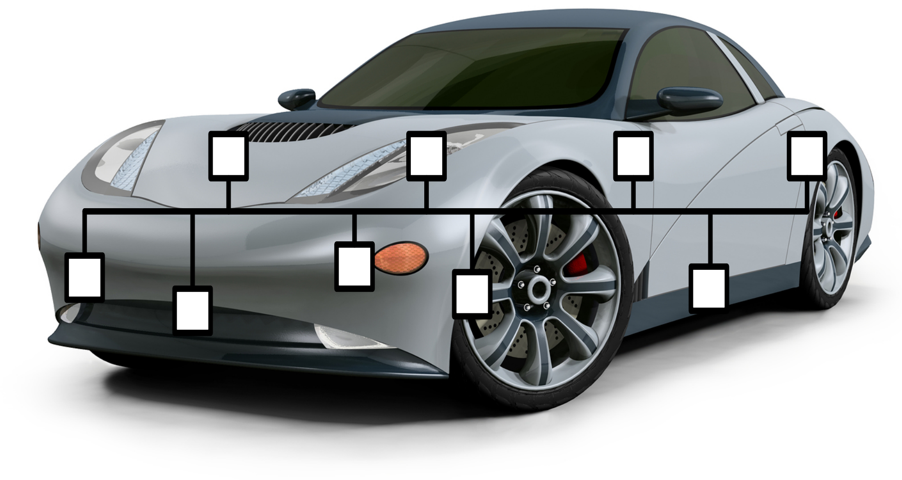
Controller Area Network
Serial Network Technology for Embedded Solutions

COPPERHILL technologies

Controller Area Network

Serial Network Technology for Embedded Solutions



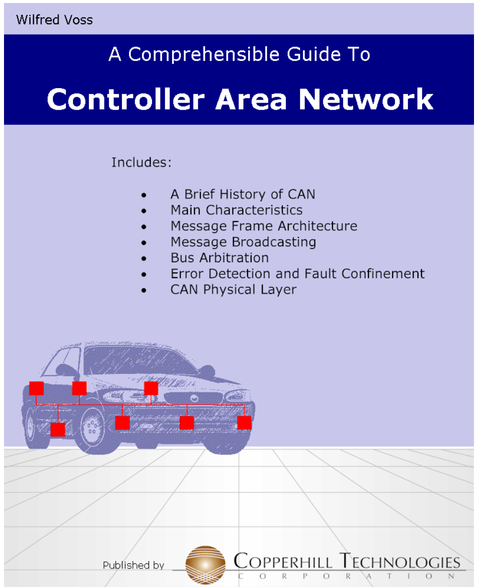
Presented by
Wilfried Voss

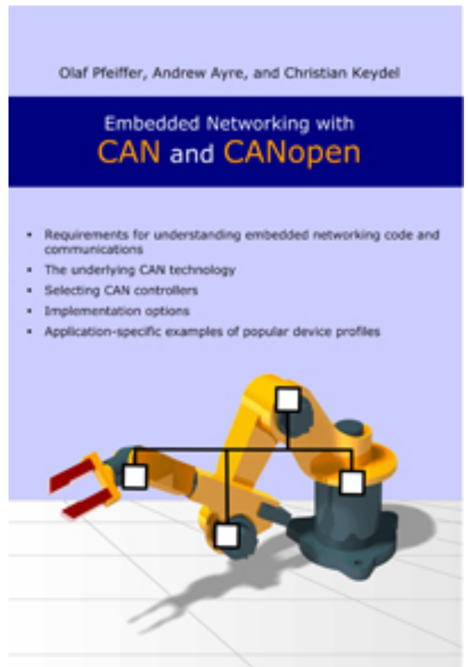
Copperhill Technologies Corp.
Greenfield, MA 01301

https://copperhilltech.com

Literature

Literature on
Controller Area Network,
CANopen and SAE J1939





https://copperhilltech.com/technical-literature/

What is CAN – General Aspects

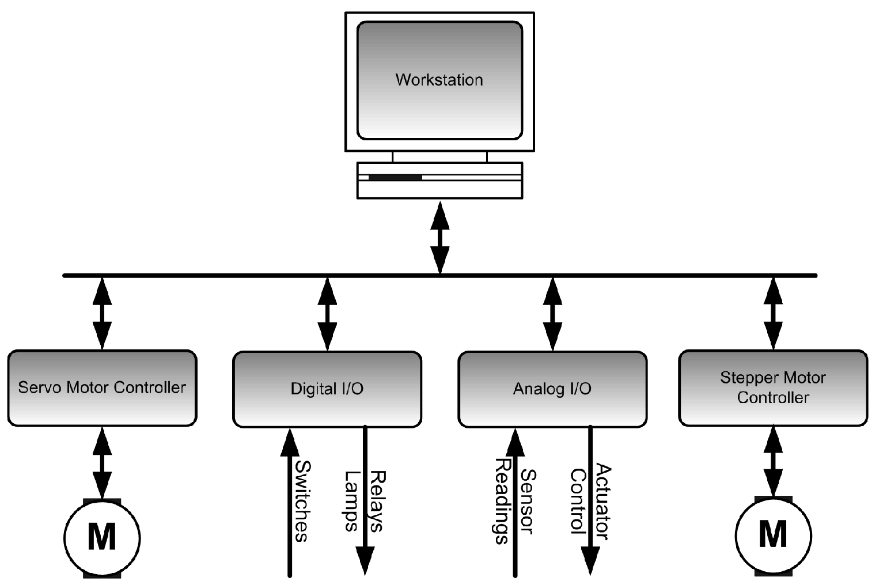
- Serial Network Technology for Embedded Solutions

- Originally designed by Bosch for automotive industry
- Became very popular in industrial automation

- Network technology established among micro-controllers
- Well suited for high speed/real-time applications
- Replaces expensive Dual-Port RAM technology
- CAN chips manufactured by Motorola, Philips, Intel, Infineon, and more
- 600 Million CAN nodes used in 2007

What is CAN – Technical Aspects

- High-integrity serial data communications bus for real-time applications
- Designed for max. performance & reliability
- Operates at data rates up to 1 Mbit/sec
- Uses short messages – 8 bytes per message
- Excellent error detection and fault confinement capabilities
- Is an international standard: ISO 11898

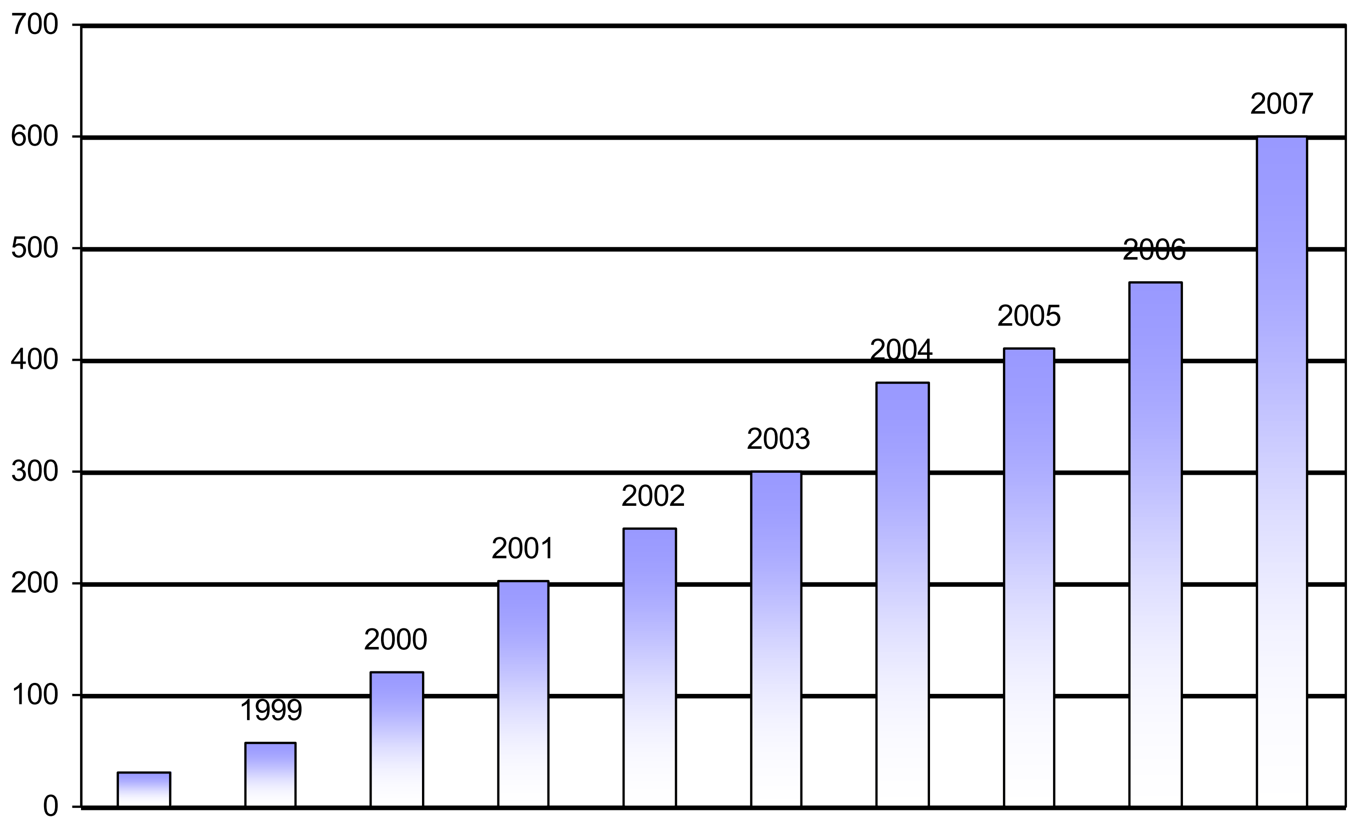


A Brief History of CAN - 1

1983	Start of Bosch internal project to develop in-vehicle network
1986	Official introduction of the CAN protocol
1987	First CAN controller chips by Intel & Philips
1991	Bosch publishes CAN specification 2.0
1992	CAN in Automation (CiA) established
1992	CAN Application Layer (CAL) protocol by CiA
1992	First automobiles equipped with CAN (Mercedes Benz)
1993	ISO 11898 standard published
1994	First International CAN Conference (iCC)
1994	Allen Bradley introduces DeviceNet
1995	ISO 11898 amendment (extended frame format)
1995	CANopen protocol introduced
2000	Development of time-triggered CAN

A Brief History of CAN - 2

Number of Million CAN Nodes sold:



CAN Applications

CAN is used wherever two or more microprocessor units need to communicate with each other.

- Passenger Cars (multiple separate CAN networks)
- Trucks & Buses, Construction Vehicles, Agricultural Vehicles (SAE J1939 protocol)
- Semiconductor Industry (Wafer Handlers, etc.)
- Robotics, Motion Control Applications
- Passenger/Cargo Trains (Brake Control, Wagon Communication)
- Aircrafts (AC, Seat Adjustment)
- Elevators (e.g. Otis)
- Building Technologies (Light & Door Control Systems, Sensors, etc.)
- Medical Equipment (X-Ray, CAT scanners, etc.)
- Household Utilities (Coffee Machine, Washer, etc.)
- Aerospace (Satellites)

