13
Controller Area Network (CAN)

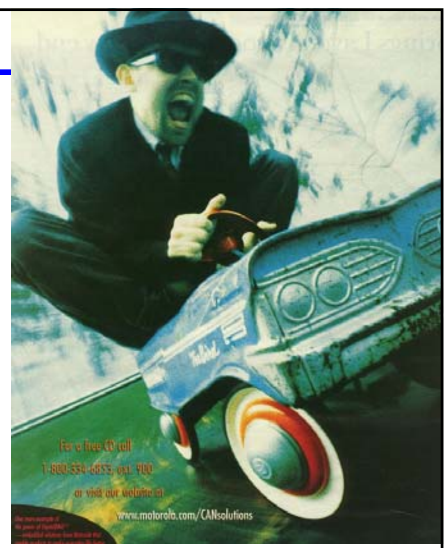
Distributed Embedded Systems
Philip Koopman
October 8, 2014

Significant material (CAN pictures) drawn from
a presentation by Siemens Corp. "CANPRES 2.0, Oct 1998"

Carnegie Mellon

Where Are We Now?

- **Where we've been:**
  - Protocol Overview
- **Where we're going today:**
  - CAN -- an important embedded protocol
  - Primarily automotive, but used in many places
- **Where we're going next:**
  - CAN performance
  - Other protocols
- **REMINDER – look at lessons learned slides for ideas on how to do better on second half of project!**



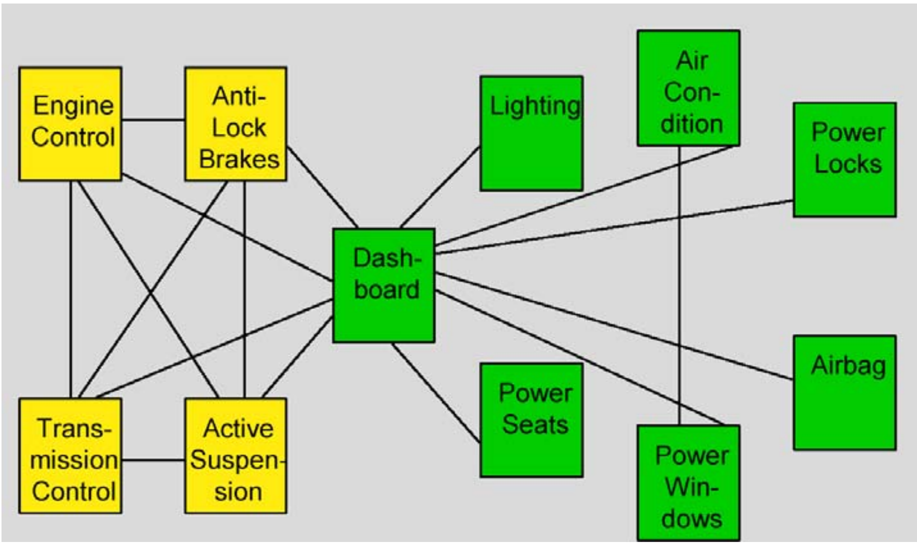
2

Preview

- **CAN – important automotive protocol**
  - Physical layer – built on bit dominance
  - Protocol layer – binary countdown
  - Message filtering layer (with add-on protocols)

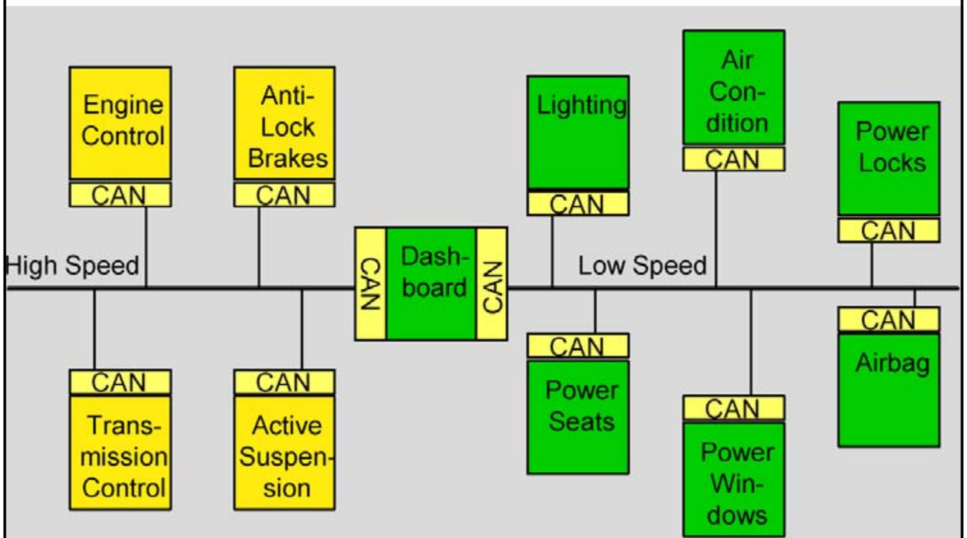
- **Keep an eye out for:**
  - Message prioritization
  - How “small” nodes can be kept from overloading with received messages
  - Tradeoffs

