A Brief Technical Introduction to CAN

Version 1.1

2008-08-14

Application Note AN-AND-1-101

Author(s)	Bruce Emaus
Restrictions	Public Document
Abstract	This application note introduces the beginning developer to the CAN Protocol and focuses on several of the key technical concepts without getting into too much detail.

Table of Contents

1.0 Overview

The world of small and large microcomputers / microcontrollers has a third serial communication peripheral alongside the UART and SPI interfaces. It is called CAN. Its effect has been rather profound, especially in the area of low-cost embedded system design.

2.0 CAN

CAN, which stands for Controller Area Network, is a significant leap forward when compared to the use of the UART or SPI.

First, notice that the name says "network". Like the UART or SPI, CAN is a serial communication method. However, the CAN protocol is no ordinary substitute for the UART or SPI. CAN's robust silicon-based implementation provides sufficient power to manage a major portion of a complete communication network solution. Realizing equivalent network behavior when using the UART or SPI requires a significant amount of communication software to provide the same functionality.

As the largest and most common small area network solution, CAN supports distributed product and distributed embedded system architectures by providing a low-cost communication method to interconnect a network of electronic nodes, stations, or modules. CAN is part of the reason why distributed product architectures are rapidly replacing the old world of centralized architectures.

With the price of CAN silicon getting cheaper and cheaper every day, the days of our old friend the UART may be coming to a close.

Developed at Bosch during the mid '80s for automotive applications, the CAN protocol has emerged to be a worldwide standard, both in the automotive industry and in several other industries including agriculture, heavy truck, bus, construction equipment, marine, industrial automation, and medical.

Figure 1 – CAN is a Worldwide Standard

While a few earlier and obsolete CAN 1.2 chips still are available, the Bosch CAN 2.0B specification has become the de facto standard that all new CAN Controller designs follow. The definition of the CAN Protocol is fixed - the protocol is essentially stable.

In the auto industry, the CAN protocol has essentially replaced all earlier internally generated, proprietary protocols which were individually developed and supported by each car company. This is happening for other industries also. CAN has been adopted by many other companies that already use distributed product architectures as a replacement for the company's internally developed proprietary protocol, which in most cases is a UART-based protocol. The basic reason for this widespread standardization is quite simple -- all things considered, CAN is a lower-cost solution than any other serial communication method.

It is true that the cost of CAN silicon is perhaps ten times higher than the cost of UART silicon. But how much is the cost of the UART today? It is below one penny. At this point on the cost curves, the cost differential is no longer business relevant. However, when the accountants factor in the additional business cost of the internal supporting infrastructure necessary to take care of a proprietary protocol, the case for selecting CAN becomes much easier to justify. The wide availability of CAN Controllers, CAN transceivers, CAN software, CAN knowledge, CAN tools, and CAN engineering services is a low cost substitute compared to using internal resources.

From the business perspective, the key reasons to use CAN include -

- low-cost CAN hardware components – widely available
- low-cost CAN tools – in comparison to making tools from scratch
- low-cost CAN software components – in comparison to writing software
- low-cost CAN training – in comparison to creating training classes
- low-cost CAN protocol improvements – compared to upgrading the OEM’s proprietary protocol

While CAN has won primarily for business reasons, from the technical standpoint the CAN Protocol is also considered to be near-perfect. From the technical side, CAN is better than most UART-based proprietary protocols for many reasons, including:

- CAN operates at the message level – the UART functions at the byte level, and message handling must be implemented by additional communication software
- CAN has automatic error handling – since the UART essentially has no error handling, this activity must be implemented by additional software
- CAN is more reliable than the UART – For CAN, the academic community has reported an undetected error rate of one message every 1000 years – the undetected error rate of most UART-based protocols is relatively unknown.

2.1 Basic CAN – Unlimited Speed

CAN is a high-speed serial communication protocol. The transfer of a CAN message is bit-oriented, and always begins with a "start of message" signal or indication. The typical transfer includes an identifier, data, a CRC, and requires an acknowledgement from all network members.

For network-based applications, CAN is typically converted to an appropriate physical representation (electrical, optical, etc.) by the selected Physical Layer before being placed on the shared medium (typically wires).

Inadvertently and unfortunately, the CAN Protocol is artificially limited to 1 Mbps. The original design intention – wire-based automotive applications – artificially set the limit. This is certainly no longer the application boundary.

Based on the governing CAN specification, the protocol speed is actually limited only by the network propagation delay. For example, why should an inter-processor communication network confined to the area of a circuit board characterized by low propagation delay be limited to 1 Mbps?