CiA – CAN in Automation

CiA

- International Users and Manufacturers Organization
- Develops, supports CAN Standards and CAN based higher layer protocols
- All activities are based on CiA members' interest

http://www.can-cia.org

CAN Newsletter

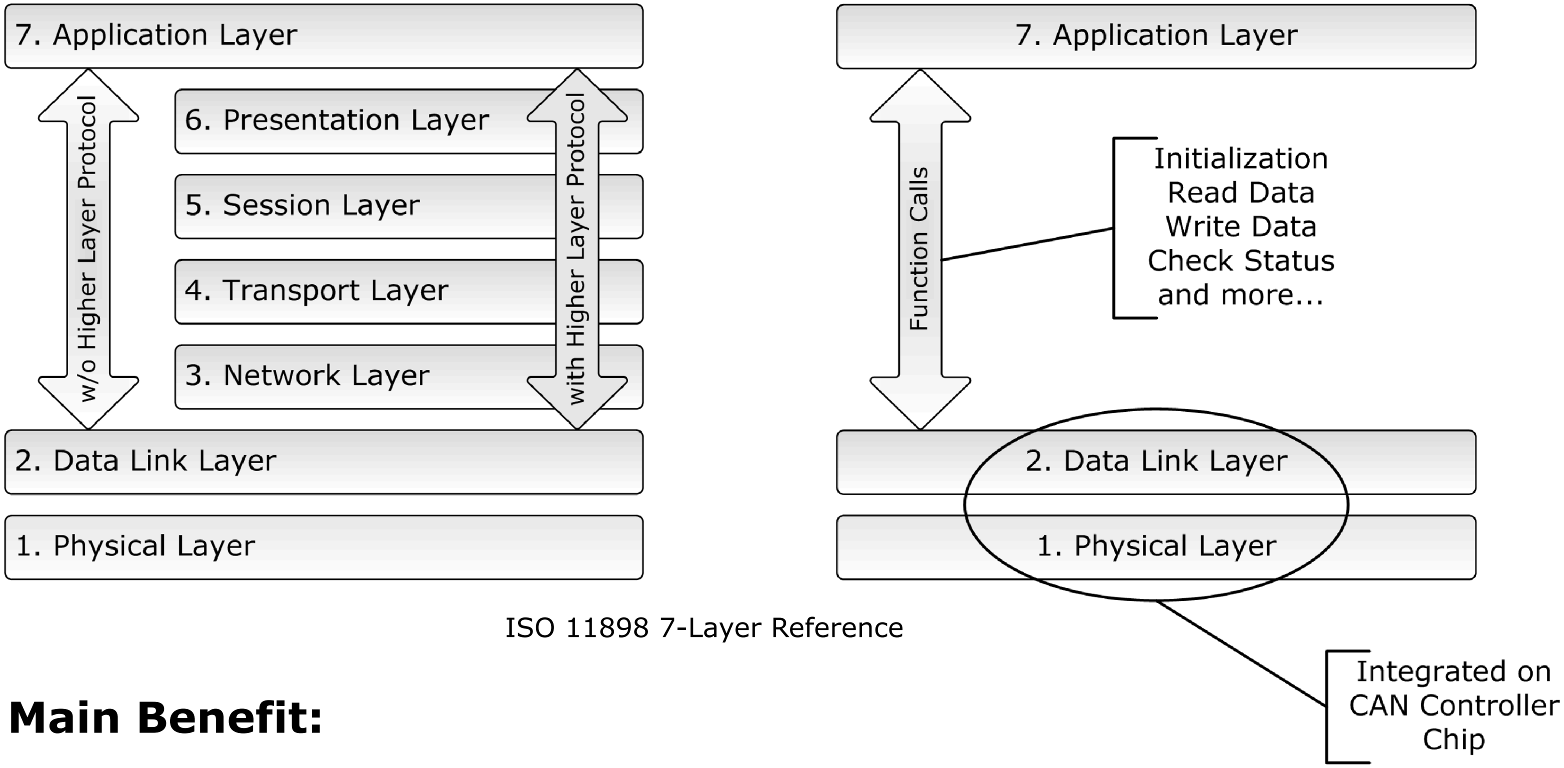
To subscribe log on to:

https://can-newsletter.org/newsletter

Main Characteristics

- Multi-Master Bus Access
- Message Broadcasting
- Message Priority (No Node IDs)
- Limited Data Length (0…8 bytes)
- 1 Mbit/sec Data Rate
- Excellent Error Detection & Fault Confinement

Benefits of Using CAN



ISO 11898 7-Layer Reference

Main Benefit:

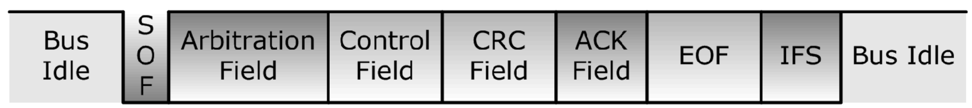
- Physical and Data Link Layer implemented in Silicon
- SW Development Engineer is not involved with writing protocol features
- Low Cost Implementation

Be aware! Whenever you attempt to add software functions between the CAN Data Link Layer and the Application Layer, you will be adding functionalities that are already covered by off-the-shelf available higher layer protocols such as CANopen and DeviceNet.

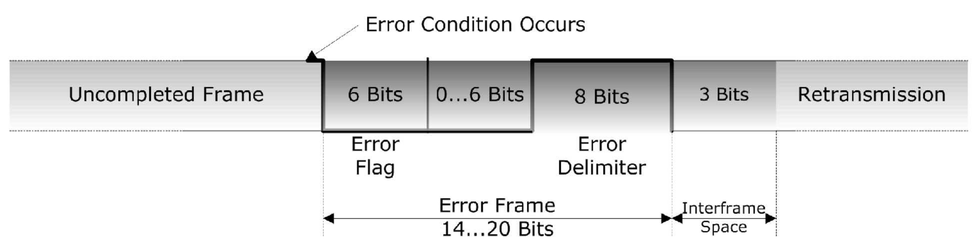
Message Frames

- **Data Frame** – Broadcasts a message to the CAN bus

- **Remote Frame** – Requests transmission of message



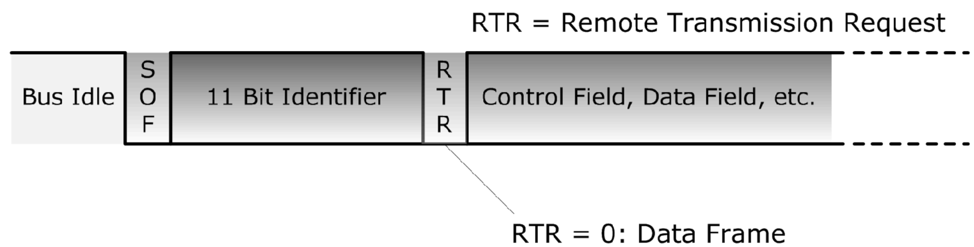
- **Error Frame** – Signals error condition



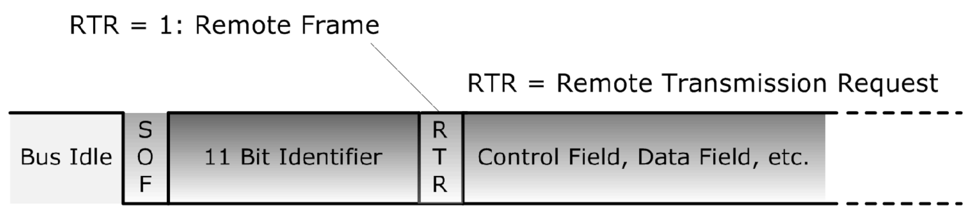
- **Overload Frame** – Special Error Frame

Remote Transmission Request

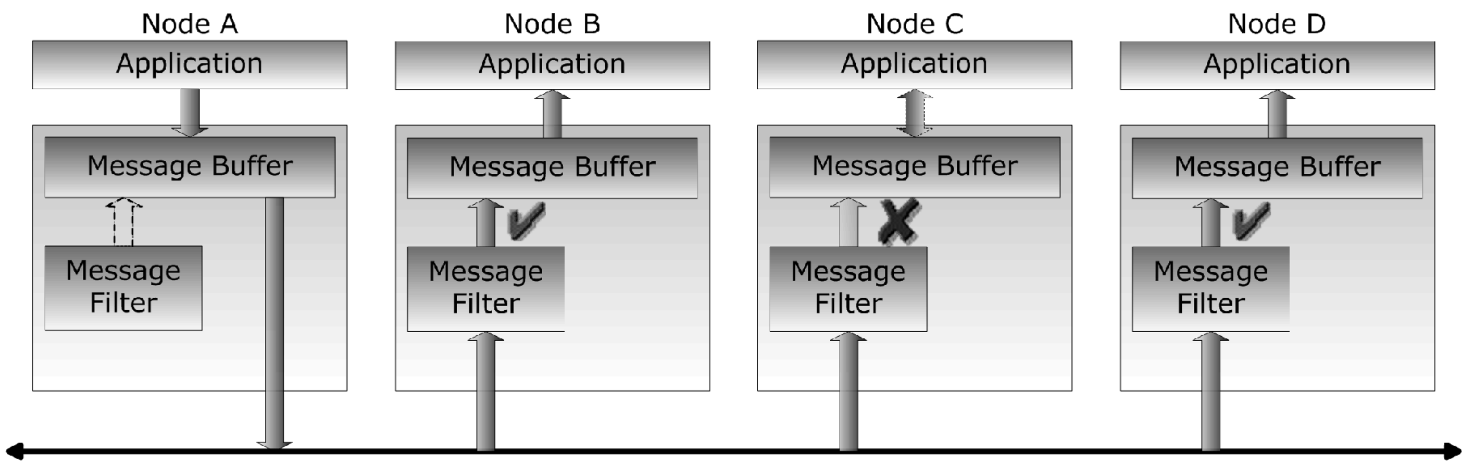
- **Data Frame** – Broadcasts a message to the CAN bus



- **Remote Frame** – Requests transmission of message

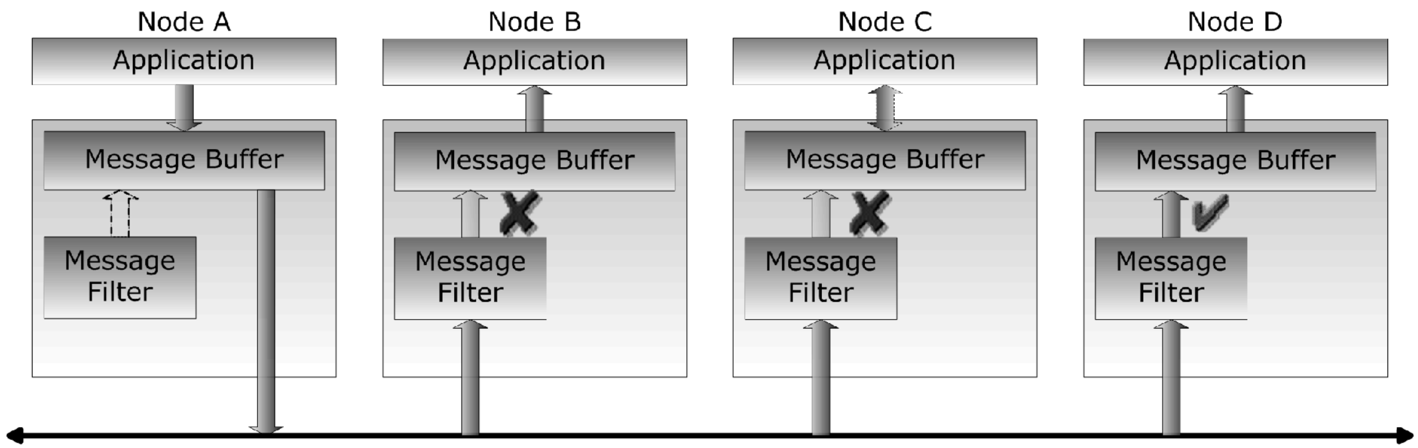


Message Broadcasting with Data Frames



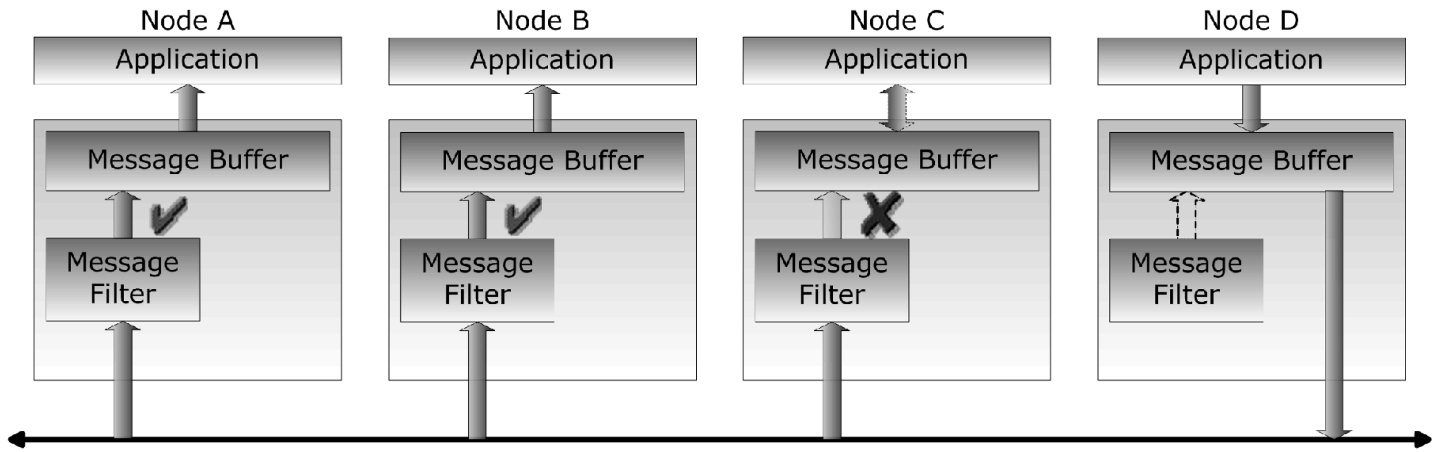
- Node A transmits a message
- Nodes B, C and D receive the message
- Nodes B and D accept the message, Node C declines