Before CAN



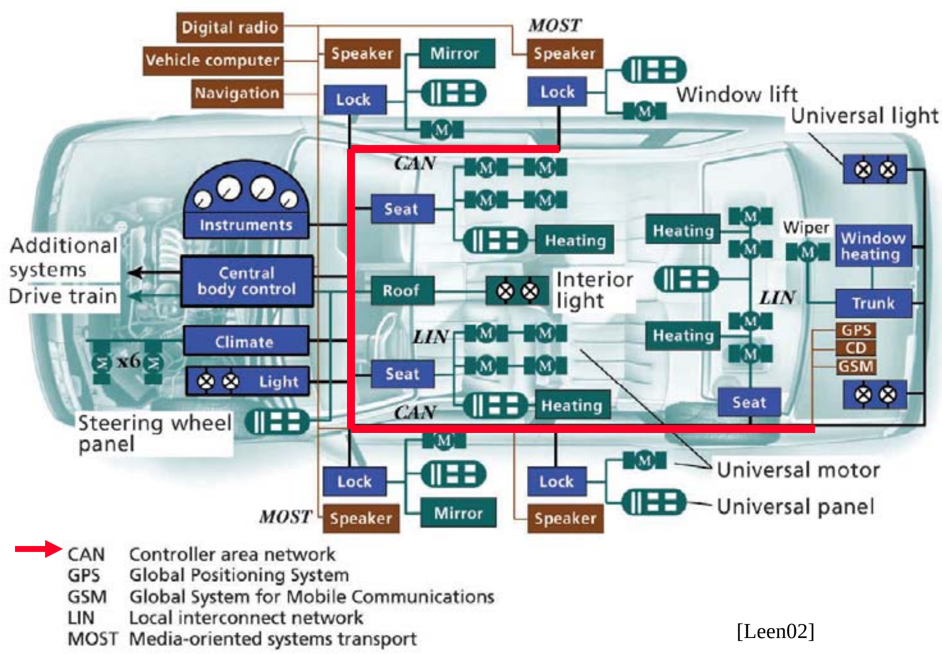
[Siemens]

With CAN



[Siemens] 5

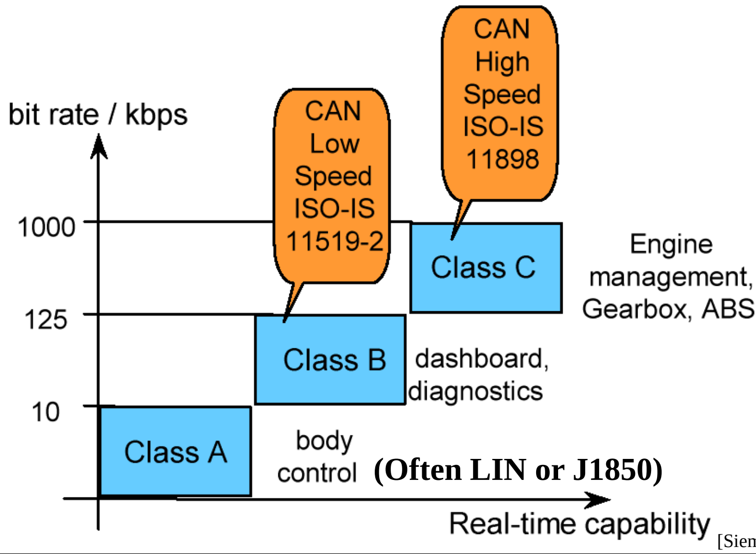
CAN Is Central To Automotive Networks



[Leen02]

