Course Introduction

Purpose

• This training course provides an introduction to Controller Area Network (CAN) technology, which is used to build networked, multiprocessor embedded systems.

Objectives

• Understand what CAN technology is, why it’s important and where it can be a good design solution.
• Learn the fundamental operating concepts and capabilities of CAN implementations.
• Find out how CAN fits into the 7-layer OSI model.

Content

• 19 pages
• 3 questions

Learning Time

• 30 minutes
What is CAN?

Controller Area Network:

- Two-wire, bidirectional serial-bus communication method
- Originally developed in the mid 1980s by Bosch for automotive use
- Main design objective: economical solution for implementing high-integrity networking in real-time control applications
- Now standardized internationally:
  - CAN 2.0A: ISO11519 — low speed
  - CAN 2.0B: ISO11898 — high speed
  - CAN Validation: ISO16845
- Usage exceeded 200,000,000 nodes in 2001, still growing at a 30% rate annually
  - Many current and potential non-automotive application opportunities
Non-automotive CAN Applications

- Electronically controlled production and packaging equipment
  - Machine tools; machines for molding, weaving, knitting, and sewing; systems for folding and wrapping; etc.
- Industrial freezers, printing machines
- Ships, locomotives, railway systems
- Farm and construction machinery
- Semiconductor manufacturing equipment
- Building automation: HVAC systems, elevators, etc.
- Hospital patient-monitoring systems
- Many others

More application information available at: www.canopen.us
Key Reasons for Using CAN

1. **Reliability**
   - Error-free communication

2. **Economy**
   - Low wiring cost
   - Low hardware cost

3. **Scalability**
   - Easy expandability
   - Low node-connection costs

4. **Availability**
   - More chips with CAN hardware
   - More off-the-shelf tools
   - Higher-level protocols

5. **Popularity**
   - Knowledge base expanding
## Main Features of CAN

<table>
<thead>
<tr>
<th>Features</th>
<th>Benefits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Has a multiple-master hierarchy</td>
<td>For building intelligent and redundant systems</td>
</tr>
<tr>
<td>Provides transfer rates up to 1 Megabit/sec</td>
<td>For adequate real-time response in many embedded control applications</td>
</tr>
<tr>
<td>Allows 0-8 bytes of user data per message</td>
<td>To accommodate diverse design requirements</td>
</tr>
<tr>
<td>Puts multiple transmit or receive message boxes at each node and assigns each an identifier</td>
<td>For flexibility in system design</td>
</tr>
</tbody>
</table>
## Main Features of CAN

<table>
<thead>
<tr>
<th>Features</th>
<th>Benefits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eliminates addresses of transmitting and receiving nodes in data messages</td>
<td>To save bus bandwidth, simplify software, and allow simultaneous transmission of node-to-node and broadcast messages</td>
</tr>
</tbody>
</table>
| Causes receiving nodes to filter messages based on their assigned identifiers (IDs) | - To simplify node hardware and software  
- To permit message prioritization  
- To allow the hardware to arbitrate the CAN bus |
| Automatically retransmits messages if corruption occurs                   | For accurate communication, even in noisy environments                    |
| Provides error detection, signaling and fault-confinement measures       | To ensure highly reliable network operation                               |
Design Factors to Consider

- **Distance/environment**
  - CAN 2.0B: 1Mbps, up to 40m
  - CAN 2.0A: 125kbps, up to 500m
  - Suitable for difficult environments — automotive, and more

- **Reliability requirements**
  - Integrated error detection and confinement
  - Automatic retransmission of corrupted message
  - Probability of undetected bad message is $<4.7 \times 10^{-11}$

- **Number of nodes**
  - Depends on Physical layer; >100 is feasible

- **Number of masters**
  - Every node can initiate communication and negotiate for the bus

- **Net data transfer rate**
  - Up to 577Kbps net at 1Mbps total data transfer rate

- **Message priority**
  - Message with lowest numerical value identifier wins if two nodes try to transmit at the same time
Data Flow

**CAN bus traffic:**
- The transmitter at a CAN node broadcasts the data frame to all nodes on the bus.
  - Nodes configured to accept the data save it
  - Other nodes do nothing with the data
- CAN 2.0A has an 11-bit message identifier and operates at a maximum frequency of 250kbps.
- CAN 2.0B has 11-bit or 29-bit message identifiers and operates at up to 1Mbps.
Question

Match each CAN item to the most appropriate explanation by dragging the letters on the left to the correct locations on the right. Click Done when you are finished.

A  CAN
B  Multiple-master hierarchy
C  Message identifier
D  Acknowledgment

A  A 2-wire serial bus communication method for multiprocessor systems
B  Enables the design of intelligent and redundant systems
C  Used for addressing, prioritization, and bus arbitration
D  Must be sent by all receiver nodes, or message is re-transmitted
Physical Interface

- Dominant low (voltage) line
- Recessive high line
- Bus must be terminated
- Most common Physical-layer choice: ISO11898-2
**Physical-Layer Implementation**

**CAN transceiver: the Renesas HA13721 ASSP IC:**

- For in-vehicle applications
- ISO11898-2 compliant
- High-speed CAN (up to 1Mbps)
- Active, Standby modes
- Over-temperature detection
- Over-current detection (Vcc-short/GND-short detection)
- Optimized EMI performance
- Txd, MODE input pins; 3.3V compatible

**Diagram:**

- CANH = C_HI
- CANL = C_LO
CAN Bus Data Frame

TX LO levels are dominant (drive bus)