Message Request with Remote Frames - 1



- Node A sends a remote frame (request)
- Node B, C, D receive message
- Node D accepts, Nodes B & C decline message

Message Request with Remote Frames - 2

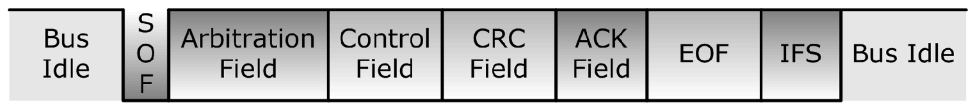


- Node D sends requested message
- Nodes A, B, C receive requested message
- Nodes A, B accept requested message, Node C declines

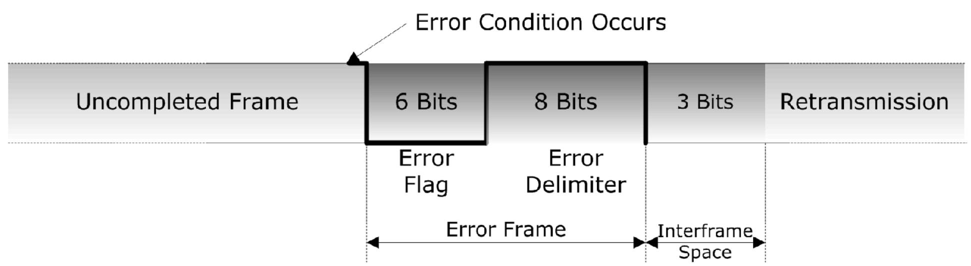
Message Frame Format - 1

- **Data Frame**

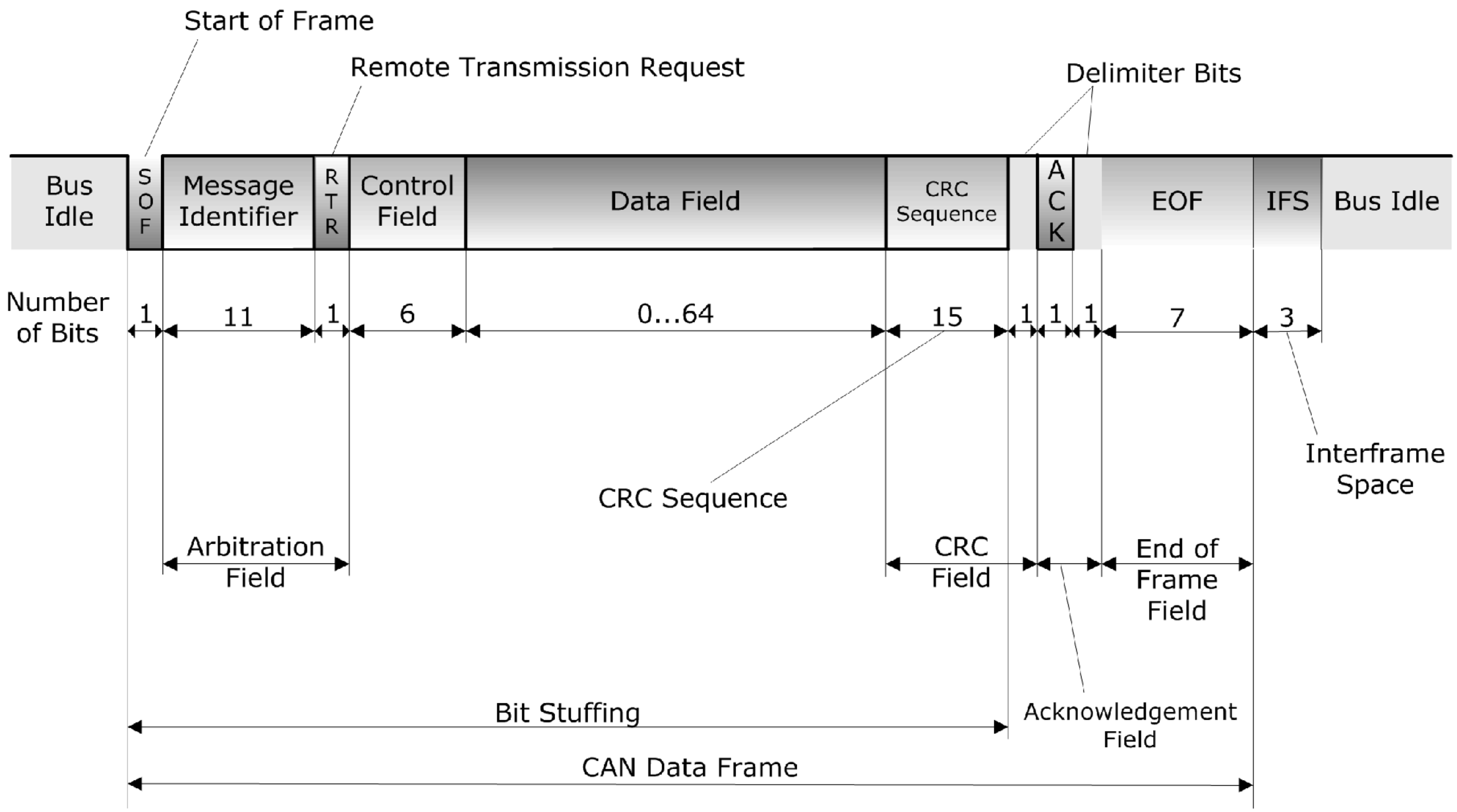
- **Remote Frame**



- **Error/Overload Frame**

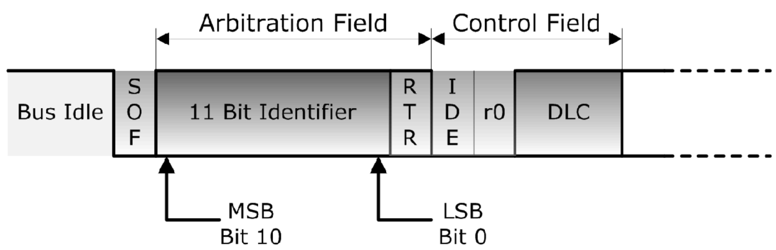


Message Frame Format - 2

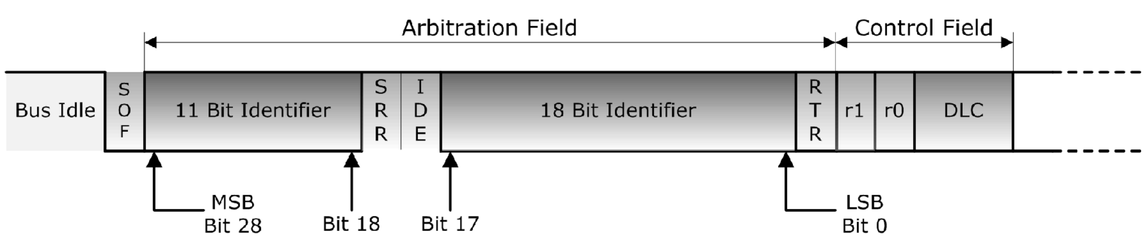


Extended CAN Protocol

- **Standard Format: 11 Bit Message Identifier**



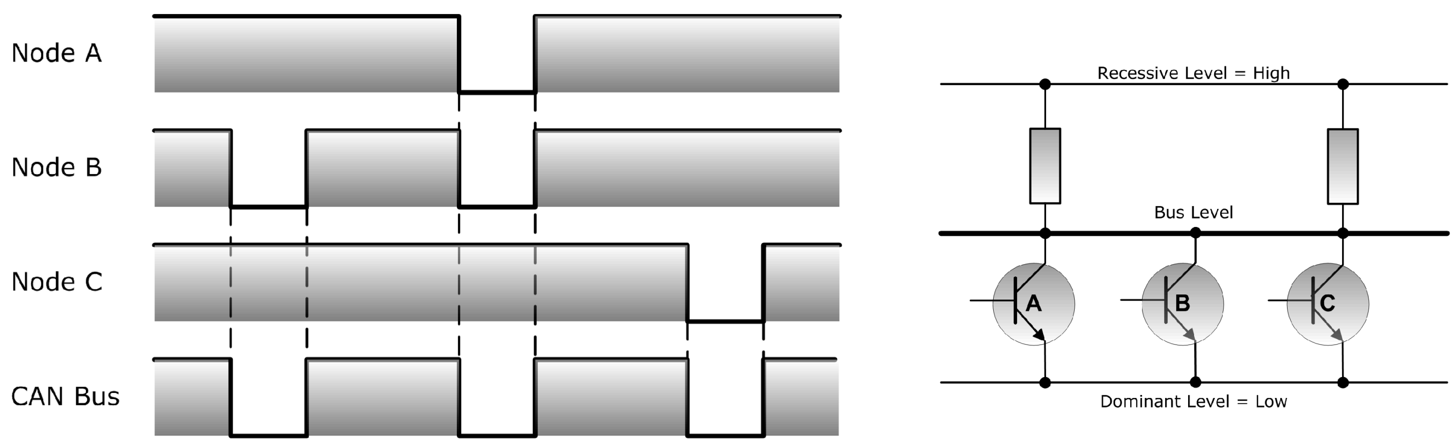
- **Extended Format: 29 Bit Message Identifier**



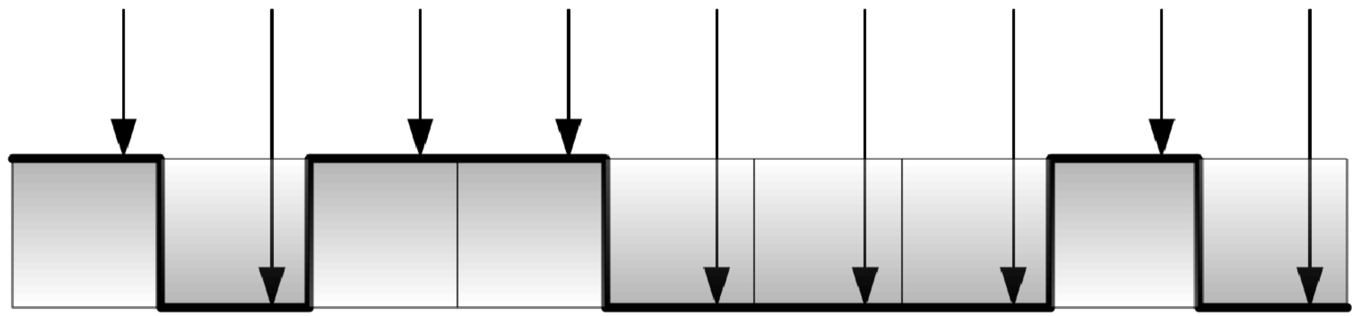
- Both formats, Standard and Extended, may co-exist on the same CAN bus
- The distinction between both formats is managed by “Identifier Extension Bit” (IDE)

