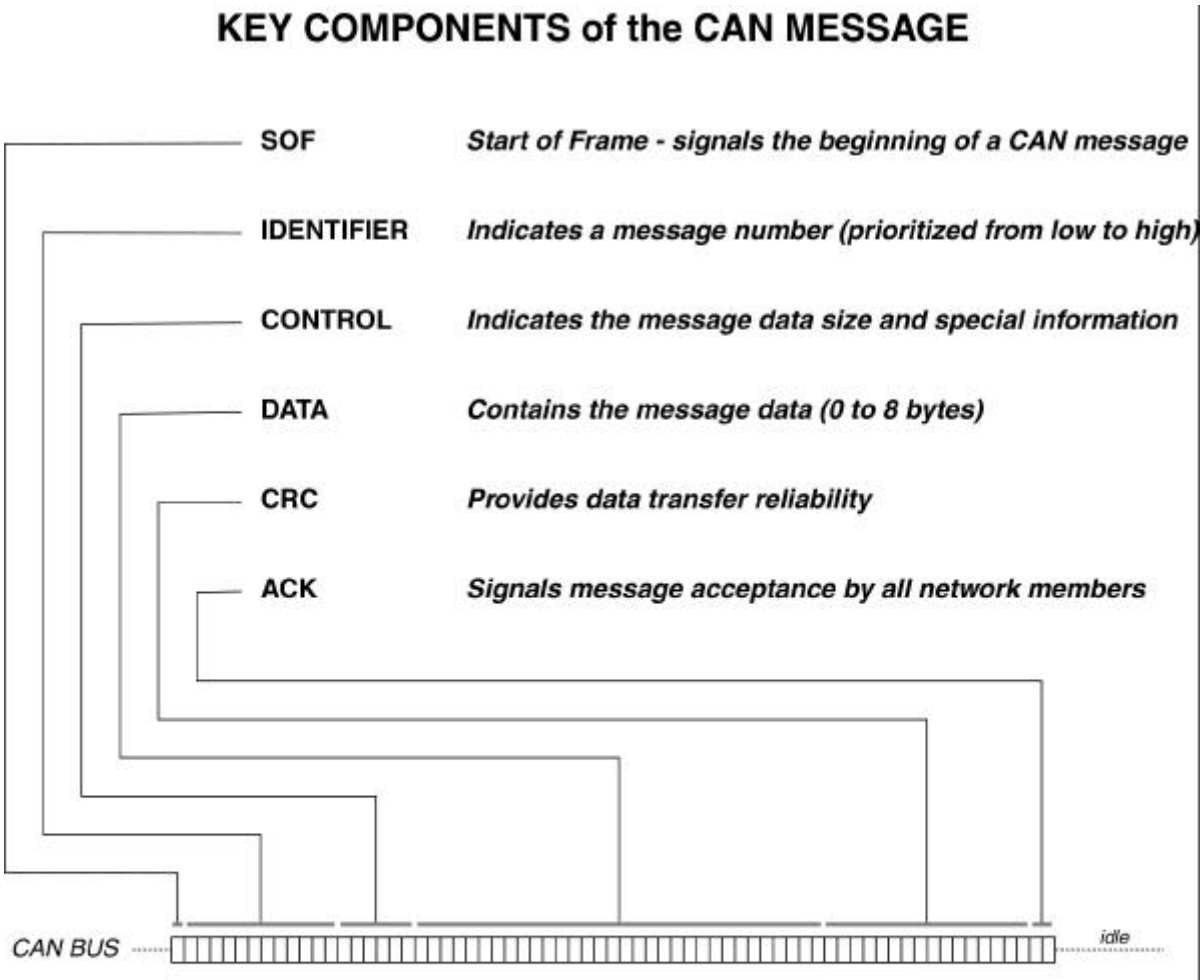
- ✓ Low Connect Cost
- ✓ Low Cost Components
- ✓ Growing Number of CAN Chips
- ✓ Increasing Knowledge Base
- ✓ Increasing Integration Service Base
- ✓ Wide Variety of CAN-based Products
- ✓ Wide Variety of Off-the-Shelf Tools Available
- ✓ Potential Lower Wiring Cost

Basically, how does CAN operate?

CAN is a serial communication protocol that may be used to transfer up to 8 data bytes within a single message. For larger amounts of data, multiple messages are commonly used. Most CAN-based networks select a single bit rate. While communication bit rates may be as high as 1 M BPS, most implementations are 500K BPS or less.

CAN supports data transfers between multiple peers. No master controller is needed to supervise the network conversation.