SAE Message Classes

- **Fast tends to correlate with critical control**
  - But, this is not always true; just often true



(Often LIN or J1850)

[Siemens] 7

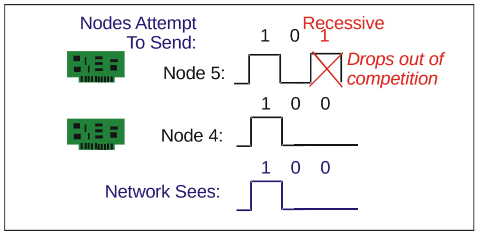
CAN & the Protocol Layers

- **CAN only standardizes the lower layers**
- **Other high-level protocols are used for application layer**
  - User defined
  - Other standards
  - We'll see one possibility at the end of this lecture

Application Layer
Object Layer - Message Filtering - Message and Status Handling
Transfer Layer - Fault Confinement - Error Detection and Signalling - Message Validation - Acknowledgment - Arbitration - Message Framing - Transfer Rate and Timing
Physical Layer - Signal Level and Bit Representation - Transmission Medium

[Siemens] 8

Remember This? Binary Countdown

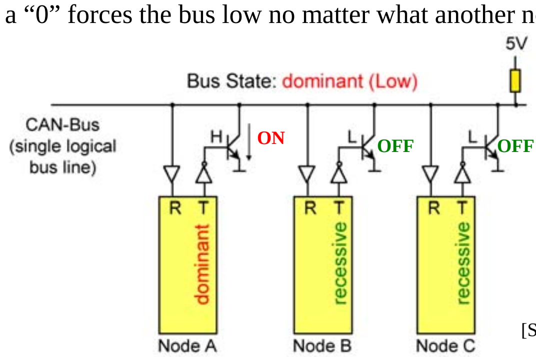


- **Operation**
  - Each node is assigned a unique identification number
  - All nodes wishing to transmit compete for the channel by transmitting a binary signal based on their identification value
  - A node drops out the competition if it detects a dominant state while transmitting a passive state
  - Thus, the node with the lowest identification value wins
- **Examples**
  - CAN – 500 Kbps or 1 Mbps
  - SAE J1850 – pretty much same as CAN, except slower (around 10 Kbps)

9

CAN – Bit Dominance In More Detail

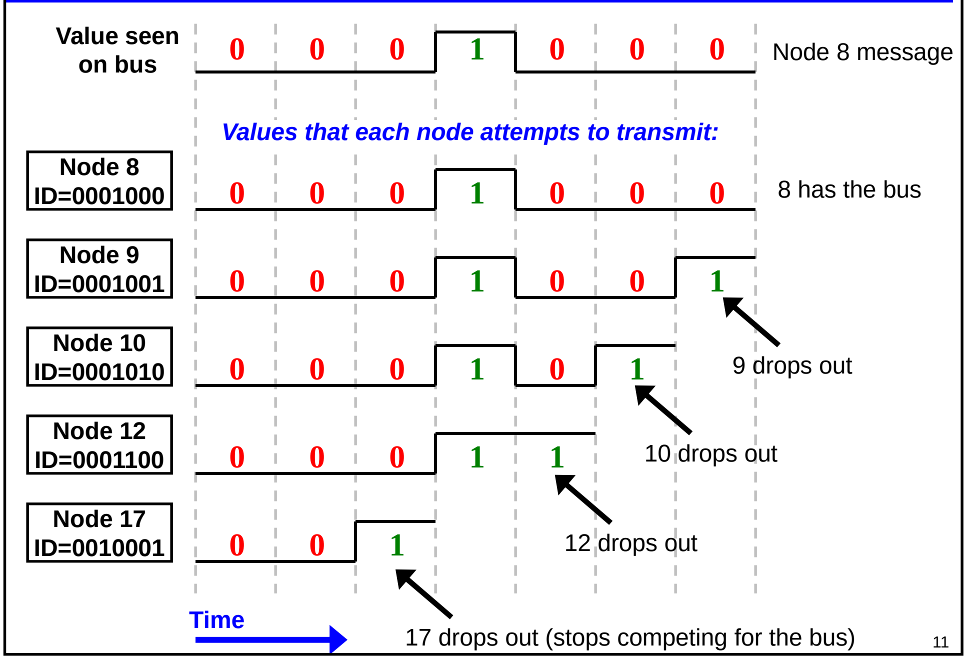
- **CAN uses the idea of recessive and dominant bits**
  - Wired "OR" design
  - Bus floats high unless a transmitter pulls it down (dominant)
  - (Other bus wire in differential transmission floats low and transmitter pulls up)
- **High is "recessive" value**
  - Sending a "1" can't override the value seen on the bus
- **Low is "dominant" value**
  - Sending a "0" forces the bus low no matter what another node is sending