Dominant/Recessive Bus Level

Node Output and Resulting Bus Level

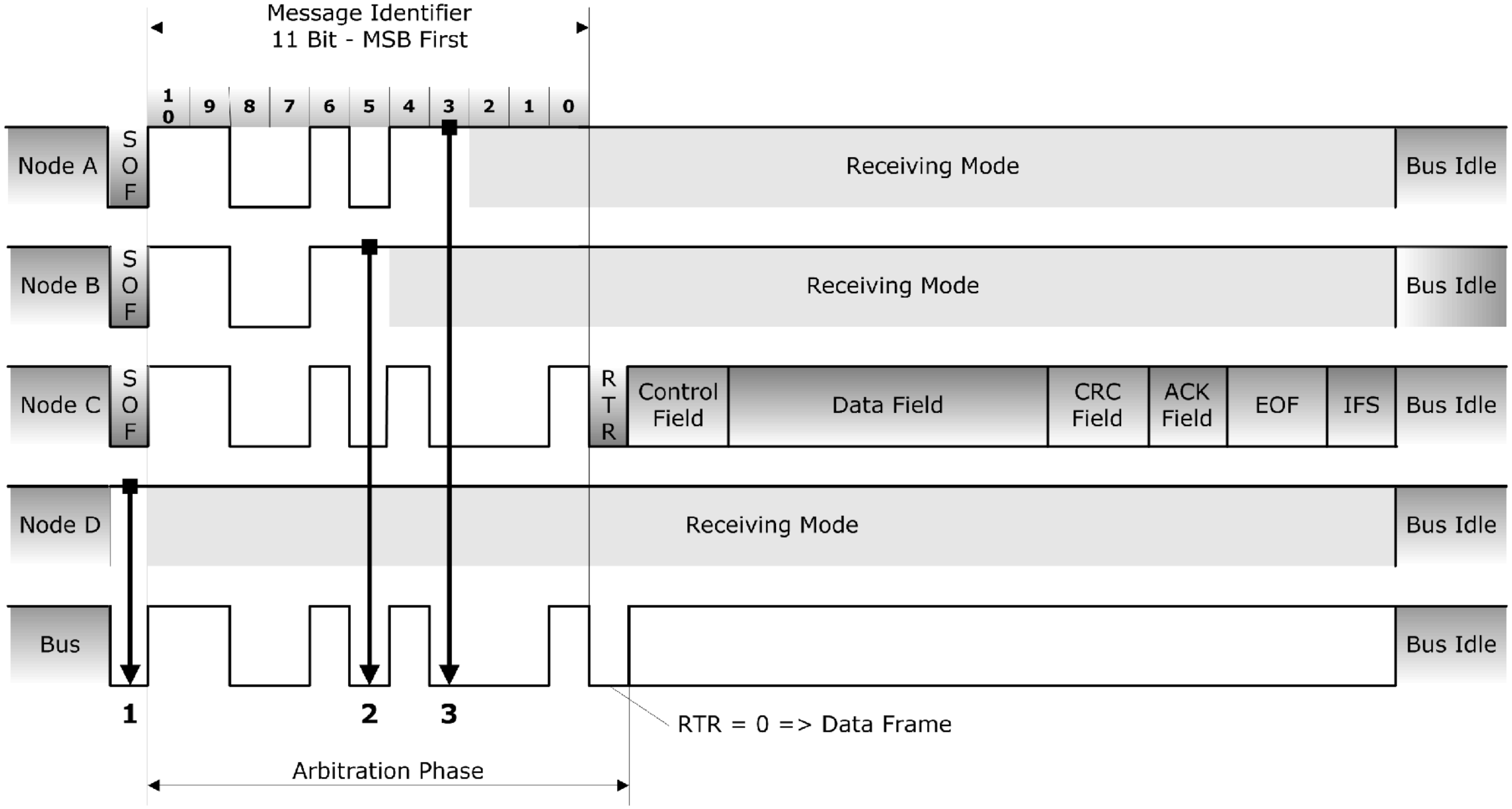


Bit Monitoring

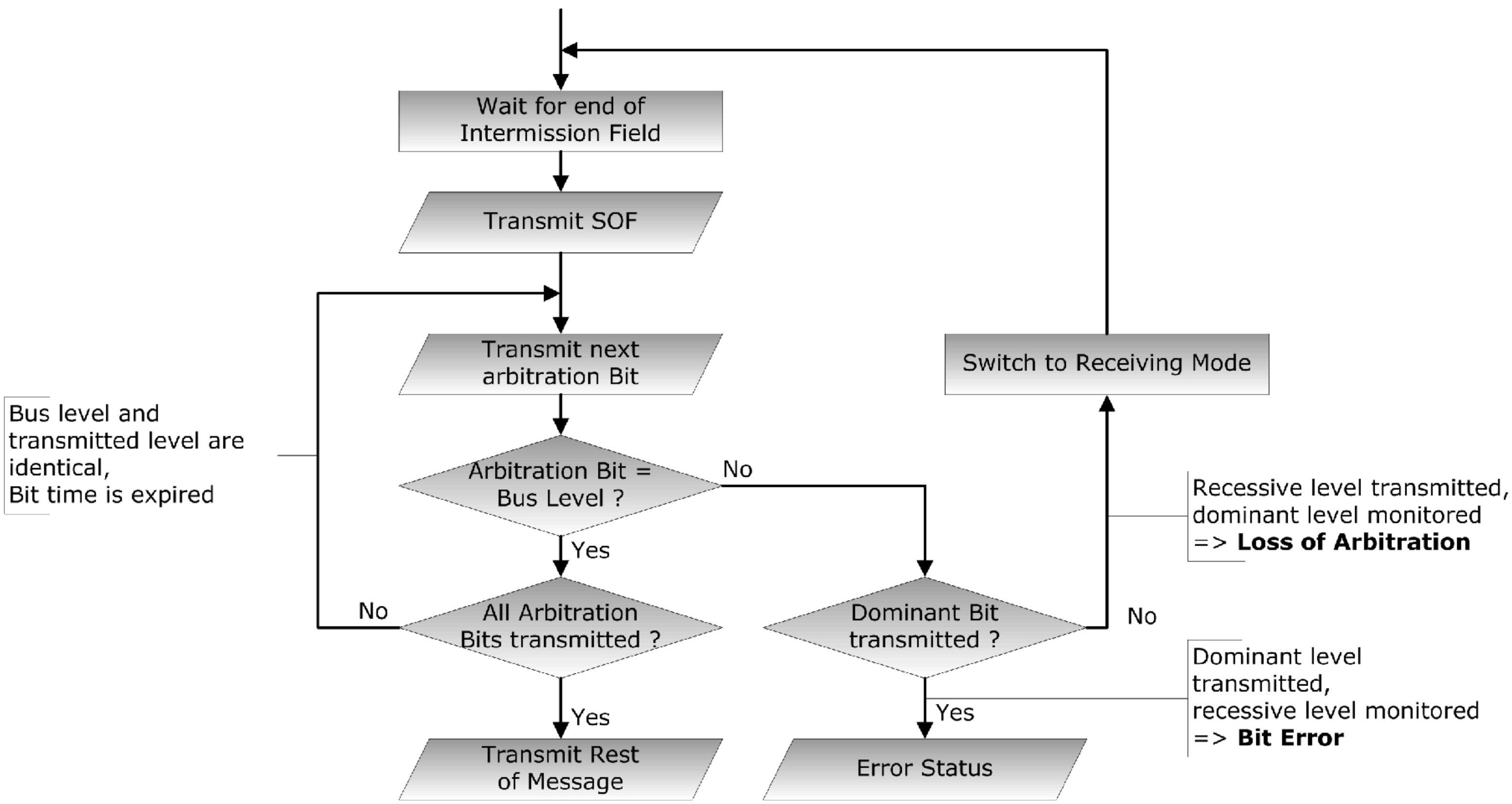


- Each transmitting node monitors the Bit level on the bus, compares it to transmitted level.
- Used during arbitration process.
- Provides immediate detection of all bus-wide and local transmission errors.

Bus Arbitration Principle - 1



Bus Arbitration Principle - 2



Controller Area Network
Serial Network Technology for Embedded Solutions

Bus Arbitration Principle - 3

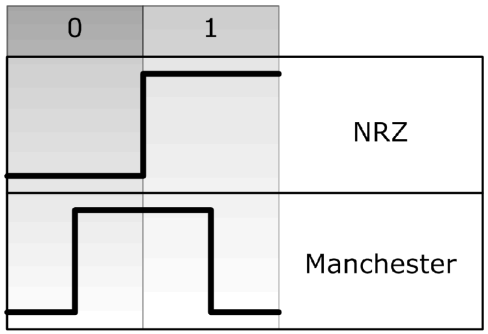
Main Rules of Bus Arbitration

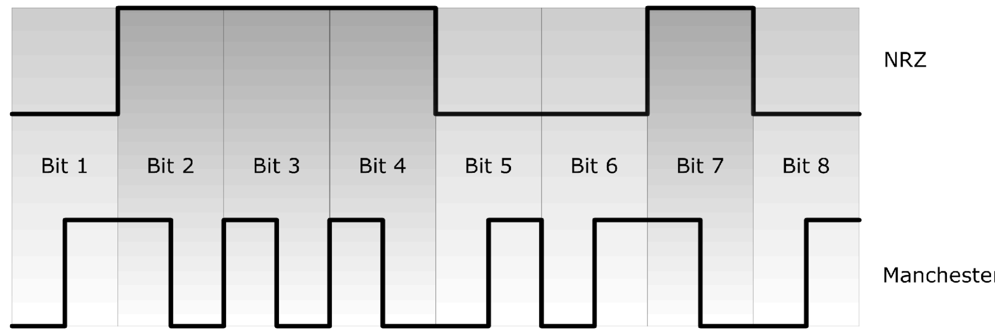
- Bus is considered idle after transmitted message plus Intermission Field
- Node that transmits message with lowest ID (highest priority) wins arbitration, continues to transmit. Competing nodes switch to receiving mode.
- Nodes that lost arbitration will start new arbitration as soon as bus is free for access again => No message gets lost

Data Transfer Synchronization - 1

Bit Coding

- Bit coding according to Non-Return-to-Zero principle



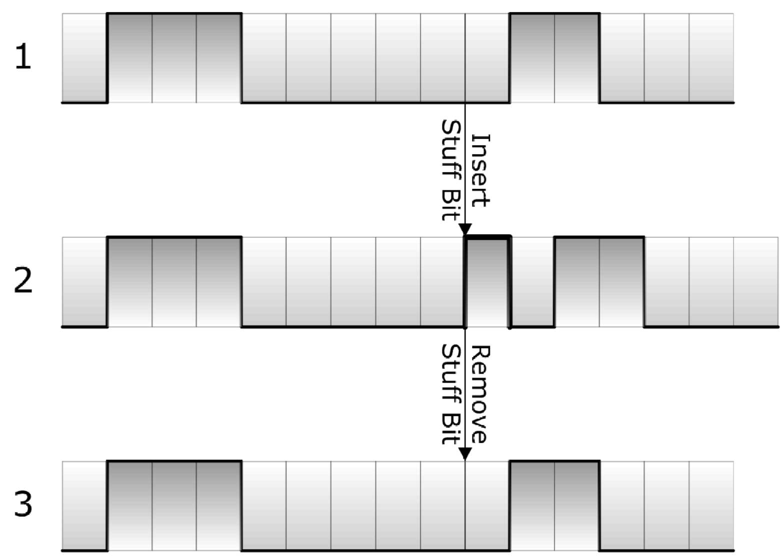


- NRZ provides highest transport capacity
- Constant Bit level over Bit time
- Insufficient signal edges for synchronization of Bit stream
- “Bit Stuffing” required

Data Transfer Synchronization - 2

Bit Stuffing

- Sender inserts complementary Bit ("Stuff Bit") after 5 successive Bits of same polarity
- Receiver filters the complementary Bit

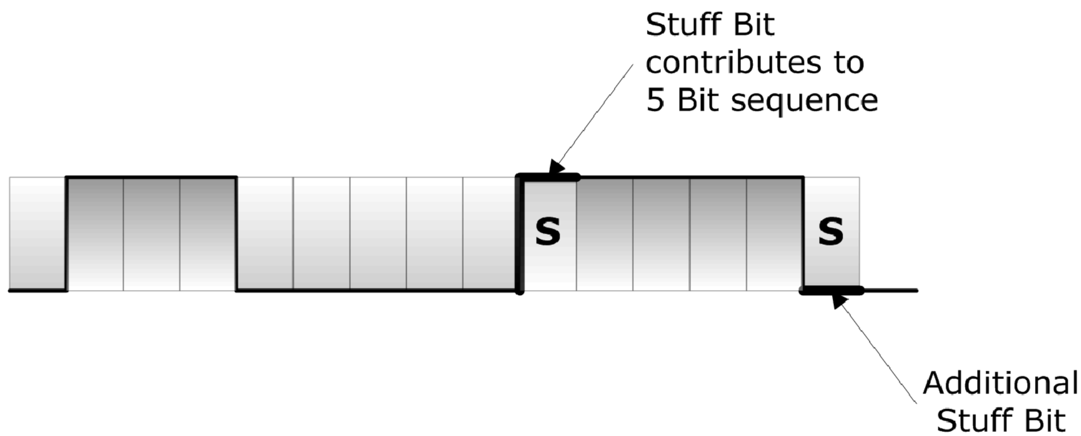


1. Bit sequence to be transmitted
2. Transmitted Bit sequence on bus after bit stuffing
3. Bit sequence at receiver after filtering Stuff Bit