The CAN message is bit-oriented, always begins with a "start of message" indication, includes an address (called identifier), may contain data, includes a CRC, requires an acknowledgement from all network members, and is converted to an appropriate signal by the selected physical layer before being placed on the shared media (typically wires).



CAN provides a simple mechanism, called **bitwise arbitration**, to eliminate competing transmitters from colliding into each other during message initiation. This feature provides an added measure of transmission efficiency by using “dominant-recessive logic” (an appropriate and welcomed re-naming of open collector or open emitter logic).

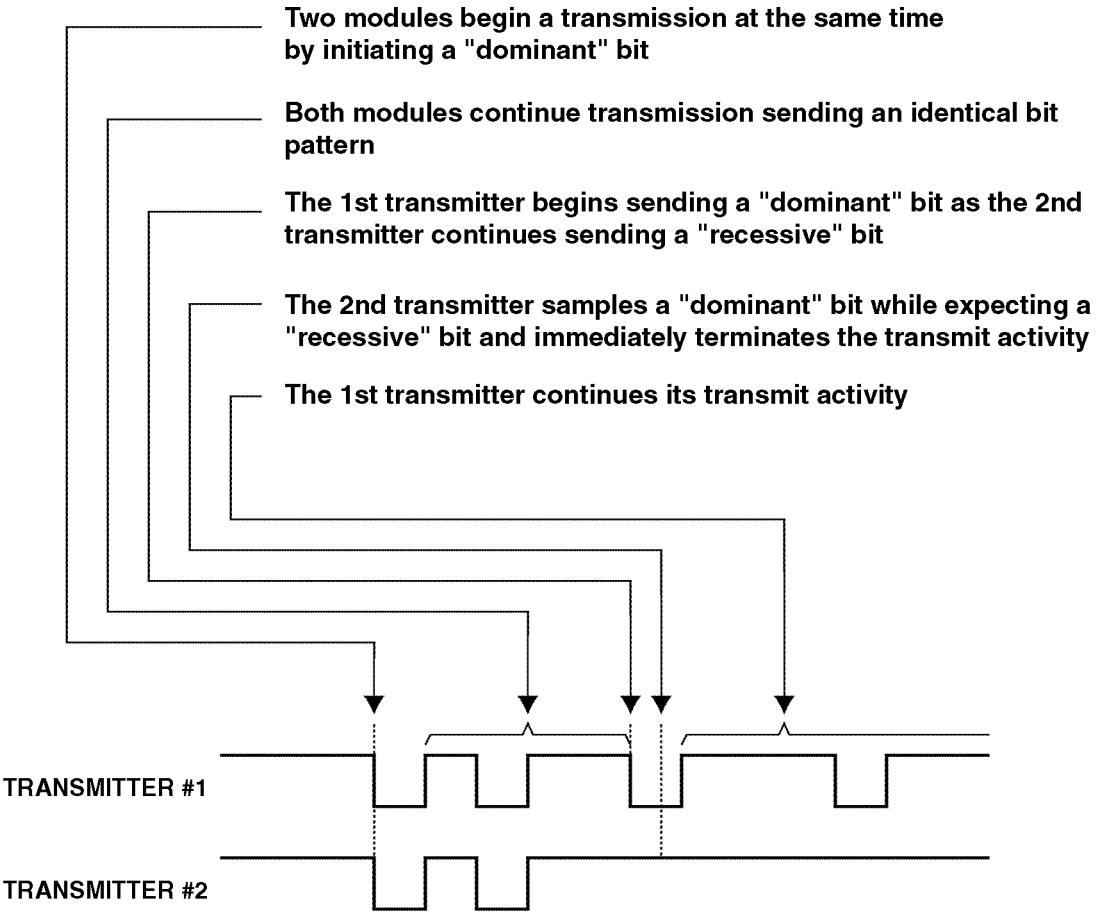
BITWISE ARBITRATION RULES

1. A "dominant" bit always wins over a "recessive" bit (similar to open collector logic)

2. A transmitter always compares its transmit bit with its receive bit.

3. If a transmitter detects that the receive bit is not the same as the transmit bit, then the transmitter will immediately stop its transmission. (with some dependencies, a subsequent re-transmission may occur)