[Siemens]

10

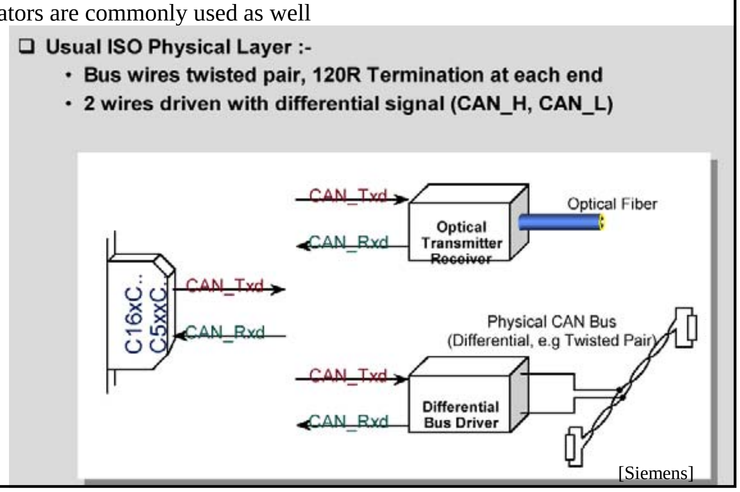
Example: Binary Countdown (highest bit first)



11

Physical Layer Possibilities

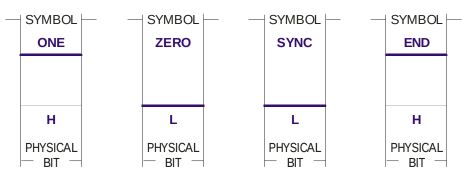
- **MUST support bit dominance**
  - Specifically rules out transformer coupling for high-noise applications
  - Differential driver used
    - Voltage across wires is dominant; high impedance (0V differential) is recessive
    - Opto-isolators are commonly used as well



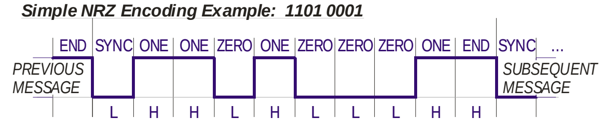
Non-Return to Zero (NRZ) Encoding

- **Send a Zero as LO;   send One as HI**
  - Worst case can have all zero or all one in a message – no edges in data
  - Simplest solution is to limit data length to perhaps 8 bits
    - SYNC and END are opposite values, guaranteeing two edges per message
    - This is the technique commonly used on computer serial ports / UARTs
  - Bandwidth is one edge per bit
    - Same bandwidth as Miller encoding, but no guarantee of frequent edges

Simple NRZ Bit Encoding



Simple NRZ Encoding Example: 1101 0001

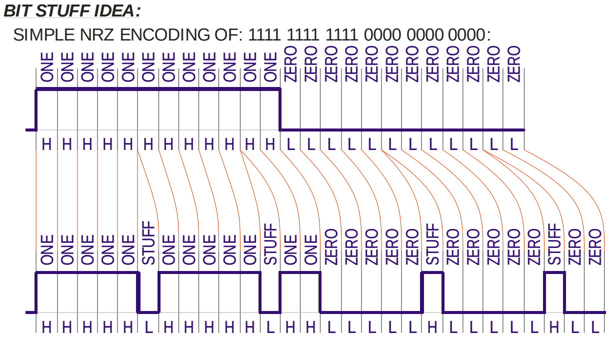


13

Bit Stuffing To Add Edges To NRZ Encoding

- **Long NRZ messages cause problems in receivers**
  - Clock drift means that if there are no edges, receivers lose track of bits
  - Periodic edges allow receiver to resynchronize to sender clock
- **Solution: add "stuff bits"**
  - Stuff bits are extra bits added to force transitions regardless of data
  - Typical approach: add an opposite-valued stuff bit after every 5 identical bits
  - In best case you don't need stuff bits – they only are needed for runs of values