Data Transfer Synchronization - 3

Bit Stuffing

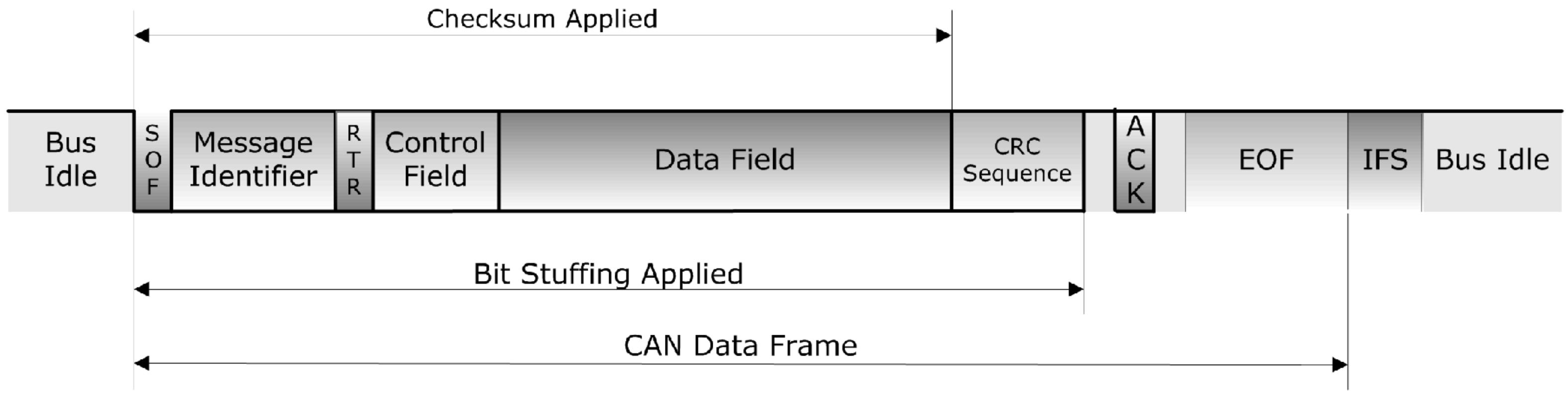
- Provides additional signal edges for data synchronization
- No Bit Stuffing in static format fields of message frame
- Special case: Stuff Bit contributes to 5 Bit sequence



Further Bit Synchronization through

- Programmable Phase Buffers
- Resynchronization Jump Width

Bit Stuffing Range



Frame Length Due to Bit Stuffing

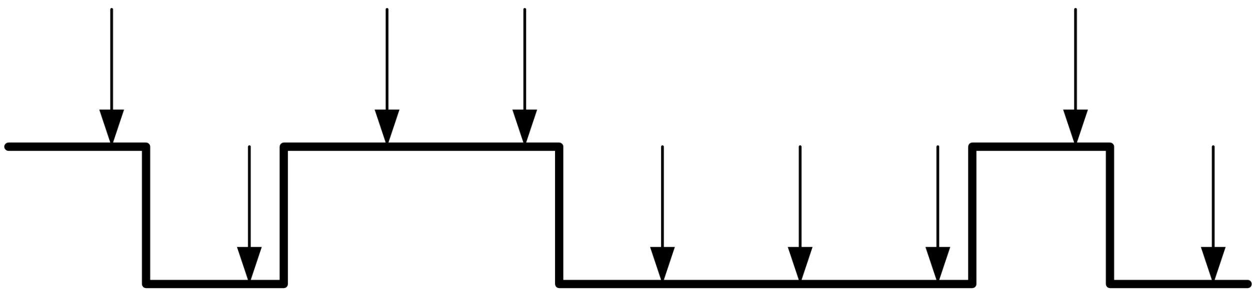
- Frame length varies due to bit stuffing

	Data Field 0 Bytes	Data Field 8 Bytes
No bit stuffing	**47 bits**	**111 bits**
Max. bit stuffing (worst case scenario)	**55 bits**	**135 bits**
Average bit stuffing	**49 bits**	**114 bits**

- Based on 11-Bit Identifier
- Average bit stuffing determined per mathematical model

Bit Monitoring - 1

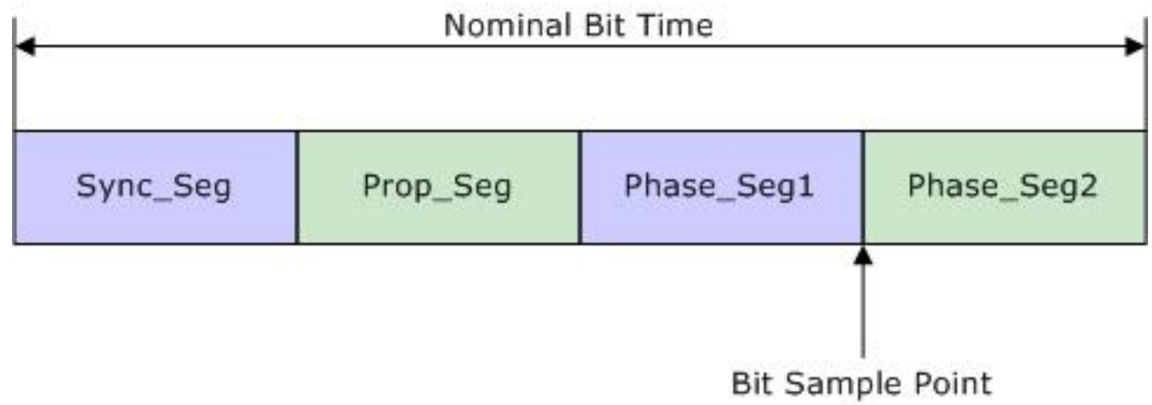
- When exactly does a receiving CAN node read the bit information ?



Bit Rate [kBit/ sec]	Nominal Bit-Time [µsec]
1000	1
500	2
250	4
125	8

Bit Monitoring - 2

- Partitioning of CAN Bit Time into Four Segments



Sync_Seg: Signal edge is expected here. Any deviation will affect Phase Buffer lengths.

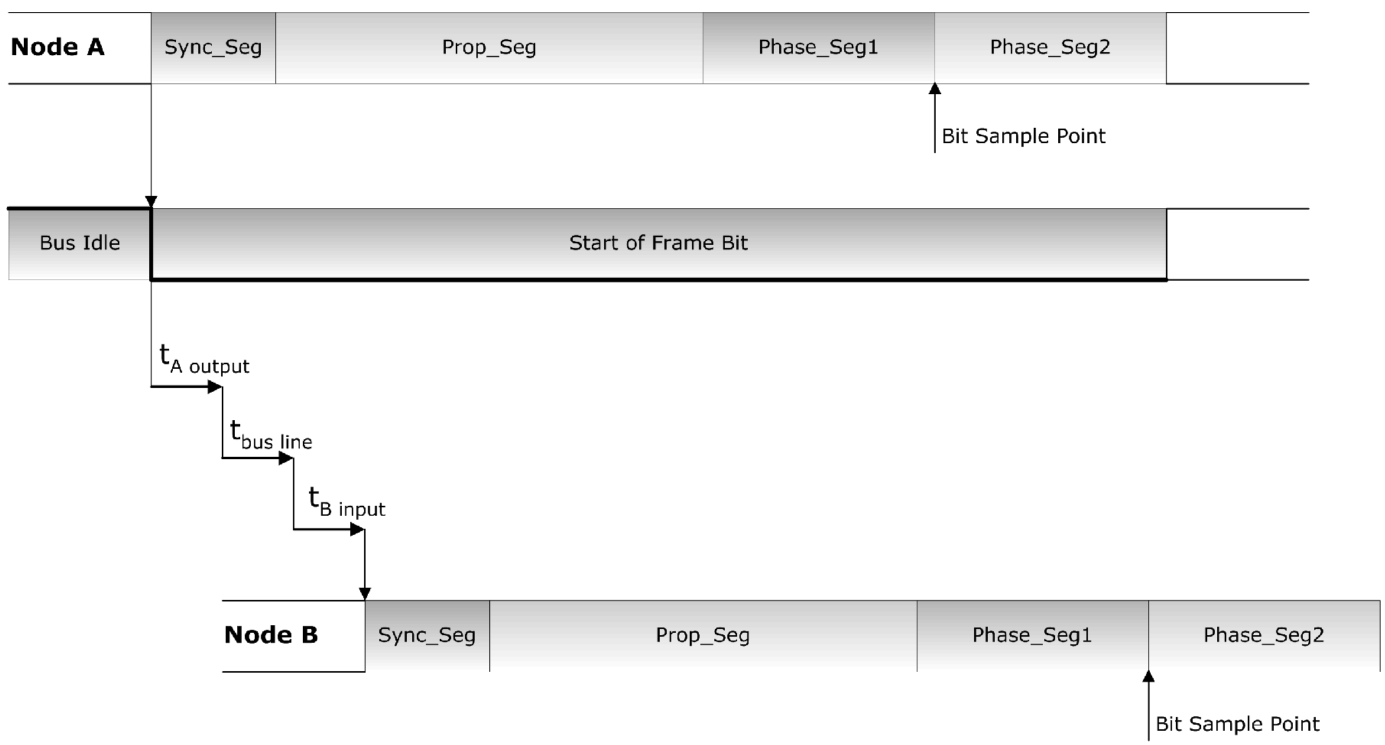
Prop_Seg: Compensates for signal propagation times within the network.

Phase_Seg1/2: Compensate for signal edge phase errors by adjusting their length.

Resynchronization Jump Width: Defines the upper limit to adjust phase buffer lengths.

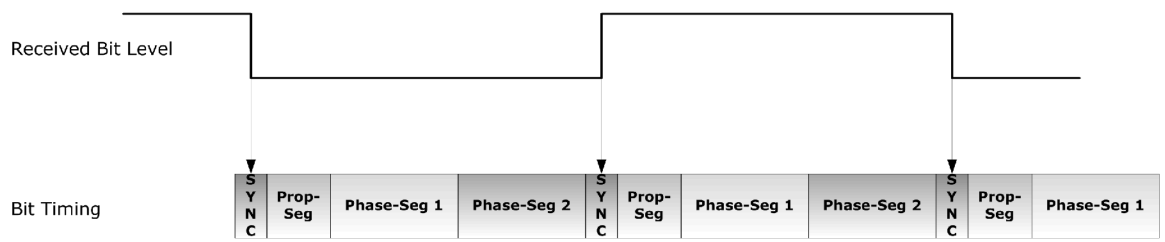
Hard Synchronization

- Hard Synchronization at Start of Frame (SOF)
- Resets internal timing of receiving node



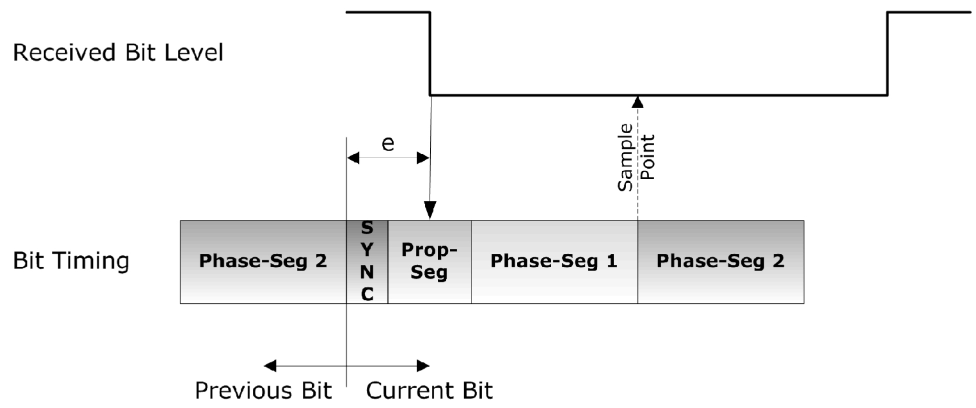
Bit Synchronization

- Bit Synchronization within a frame
- Phase_Seg1 and Phase_Seg2 correct position of bit sample point according to phase error