EXAMPLE OF BITWISE ARBITRATION



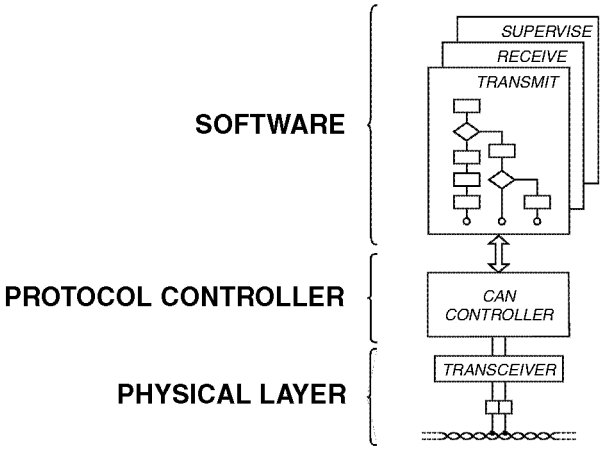
NOTES

1. *As shown a "dominant" bit is LOW and a "recessive" bit is HIGH; this form of Dominant/Recessive Logic is identical to the operation of open collector logic.*
2. *Several CAN interface circuits use a combination of both open collector and open emitter logic to implement a differential transceiver.*

What is needed to implement CAN?

CAN IMPLEMENTATION COMPONENTS

To implement CAN, three components are required - software, a CAN controller, and a physical layer



NOTES

1. The CAN software must process the maximum expected bus traffic while sharing internal resources with the intended application.

2. CAN controllers are available as either discrete stand-alone ICs or may be integrated as a peripheral on the same silicon as the microcontroller.

3. Many physical layer choices exist for CAN. Most medium to high speed implementations use two wires with twisted pair cabling. Fiber optic is also possible.

The **Physical Layer** encompasses the network interconnection, wiring, connectors, and transceiver chip.

The **Protocol Controller** is the chip or integrated controller that processes the CAN protocol as defined by the CAN 2.0B specification. It connects to the local microcontroller and the transceiver.

The **Software** interfaces to the CAN Controller; transfers messages into the controller for transmission, collects received messages, and processes network supervision tasks like network initialization and maintaining network relationships.

What are some of the common CAN physical layers?

While CAN does require a suitable physical layer, the CAN protocol is essentially independent of the type of physical layer implementation as long as the interface or transceiver supports dominant-recessive logic.

Several industry-specific physical layers have been created as standards. These customized CAN interfaces are tailored to the expected environment and, in some instances, may include cable shielding or additional power connections.

COMMON CAN PHYSICAL LAYERS

NAME	BIT RATE (BPS)	DESCRIPTION	TYPICAL TRANSCEIVER	APPLICATION AREA
High Speed	100K - 1M	2 wire, twisted pair	250-type	General high speed distributed functions
SAE J1939-11	250K	2 wire, twisted pair, with shield	250-type	Heavy Truck, Bus, Construction
SAE J1939-12	250K	2 wire, twisted pair, shielded with integrated 12V power	250-type	Agriculture
SAE J2284	500K	2 wire, twisted pair, unshielded	250-type	Automotive - High speed, motion control
SAE J2411	25K 40K	1 wire	SWC-type	Automotive - Low speed, human control
NMEA-2000	62.5K, 125K 250K, 500K 1M	2 wire, twisted pair, shielded with integrated power	250-type	Marine
DeviceNet	125K 250K 500K	2 wire, twisted pair, shielded with integrated 24V power	250-type	Factory Automation
CANopen	10K, 20K 50K, 125K 250K, 500K 800K, 1M	2 wire, twisted pair, optional shield, with optional power	250-type	Factory Automation
SDS	125K, 250K 500K, 1M	2 wire, twisted pair, shielded with optional power	250-type	Factory Automation
Fault Tolerant	<125K	2 wire, twisted pair, unshielded	252-type	Communicates with single wiring fault

NOTE

1. Because each listed physical layer is controlled by an industry-led committee, some information may be changed or outdated. Contact the appropriate group for specific details.

Learning More about CAN

> ***Learning about CAN down to the protocol's atomic level is not recommended.***
>
> ***A better approach is to focus on that particular aspect of CAN that is relevant to your area of expertise or involvement.***

The difference between being a distributed systems designer, or a module hardware developer, or a electrical system wiring expert, or a software programmer has everything to do with what portions of the CAN knowledge base you need to know. In most cases, there is a certain amount of information that you don't need to know.

For example, the difference between the CAN1.0 and CAN2.0 specifications is only of historical or academic significance. This and other low quality information should be avoided. Likewise obsolete CAN chips, comparisons to other dead small area network protocols, philosophy, and countless opinions abound to create additional obstacles.

For the Bosch CAN specification and additional CAN reference material - visit the Vector CANtech web site at **vector-cantech.com**