TX HI levels are recessive (bus termination controls)

MCU Output to Transceiver

Transceiver Output to Bus

CAN uses non-return-to-zero (NRZ) serial data
Message Bit Time: 4 Segments

**SYNC SEG**: Nodes are synchronized within this phase

**PROP SEG**: Propagation delay compensation value \[= 2 \times (\text{signal propagation time} + \text{input comparator delay} + \text{output driver delay})\]

**PHASE_SEG1** and **PHASE_SEG2**: Establish correct sampling point
Maintaining Synchronization

‘Bit stuffing’ is applied as needed to keep the bus synchronized:

• Too many consecutive dominant or recessive bits cause the transmitting node to insert a bit of the opposite polarity

• Resulting signal edge is used to establish timing synchronization at all nodes on the bus

• The bit is inserted whenever a sequence of five bits with the same polarity occurs
Maintaining Synchronization

Stuffed bit
Question

Which of these statements correctly describe voltage and timing aspects of CAN bus operation? Select all that apply and then click Done.

A dominant value (positive differential voltage >900 mV) is created by driving the C_HI line high and the C_LO line low.

Mandatory CAN bus termination resistors create a recessive value when all bus nodes go to a high-impedance state.

Because CAN uses NRZ serial data, synchronization between nodes is maintained automatically.

The PROP_SEG portion of the bit time is used to compensate for physical delays within the network.

Done
CAN in the OSI Model

Higher-Layer Protocols
- Data
  - Application
    - Network Process to Application
- Data
  - Presentation
    - Data Representation and Encryption
- Data
  - Session
    - Interhost Communication
- Segments
  - Transport
    - End-to-End Connections and Reliability
- Packets
  - Network
    - Path Determination and IP
      - (Logical Addressing)
- Frames
  - Data Link
    - MAC and LLC
      - (Physical Addressing)
- Bits
  - Physical
    - Media, Signal and Binary Transmission

ISO 11898

CAN Features
- Data Link
  - LLC
    - Acceptance Filtering
    - Overload Notification
    - Recovery Management
- MAC
  - Data Encapsulation/Decapsulation
  - Frame Coding (Stuffing, Destuffing)
  - Medium Access Management
    - Error Detection
    - Error Signaling
    - Acknowledgment
    - Serialization / Deserialization
- Physical
  - Bit Encoding/Decoding
  - Bit Timing
  - Synchronization
  - Driver/Receiver Characteristics

Mouse over any of the blocks containing fine print to learn more.
Higher-layer CAN Protocols

Automotive
- Incompatible OEM
  - GM (LAN3.0)
  - Daimler-Chrysler
  - Ford
  - Toyota, etc.

Industrial
- DeviceNet
- CAN Open
- Proprietary

Other
- NMEA2000
  - (marine)
- CANaerospace
  - (avionics)
- SAE J1939
  - (heavy trucks)
- ISO11783
  - (agricultural vehicles)
- Proprietary

CAN Interface
Glossary

- **Advanced CAN**: CAN peripheral with varying numbers of buffers configurable for transmit/receive. Receive buffers have hardware filtering on at least mask/match identifier content.
- **Basic CAN**: CAN peripheral with no hardware filtering. Typically two receive buffers act as a FIFO and accept all bus traffic. Usually one transmit buffer.
- **Bit Time**: Nominal time of one bit on the CAN bus. Made up of multiple segments that allow each node to synchronize to the received bus traffic. All nodes on a bus must be configured to the same (nominal) bit time.
- **CAN**: Controller Area Network.
- **CAN 2.0B**: Version 2.0 was the last version of CAN defined by Bosch. Part B added extended identifiers and the idea of hardware filtering.
- **CIA**: CAN in Automation. Group controlling the CANOpen protocol.
- **CANOpen**: Multi-area communication protocol using CAN.
- **CRC**: Cyclic Redundancy Check.
- **DeviceNet**: Industrial communication protocol using CAN.
- **Dominant/Recessive**: Dominant bits on Physical layer can override recessive bits.
- **Filters**: Hardware in the CAN peripheral that can mask/match bits within the identifier field. Used to determine whether or not to route bus data to a mailbox.
- **GM LAN 3.0**: GM protocol. Encompasses all GM serial protocols.
- **Identifier**: Frame field that indicates the message content. This field also is used to arbitrate the message priority on the bus. A lower ID has a higher priority.
  - **Standard Format**: Frames use an 11-bit identifier.
  - **Extended Format**: Frames use a 29-bit ID.
- **ISO 11898**: ISO standardized version of CAN.
- **Mailbox**: CAN hardware buffer that can be used to transmit or receive data. Most FullCAN implementations have at least 16 mailboxes.
- **Time Quanta**: Smallest time unit used by CAN. Multiple time quanta make up the segments of a bit time.
- **TT CAN**: Time Triggered CAN. More deterministic form of CAN. Assigns time slots when nodes may transmit.
- **FlexRay**: Next-generation automotive network. Time slots on the bus provide more deterministic behavior.
- **Vector-CanTech**: Supplier of the majority of CAN software drivers and tools for ECUs of North American and European automobiles.
Question

Is the following statement true or false? Click Done when you are finished.

“Basically, CAN is concerned with the lowest layers of the OSI model, but CAN 2.0B also implements part of the transport layer.”

True

False

Done
Course Summary

- CAN description, applications, features/benefits
- Design factors/parameters
- Data frame and data flow
- Timing issues
- Physical interface, OSI model, and higher-level protocols
- Glossary