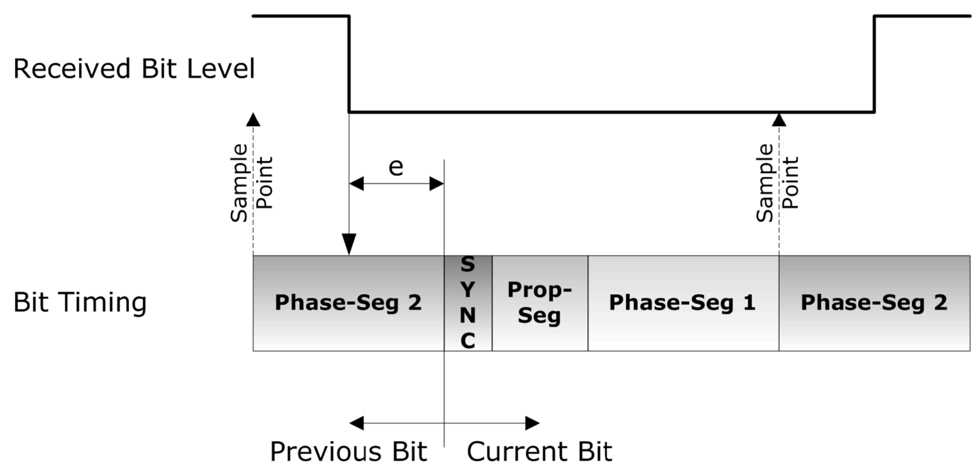


Ideal Bit Timing

Phase Errors

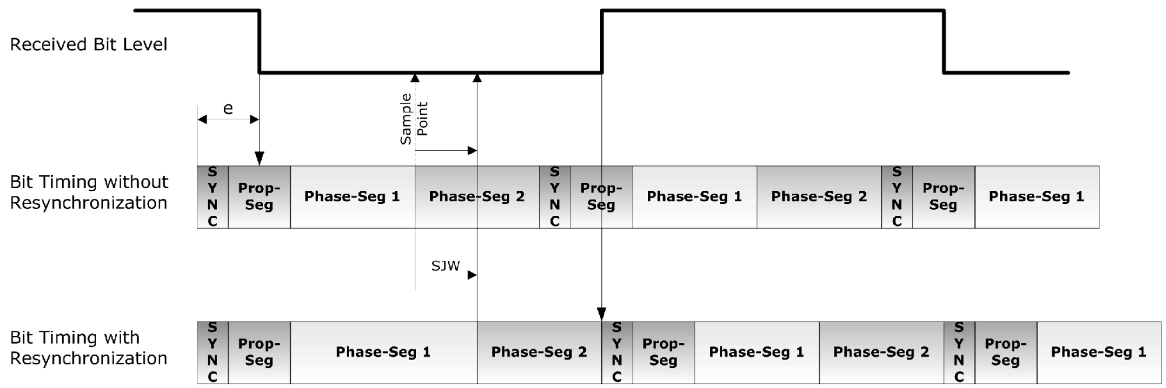


Positive Phase Error

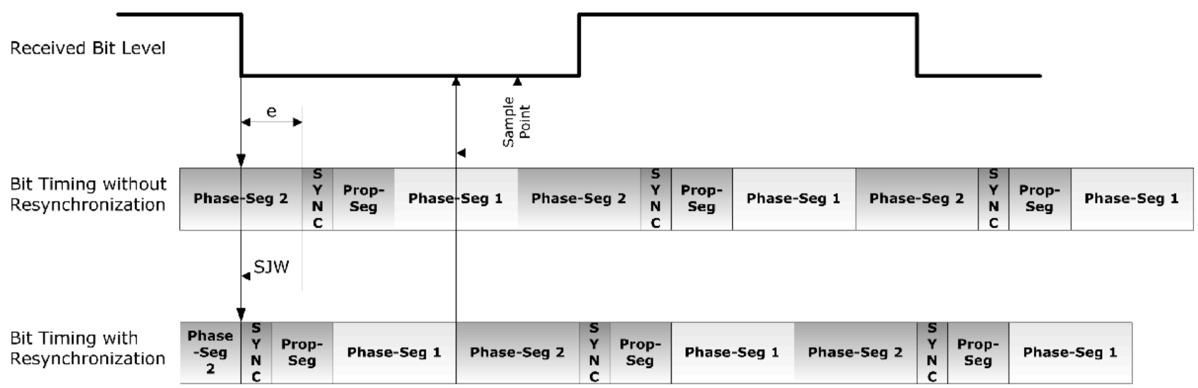


Negative Phase Error

Compensation of Phase Errors



Compensation of Positive Phase Error



Compensation of Negative Phase Error

Error Detection Method

- **Bit Monitoring**

Each transmitting node monitors the Bit level on the bus, compares it to transmitted level. Provides immediate detection of all bus-wide and local transmission errors.

- **Stuff Error**

More than 5 Bits of same polarity outside of “bit-stuffed” segment

- **CRC Error**

Comparison of received CRC sequence and calculated CRC. Provides detection of local receiver errors.

- **Form Error**

Violation of fixed format Bit fields

- **Acknowledgement Error**

Transmitted message receives no acknowledgement. ACK confirms only the successful transmission. Is used for error confinement.

Error Detection Analysis

Probability of non-detected faulty CAN messages

Example:

- 1 Bit error each 0.7 sec
- 500 kBit/sec
- 8 h/day
- 365 days/year

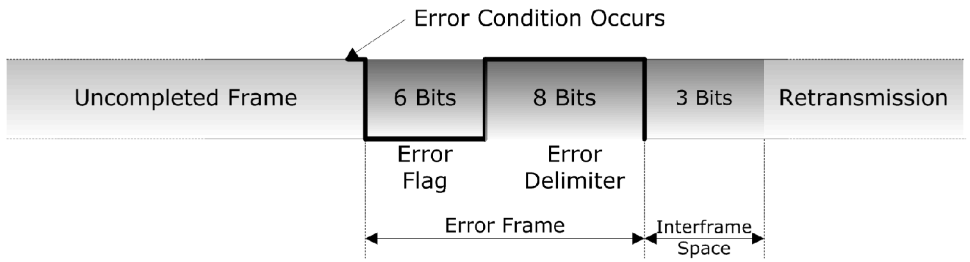
Residual Error Probability :

1 undetected error in 1000 years

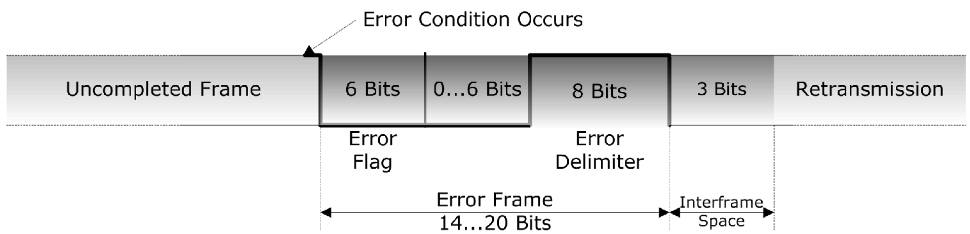
Error Detection

Error Frame

Basic Error Frame

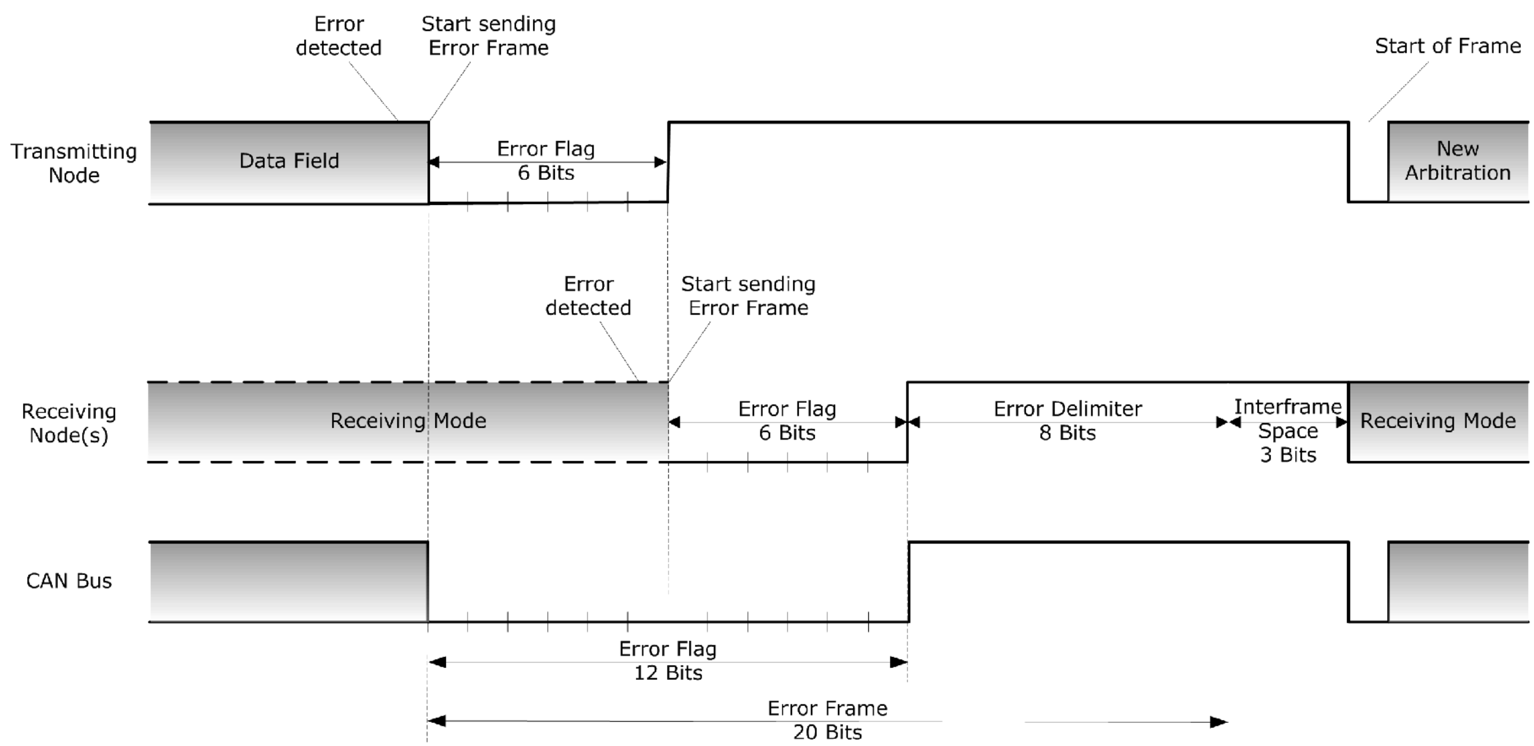


More “realistic” Error Frame



Error Recovery Time = Error Flag + Error Delimiter + Intermission Field = 12 + 8 + 3 = 23 Bits

Error Frame Super Positioning



Fault Confinement - 1

- Guarantees proper network operation even in cases where malfunctioning nodes produce continuous error condition
- CAN error detection can pinpoint to “perpetrator”
- Distinction between temporary and permanent node failures
- Identification and removal (self-retirement) of malfunctioning nodes from the bus

Transmit/Receive Errors

Possible error scenarios in a CAN network:

1. Transmit Error

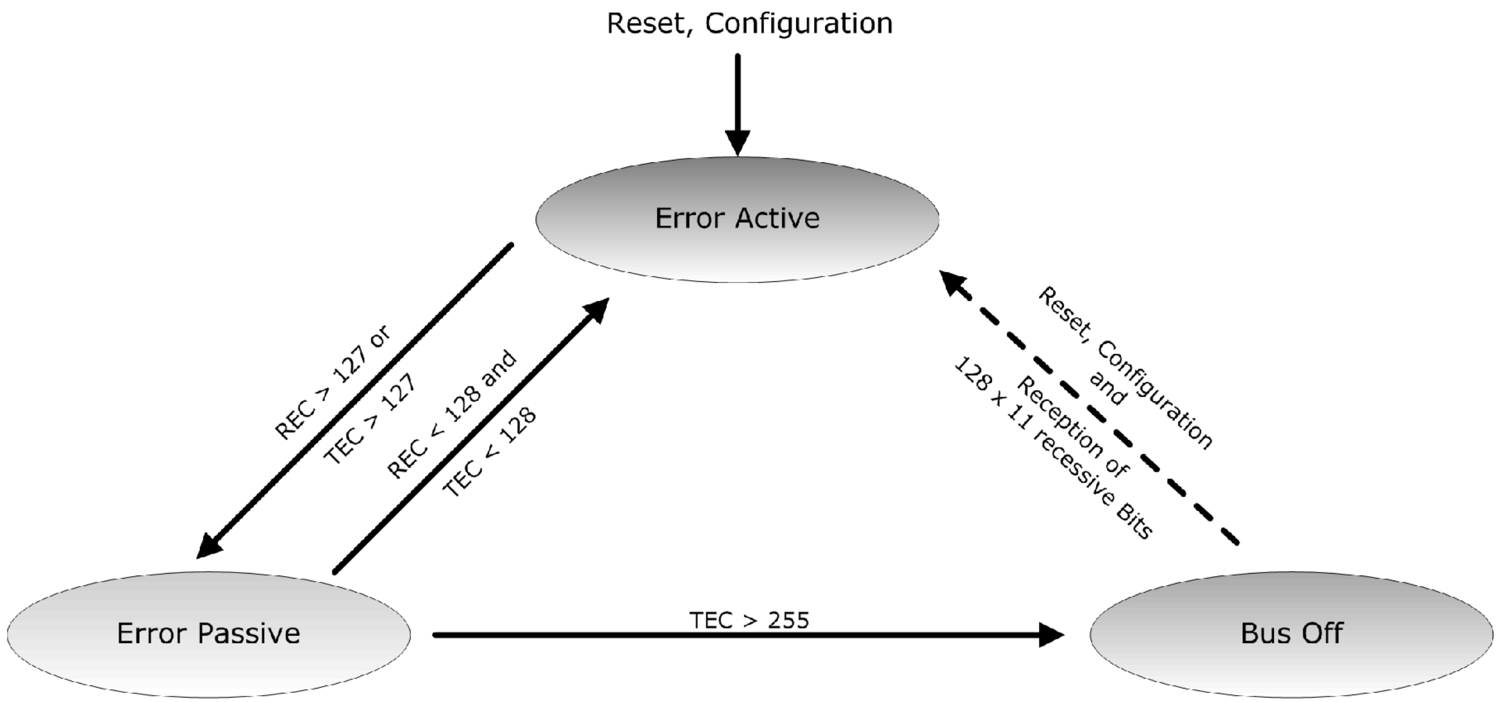
- A transmitting node sends a faulty message
- ALL receiving nodes in the network respond with an error frame
- Through "majority vote" the transmitting node is being flagged as the "perpetrator"