Perhaps the speed limit should be erased from the CAN specification limit, allowing the speed of silicon, copper, or whatever materials are used to establish the true limit.

2.2 CAN Message Transfer – Unlimited Data Size

A UART- or SPI-based protocol is only capable of making single byte transfers. To transfer a message with multiple bytes, a software-based protocol must be constructed that handles the processes of message transmission, message reception, and message error handling. If the CAN Protocol is used instead, the CAN Controller will handle message transmission, message reception, and message error handling automatically – these message transfer processes are completely implemented in silicon.

A single CAN message allows a data transfer up to eight bytes in length. This means that each message can transfer 0, 1, 2, 3, 4, 5, 6, 7, or 8 bytes.

Well, what about transferring more than eight bytes? What happens when transferring files? Perhaps flash programming the module using the CAN bus is desirable. This is no problem. Just use some additional software to handle transfers larger than eight bytes. All the car companies do it. They use a special higher-level protocol called a Transport Protocol, which essentially handles a file transfer.

Each CAN message includes an identifier. Depending on the point of view – system, software, or hardware – the identifier essentially "names" the message or points at the data or acts like an address. For example, we could use identifier 100 to be the 'headlights on' message and identifier 101 to be the 'headlights off' message. Typically, a database of all message names and their corresponding identifiers is used to describe the network messages at the system level.

Each CAN message includes a CRC, also called the Cyclic Redundancy Check, to increase the reliability of the message transfer. The CRC serves the purpose of a checksum. If the receiver's CRC matches the transmitted CRC, then the likelihood of a transfer error is very low. The exact degree of CAN transfer reliability provided by the CRC is subject to debate.

While the overall structure of an actual CAN message has slightly more detail, Figure 2 shows the essential key components of a basic CAN message.

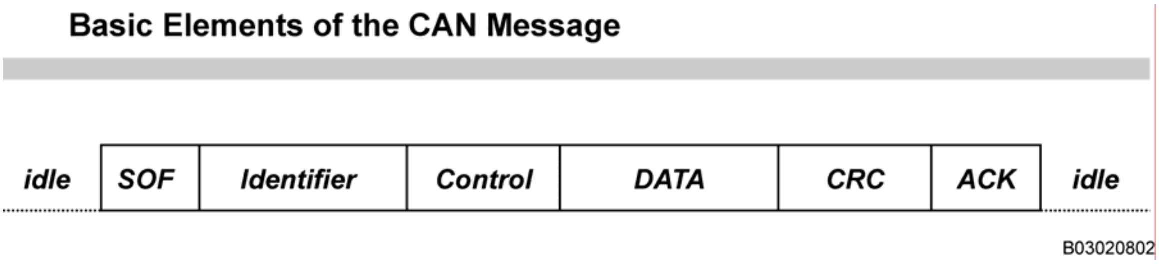

Figure 2 – The Basic CAN Message Structure

2.3 Basic CAN Acknowledgment – Guaranteed Message Consitency

Each transmitted CAN message includes an acknowledgment activity which occurs during the transfer of that message. CAN Protocol rules require that the entire network membership must participate in acknowledging a message. All message receivers must signal either a positive or negative acknowledgment ("error frame"), and this activity typically occurs during the final portion of the CAN message data transfer. A simplified model of the acknowledgment process at the system level is shown in Figure 3.