BIT STUFF IDEA:

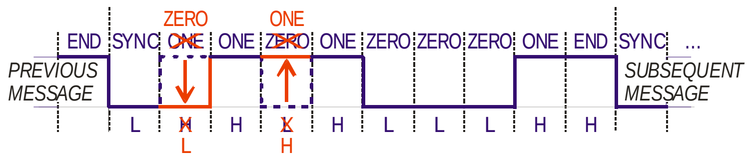


BIT-STUFFED NRZ ENCODING OF: 1111 1111 1111 0000 0000 0000:

14

NRZ Encoding Error Susceptibility

- **A single inverted physical bit is undetectable with Simple NRZ**
  - High efficiency comes at price of poor error detection



  - (Can be detected via CRC sometimes; but CRCs have limitations)

- **Bit stuffing error detection in general case:**
  - Improves error detection if stuffing rule is violated
  - Any six identical data bits in a row is an stuffing error
  - But, there is a subtle problem with bit stuffing…

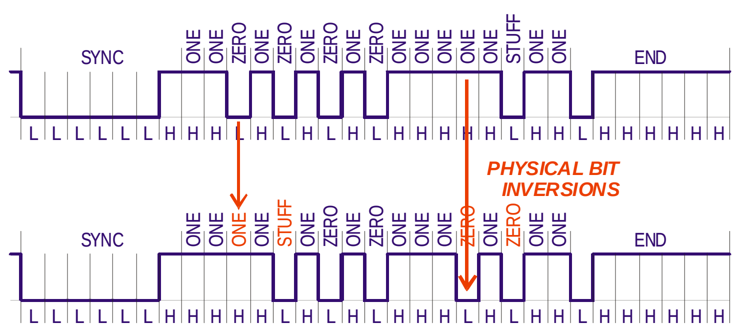
15

Cascaded Bit Stuffing Errors

- **Bit inversions in just the wrong place can confuse bit stuffing logic**
  - Worst errors occur in pairs that create and then break runs of bits
  - Data bit is converted to stuff bit; stuff bit to data bit
  - Net effect is same message length BUT, it shifts intervening data bits
  - CAN has this problem; can cause 2-bit error to escape CRC detection!

Cascaded bit stuff error example:

TRANSMITTED LOGICAL BITS: 1101 0101 0111 1111



RECEIVED LOGICAL BITS: 1111 1010 1110 1011

16

General CAN Message Format

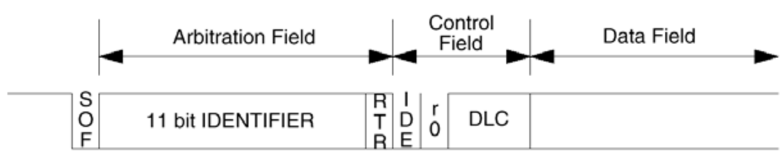
SYNC	HEADER	DATA	ERROR DETECTION	END

- **Header**
  - Application can set any desired value in 11- or 29-bit header
  - Global priority information (which message gets on bus first?)
  - Header often contains source, destination, and message ID
- **Data**
  - Application- or high-level-standard defined data fields
  - 0 to 8 bytes of data for CAN
- **Error detection**
  - Detects corrupted data (uses a 15-bit CRC):
    - All 15-bit or shorter burst errors (groups of flipped bits clumped together)
    - All 5-bit errors regardless of where they occur …
      … except bit stuffing problem reduces this to all 1-bit errors