2. Receive Error

- A transmitting node send a perfectly good message
- Only ONE node in the network responds with an error frame
- Through "majority vote" the error reporting node is being flagged as the "perpetrator"

Fault Confinement - 2

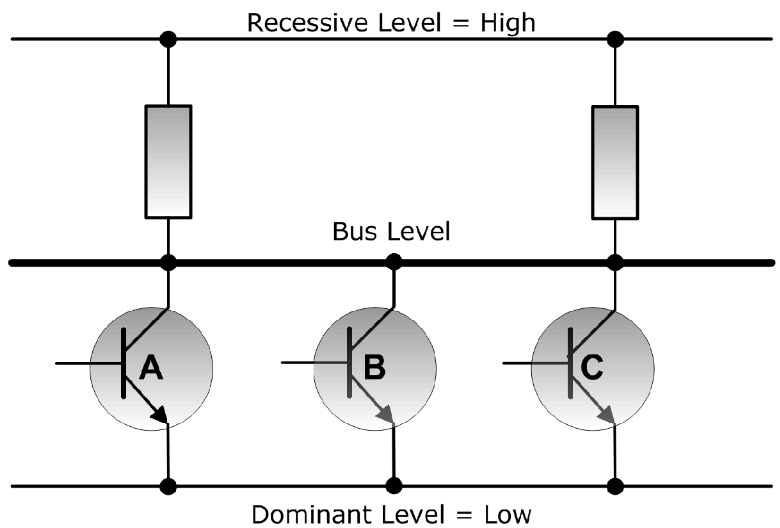
CAN Node Error States



REC: Receive Error Counter
TEC: Transmit Error Counter

Bus Medium

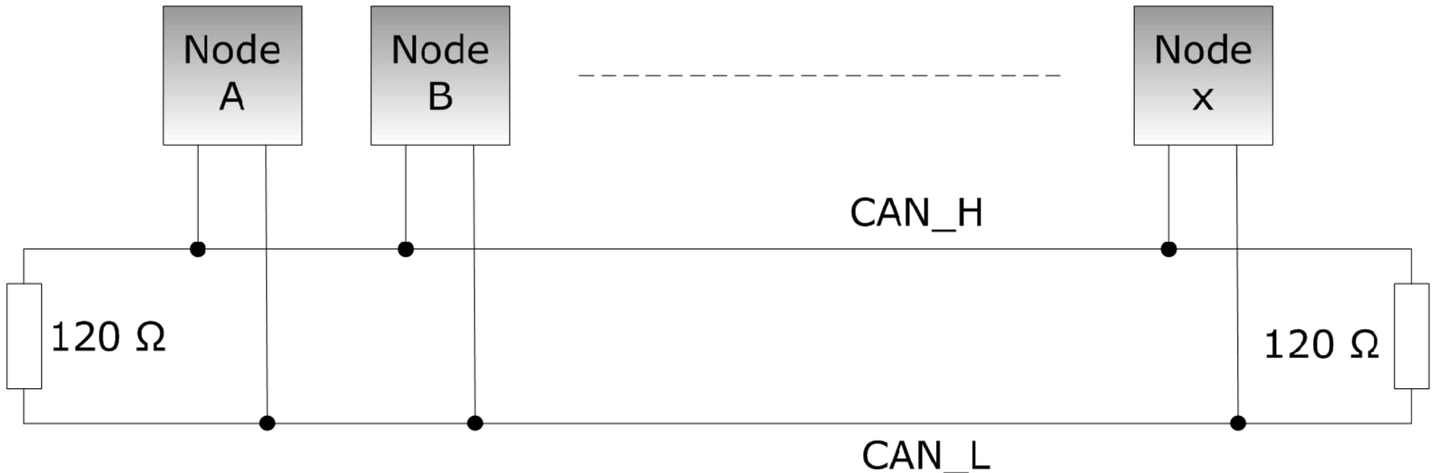
- Physical media must support “dominant” and “recessive” bus level. Dominant level always overrules recessive level, especially during bus arbitration.