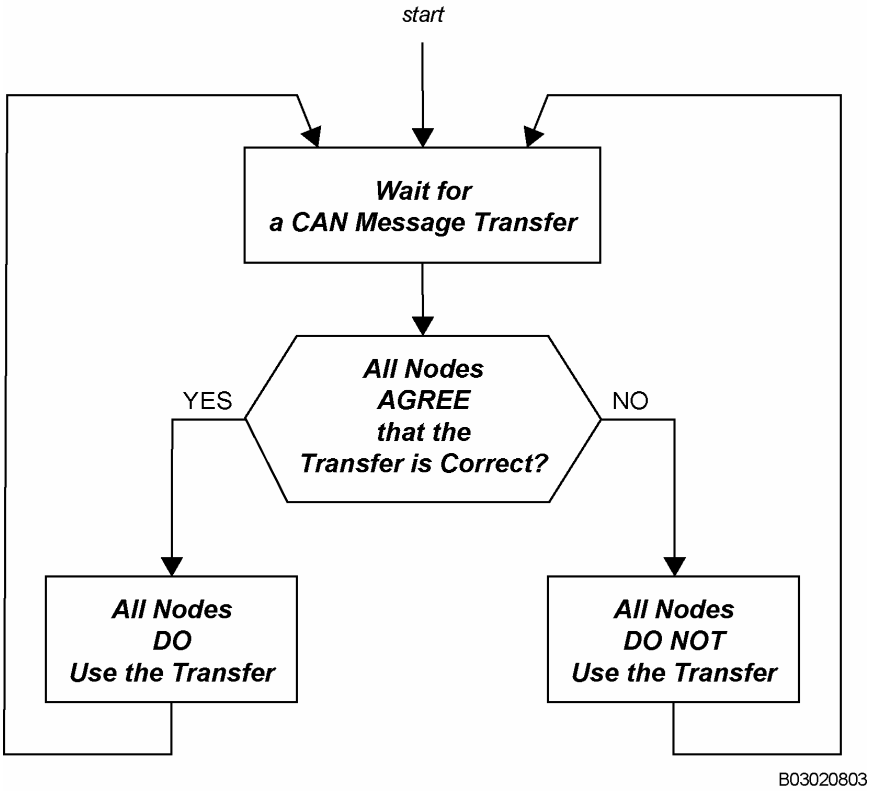
start
Wait for a CAN Message Transfer
All Nodes AGREE that the Transfer is Correct?
YES
NO
All Nodes DO Use the Transfer
All Nodes DO NOT Use the Transfer
B03020803

Figure 3 – All CAN Nodes Participate in the Acknowledgment Process

It is important to note that the acknowledgment process is entirely handled by the CAN Controller – no software is necessary.

While this acknowledgment process is somewhat hard to understand, it is one of the most important foundations of the CAN Protocol. The rule – that each and every CAN node must agree that each and every message transfer is correct – and confirm it with an indication – is a virtual guarantee that the entire network membership is using exactly the same data. This rule is extremely valuable for real-time distributed control systems. This "guaranteed data consistency" feature is perhaps one of the most significant technical reason for CAN's success.

The acknowledgment process solves the problem of what happens if there is noise on the communication wires that corrupts the message transfer. Each node that detects a message transfer error will not acknowledge that the message is correct and subsequently will signal a negative acknowledgment. This negative acknowledgment, transmitted as an "error frame", is detected by all network nodes – even by those nodes which earlier indicated a positive acknowledgment. Detection of the negative acknowledgment, or "no agreement", means that no receiving node may use the data. However, when the transmitter detects the negative acknowledgment it automatically tries to send the message again.

The CAN Protocol acknowledgment rule means:

All agree OR all do not agree – there is no in between.

Again, it is important to remember that these rules are implemented in the CAN Controller silicon.

2.4 CAN Error Handling – Guaranteed Message Delivery

The CAN Protocol rules require that each CAN Controller include elaborate transmission and reception error detection and handling processes. This automatic error handling includes the detection of bit errors and error frame detection or indication and further increases the CAN message transfer reliability.

A similar error handling mechanism for a UART-based or SPI-based protocol must be implemented in software.

In combination, the acknowledgment process, the CRC, and the error handling process – all accomplished in the CAN Controller silicon – virtually guarantee each and every message transfer.

2.5 Implementing CAN - Only Three Ingredients are Required

As shown in Figure 4, functionally implementing a CAN node in a module essentially requires three key items – software, a CAN Controller, and a suitable Physical Layer.

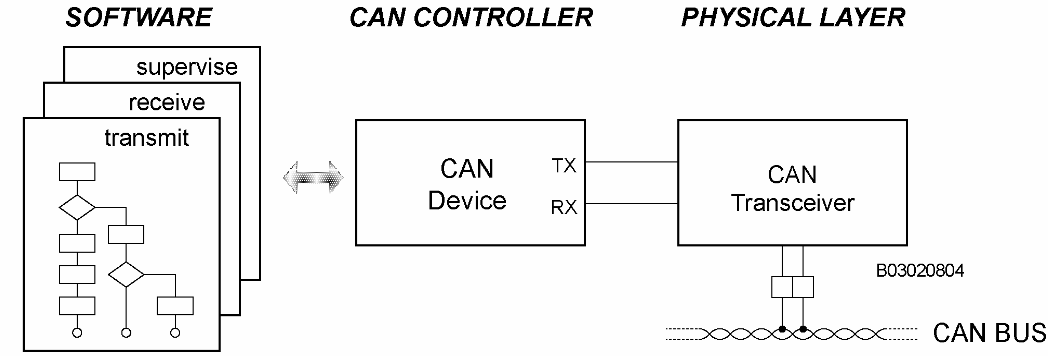

Figure 4 – Key Implementation Components

2.5.1 CAN Controller

CAN Controllers are available as integrated silicon peripherals co-resident with the microcontroller or as separate, external integrated circuits. A wide variety of CAN Controllers are available across the entire semiconductor industry. Some devices include multiple CAN Controllers. The CAN Controller has two major interfaces – one, an interface that interconnects to the selected Physical Layer, and the other interface that interconnects to the network communication software.