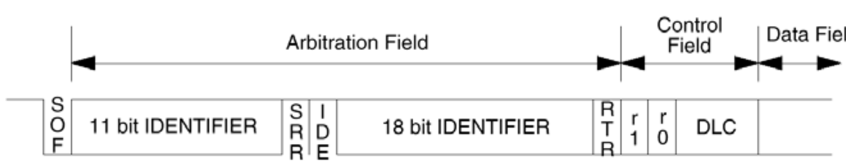
17

Two Sizes of CAN Arbitration Fields

Standard Format



Extended Format



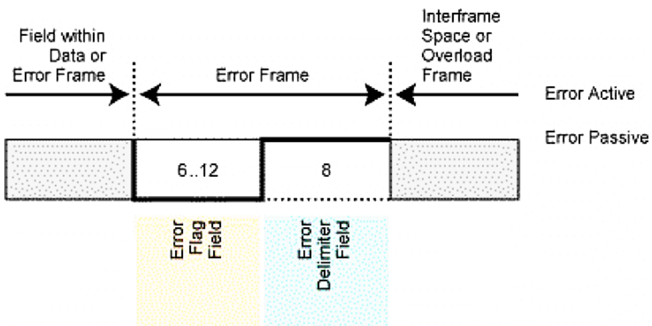
[Bosch] 18

CAN Message Fields

- **SOF – Start of frame (SYNC symbol)**
  - Single dominant bit
- **Arbitration field – binary countdown priority value; set by application**
  - Also an RTR (remote transmission) field for atomic transactions; seldom used
  - SRR is a dummy bit to let standard format RTR messages win arbitration
- **Control field**
  - 4-bit data length (number of bytes in data field); valid values: 0 .. 8
  - 1 bit specifies standard or extended format; 1 bit unused
- **Data field**
  - 0 to 8 bytes
- **CRC field**
  - 15-bit CRC, followed by one recessive delimiter bit
- **Ack field**
  - If message received OK, assert as dominant bit (**at least one node** received)
- **END of frame delimiter**
  - Seven recessive bits mark end of frame (phase violation for bit stuff pattern)

Error Frame Messages

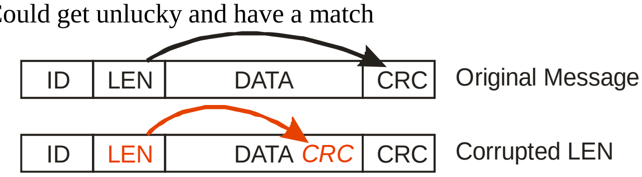
- **Error frame alerts transmitter if message garbled at some receivers**
  - Sent if bit stuff violation detected or CRC error detected
  - Error flag is six dominant bits in a row – guaranteed to violate bit stuffing rules
    - (Unless the Error Frame itself suffers a bit error)
  - If transmitter sees that it has been pre-empted by an error frame, it attempts retransmission
    - Note – this is a source of nondeterminism in protocol – timing varies depending on errors encountered!



[Siemens]

CAN vs. FlexRay Length Field Corruptions

- **CAN does not protect length field against ONE-BIT errors**
  - Corrupted length field will point to wrong location for CRC!
    - One bit error in length field circumvents HD=6 CRC
    - Could get unlucky and have a match



- **FlexRay solves this with a header CRC to protect Length**

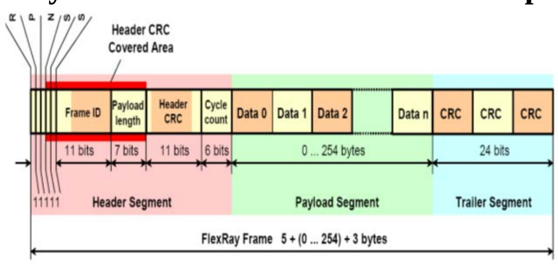


Figure 4-1: FlexRay frame format.

Source: FlexRay Standard, 2004

21

Other CAN Issues

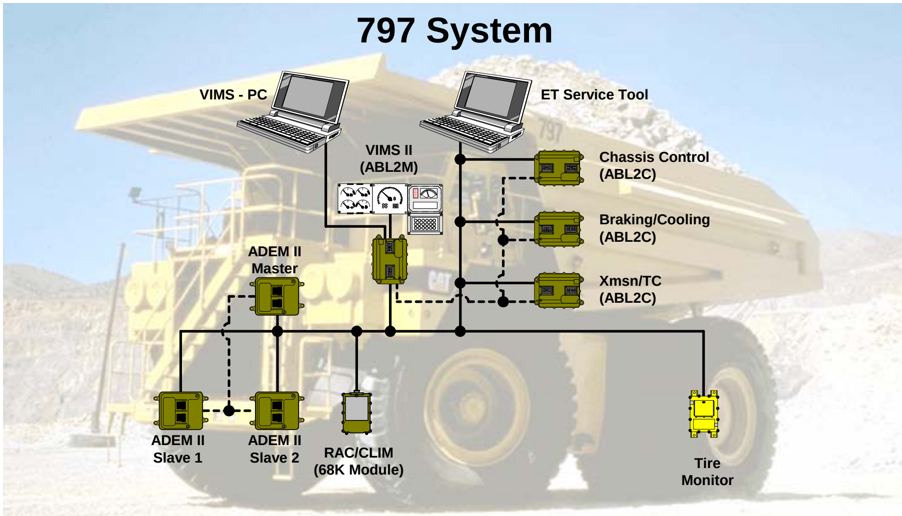
http://betterembsw.blogspot.com/2012/02/can-protocol-vulnerabilities.html