- Two-wire bus terminated with line impedance to avoid signal reflections

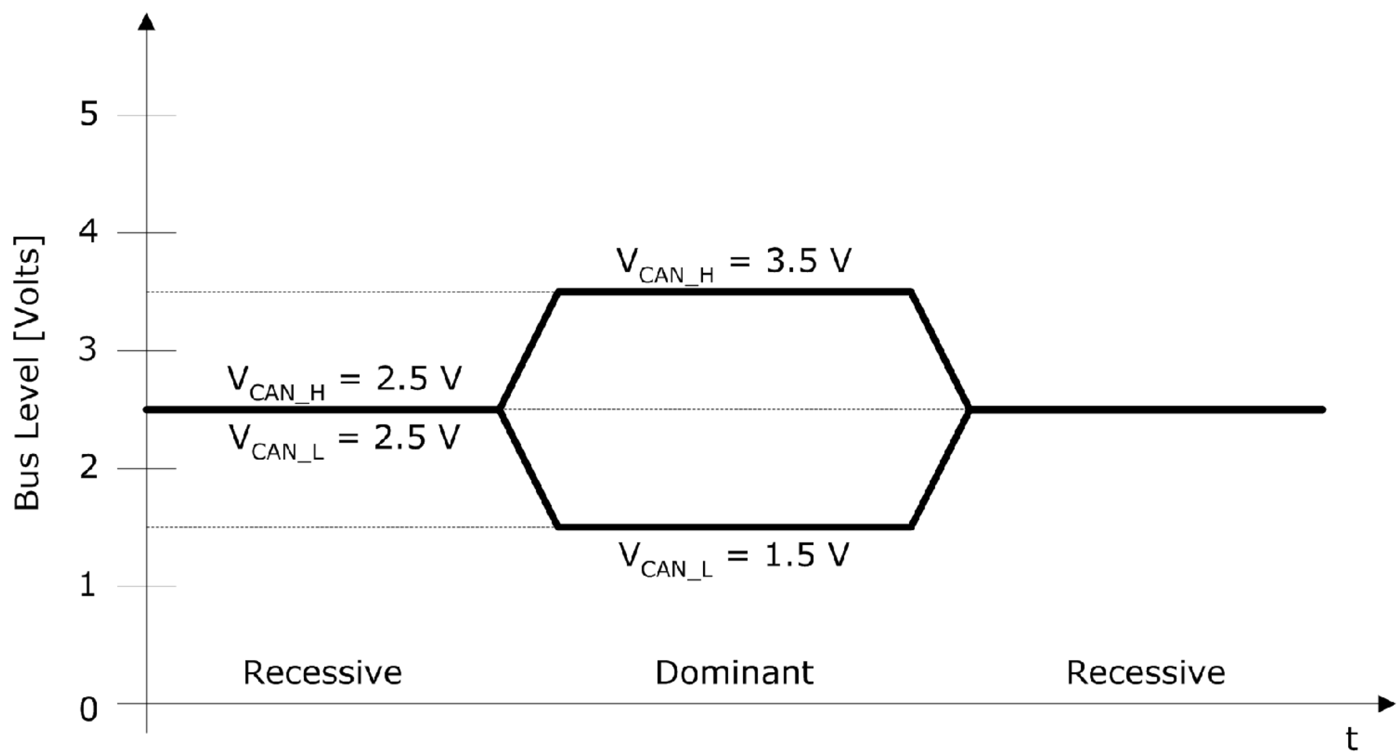
Bus Topology

Bus Topology according to ISO 11898



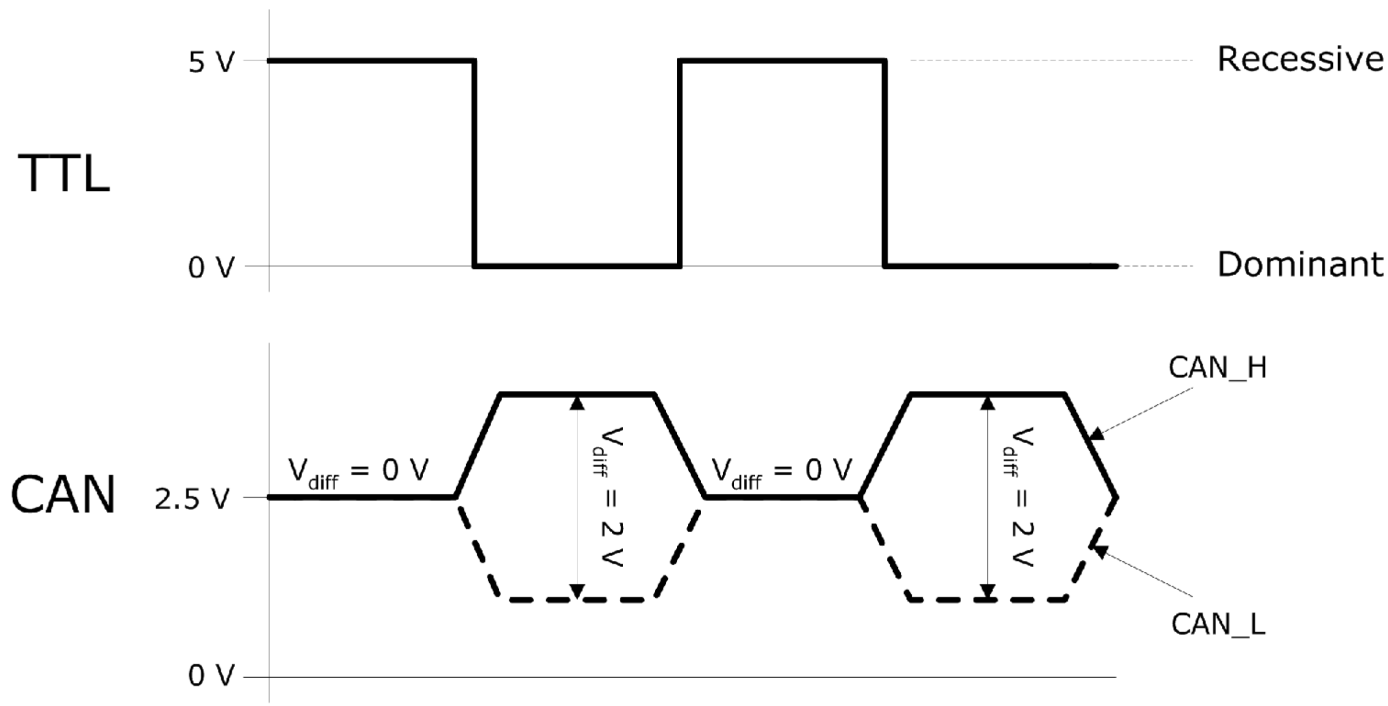
Bus Level - 1

Bus Levels according to ISO 11898

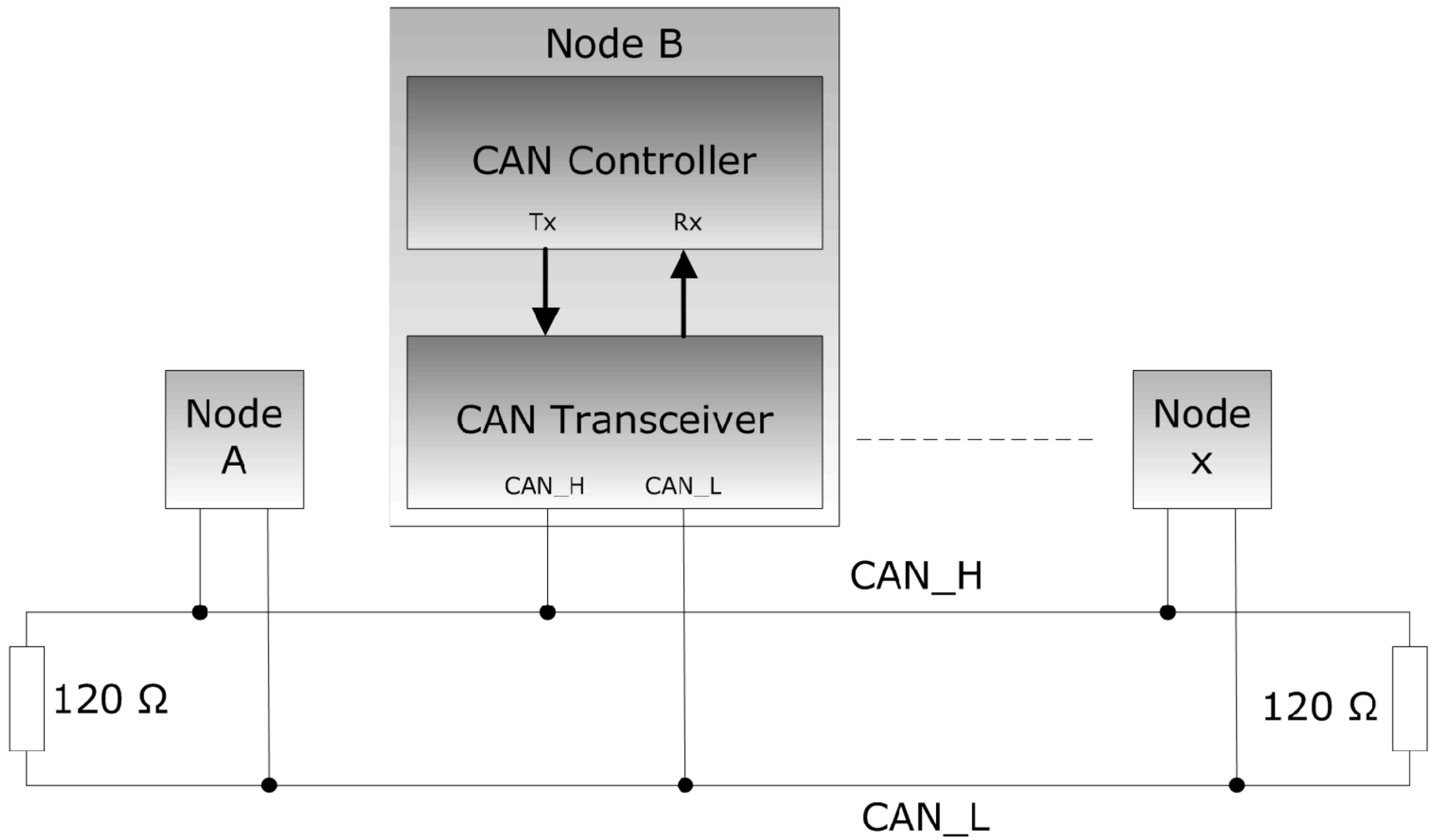


Bus Level - 2

Bus Levels according to ISO 11898

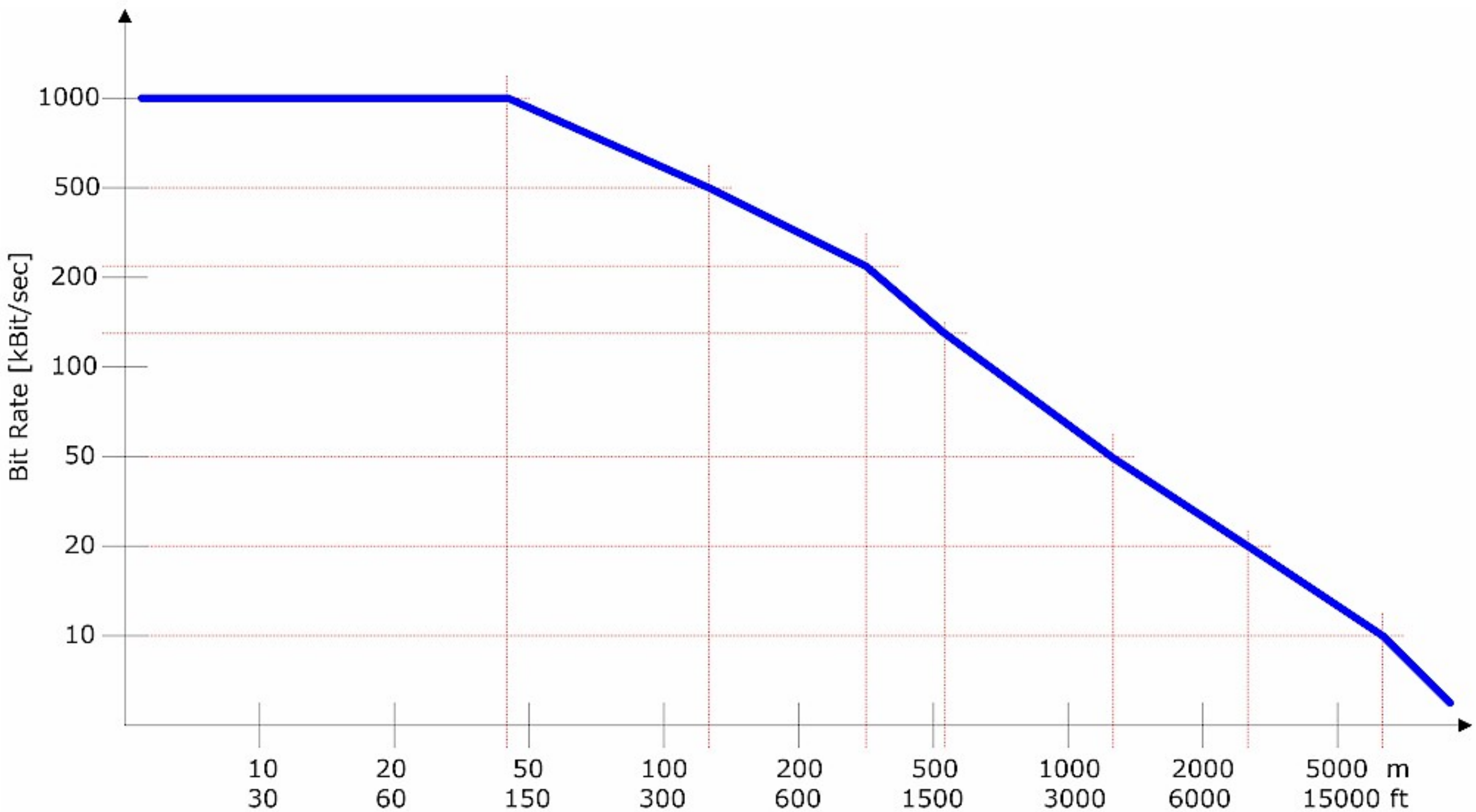


Bus Connection

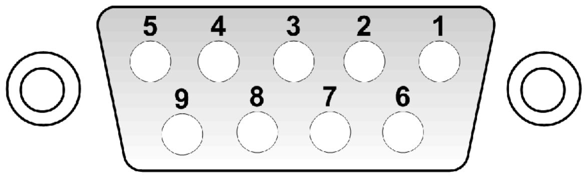


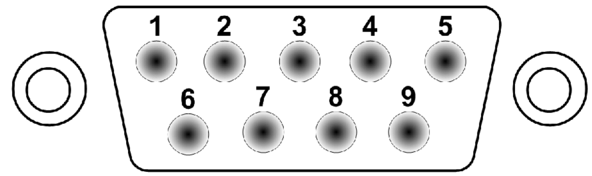
Maximum Bus Length

- **Bus Length is limited due to Bit Monitoring (Signal Propagation Time)**



Wiring and Connections





Pin	Signal	Description
1	-	Reserved
2	CAN_L	CAN_L bus line (dominant low)
3	CAN_GND	CAN Ground
4	-	Reserved
5	CAN_SHLD	Optional CAN shield
6	GND	Optional CAN Ground
7	CAN_H	CAN_H bus line (dominant high)
8	-	Reserved (error line)
9	CAN_V+	Optional CAN external positive supply

CAN Controller Chips - 1

Two different types of CAN applications:

- Stand-Alone CAN Controller
- Microprocessor with integrated CAN Controller

Many major semiconductor manufacturers, such as Motorola, Philips, Intel, Infineon, and many more, sell CAN chips.

Most semiconductor manufacturers who usually integrated a UART with their microprocessor design, in order to support serial communication for RS 232/485, nowadays tend to integrate CAN instead.

CAN Controller Chips - 2

Basic CAN

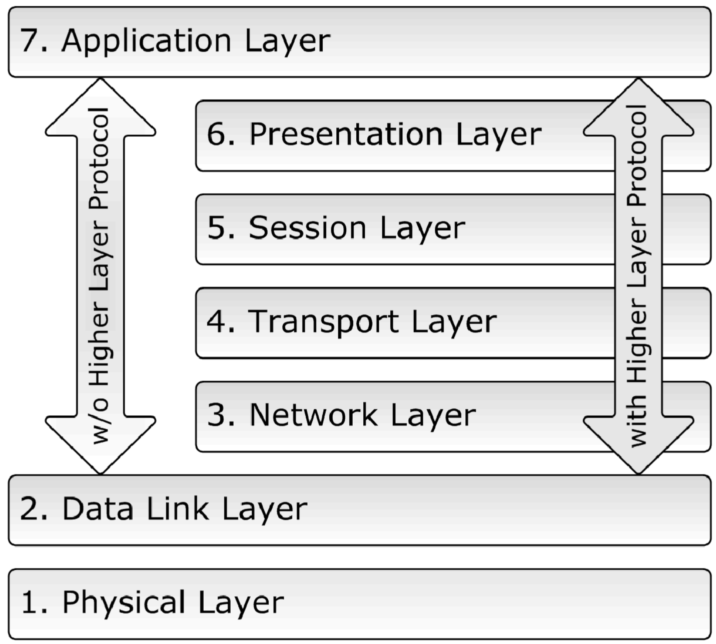
- One receive, one transmit message FIFO buffer
- Low cost solution
- Requires good CPU performance or low CAN data traffic

Full CAN

- Several programmable receive and/or transmit message buffers
- Most designs also provide Basic CAN features
- Allows low CPU performance or high CAN data traffic

Higher Layer Protocols - 1

Why Higher Layer Protocols...?



- Data Transport of more than 8 bytes
- Embedded Systems require appropriate communication model based on Master/Slave configuration
- Network Management (Network Start-Up, Node Monitoring, Node Synchronization, etc.)

Controller Area Network
Serial Network Technology for Embedded Solutions

Higher Layer Protocols - 2

CANopen

- Suited for embedded applications
- Was originally designed for motion control
- Developed/Maintained by CAN-in-Automation User Group
- Manufacturer-Independent Protocol

http://www.can-cia.org

DeviceNet

- Suited for industrial applications (floor automation)
- Developed by Allen Bradley/Rockwell
- Maintained by Open DeviceNet Vendor Association (ODVA)
- Standard “controlled” by Allen Bradley/Rockwell

http://www.odva.org

SAE J 1939

- Communication for vehicle networks (trucks, buses, etc.)
- Standard developed by Society of Automotive Engineers (SAE)

http://www.sae.org

More Info on CAN

CAN-in-Automation (CiA)

http://www.can-cia.org

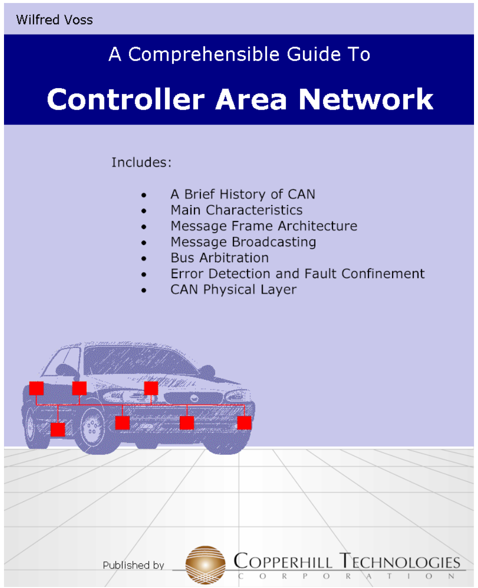
Open DeviceNet Vendor Association (ODVA)

http://www.odva.org

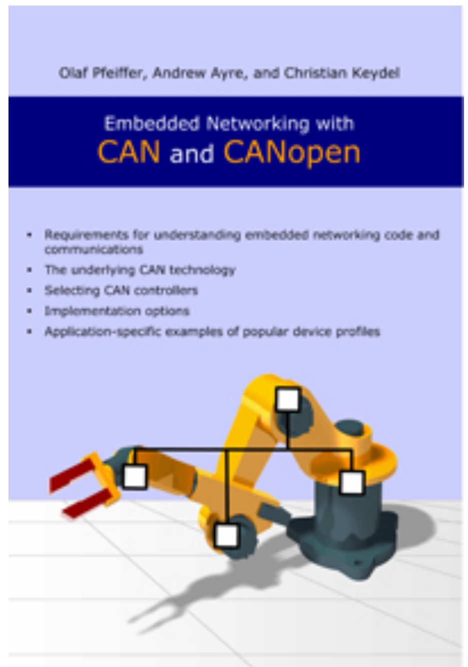
Society of Automotive Engineers (SAE)

http://www.sae.org

Literature



Literature on
Controller Area Network,
CANopen and SAE J1939



https://copperhilltech.com/technical-literature/