Two prominent CAN Controller architectures are prevalent:

- a simplified transfer register buffer architecture – also sometimes referred to as "Basic CAN"

- a moderately complex Dual Port RAM-based architecture – also sometimes referred to as "Full CAN"

All new CAN Controllers implement all of the CAN Protocol rules as defined by the CAN 2.0B specification. Several semiconductor manufacturers also provide special features to allow special CAN operations. For example, a "listen only" mode is available on some CAN Controllers.

2.5.2 CAN Communication Software

CAN communication software is necessary to interconnect the network-based communication system to the module application. Receiving a message in the CAN Controller does not automatically result in any action – the information from the message must be collected, examined, and then the appropriate action taken. The CAN communication software handles this process.

Likewise, when a module needs to move some particular application-specific information onto the network, the communication software is used to load the corresponding message into the CAN Controller for transmission.

Typical CAN communication software components might additionally include:

- network initialization
- transmit message construction
- receive message parsing
- network supervision
- network relationship management
- network error management – like "bus off" handling

2.5.3 Using Off-the-Shelf CAN Communication Software

Many companies use third-party off-the-shelf CAN communication software components rather than devote internal resources to develop their own. One key advantage of using off-the-shelf CAN communication software is the significant reduction in overall software development. Why spread resources across both the communication software and the application software activities? Instead, using prepackaged communication software allows a company to concentrate on its core product software development.

Others have chosen off-the-shelf software simply to reduce the time to market.

While the off-the-shelf software concept may be somewhat unfamiliar, its practice is widespread in the automotive industry. The reason for this is simple. Virtually all car companies realize that off-the-shelf communication software offers a consistency level well beyond the software which is individually developed by each supplier. This is one of the major lessons the auto industry has learned – misinterpreted communication specifications equals delayed programs.

Vector is a major off-the-shelf software component supplier for the auto industry. If you are interested in learning more about Vector software components, please contact us – we have several application notes to introduce the general concepts, both from the business and technical aspects.

2.5.4 CAN Physical Layer

The CAN Physical Layer encompasses the network interconnection, the physical wiring or the selected media, connectors, in most cases a transceiver device, and any other interface-associated components. The Physical Layer provides a direct logical interface to the CAN Controller, and is one of the areas directly related to hardware design.

Even though a suitable Physical Layer is required, the CAN protocol is essentially independent of the type of Physical Layer implementation as long as two simple requirements are satisfied:

- the Physical Layer must support Dominant/Recessive Logic
- the interface must compare the transmitted bit with the receive bit

In combination, dominant/recessive logic and the ability to compare the transmit bit with the receive bit eliminates the possibility that messages from competing transmitters will collide during a message transfer. Unlike Ethernet and other peer-to-peer UART-based protocols, which must detect and resolve collisions, the CAN Protocol guarantees that there will be no message collisions. All of the network bandwidth is available for message transfer.

Several Physical Layers are available, and many are industry-specific standards. For example, ISO 11898 is perhaps the largest auto industry high-speed CAN Physical Layer specification.

The semiconductor industry has created a wide range of CAN transceiver components to accommodate these different CAN Physical Layer standards. Several CAN interfaces are tailored to the expected environment and, in some instances, may include twisted pair wiring, cable shielding, additional protection components, or additional power connections.

For companies which do not have the opportunity to use existing off-the-shelf CAN Physical Layer standards, the selection process of a suitable Physical Layer may be somewhat difficult. Those with previous UART-based networking experience should find the transition relatively easy.

2.6 CAN - The (Almost) Perfect Protocol

Is the CAN Protocol perfect? No small area network solution is perfect, but the CAN Protocol is a quantum leap forward in comparison to any of its predecessors.

Including slight variations across different CAN Controller implementations, the CAN Protocol itself has a few flaws and wrinkles. However, these obscure anomalies are easy to deal with once they are understood. At the system level, always keep in mind that we have software sitting above the CAN Controller's message transfer mechanisms to solve issues.

3.0 CAN Application Ideas - Unlimited Applications

CAN is not limited to any particular application domain. The opportunities and application areas for CAN are unlimited.

Don't think that single network architectures are the rule – the auto industry is already using multiple CAN networks with gateways to interconnect subsystems.

The Physical Layer is not limited to wire. CAN Physical Layer choices include:

- fiber-optic
- infrared
- ultrasonic
- audio
- RF
- magnetic
- power line carrier

Some of these are already commercially available; we are waiting for the others.

Don't let the automotive industry's mindset influence creativity – the designer should feel free to play with infinity.

With a little ingenuity, CAN application areas can easily include:

- unidirectional communications – independent of distance
- interprocessor communications – faster than 1M bps
- high-speed audio distribution – simultaneous dual-channel, stereo MP3 file transfers at 500K bps
- infrared communications
- wireless communications
- communication links using RS232, RS422, or RS485

- replacement of MIL1553-based subsystems

4.0 Next Step Considerations

4.1 Key Self-Discovery Points on the CAN Learning Curve

The following is a list of difficult hurdles that will be encountered while going up the learning curve

- understanding dominant/recessive logic
- understanding transmit-receive bit comparison
- understanding Physical Layer issues – especially implementing high-speed CAN
- understanding the error frame
- understanding CAN bit timing
- understanding differences in CAN Controllers
- understanding CAN Protocol anomalies
- understanding tool requirements to solve problems

4.2 Learning More About CAN

Getting up the CAN learning curve is not a trivial task, but learning about the CAN Protocol down to the atomic level is definitely not recommended. Few need to understand all the details. A better approach might be to focus on that particular aspect or area of concentration that is relevant to your involvement.

The difference between being a distributed systems designer, a module hardware developer, an electrical system wiring expert, or a software programmer has everything to do with what portions of the CAN knowledge base need to be mastered. In many cases, there is a big chunk of probably unnecessary information – knowing what is important and what is not important makes the learning process a little less difficult.

4.3 CAN Tools

While most CAN developers – especially in the automotive industry – use the higher-powered Vector tool CANoe, the beginning engineer new to CAN is perhaps better off using the more economical CANalyzer. True, other lower-cost tools are available for getting started, but most of these tools fail at one very important point: the inability to detect error frames. This somewhat obscure subtlety of the CAN Protocol, which usually has no importance during the beginning portion of the learning curve, means the encounter requires a certain amount of self-discovery by the engineer. Many make the transition to CANalyzer when they discover that error frame detection is extremely important.

Vector can help with tool selection.

4.4 Vector Training and Consulting Services

To understand the CAN Protocol in more depth, we would suggest taking take a CAN training course that provides the opportunity to learn more about small area network technology, the details of the CAN Protocol, and many of the key aspects of the CAN Physical Layer. Vector offers a 2-day combination CAN Protocol plus CANalyzer training class every month.

For those companies preparing to deploy CAN into their new distributed product architectures, consider using Vector technical and business consulting services.

5.0 Contacts

Vector Informatik GmbH
Ingersheimer Straße 24
70499 Stuttgart
Germany
Tel.: +49 711-80670-0
Fax: +49 711-80670-111
Email: info@vector-informatik.de

Vector CANtech, Inc.
39500 Orchard Hill Pl., Ste 550
Novi, MI 48375
USA
Tel: +1-248-449-9290
Fax: +1-248-449-9704
Email: info@vector-cantech.com

VecScan AB
Theres Svenssons Gata 9
41755 Göteborg
Sweden
Tel: +46 (0)31 764 76 00
Fax: +46 (0)31 764 76 19
Email: info@vecscan.com

Vector France SAS
168 Boulevard Camélinat
92240 Malakoff
France
Tel: +33 (0)1 42 31 40 00
Fax: +33 (0)1 42 31 40 09
Email: information@vector-france.fr

Vector Japan Co. Ltd.
Seafort Square Center Bld. 18F
2-3-12, Higashi-shinagawa,
Shinagawa-ku
Tokyo 140-0002
Japan
Tel.: +81 3 5769 7800
Fax: +81 3 5769 6975
Email: info@vector-japan.co.jp

Vector Korea IT Inc.
Daerung Post Tower III, 508
182-4 Guro-dong, Guro-gu
Seoul, 152-790
Republic of Korea
Tel.: +82-2-2028-0600
Fax: +82-2-2028-0604
Email: info@vector-korea.com