- **CAN advertises "exactly once" delivery semantics**
  - A message corrupted only in some places is retransmitted… … and received more than once by some nodes

  [Rufino 1998]
- **"Stuck at zero" (dominant) transmitter output locks up network**
- **CAN retry:**
  - "Error frame" scheme causes re-transmit
    and ALSO,
  - Node monitors network and looks for data sent == data received
    - If no match, assumes corruption and tries again
  - But what if the transmitter is what is broken?
    - CAN node can lock up the network with retries [Perez 2003]
- **In general, CAN unsuitable for highly critical applications**
  - That's one reason we have FlexRay (and TTP) protocols
  - This is news to some embedded folks (e.g., ARINC 825 aviation standard)

22

CAN (SAE J1939) Example: Caterpillar 797

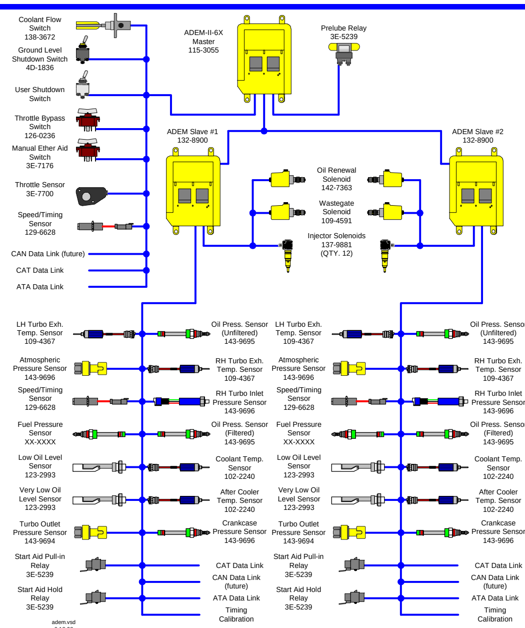


+ 195 sensors and actuators

+ wireless data link

[Slide courtesy of Caterpillar Inc.] 23

ADEM II Engine Control



[Slide courtesy of Caterpillar Inc.] 24

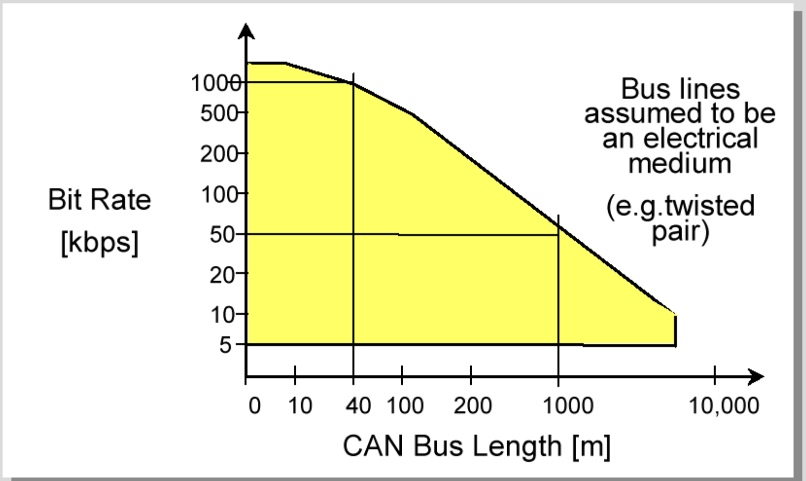
More Than Just Vehicles

Liebert Modular UPS (uses CAN)

Arbitration Limits Network Size

- **Need 2*t_{pd} per bit maximum speed**

❑ **Up to 1Mbit / sec @40m bus length (130 feet)**



[Siemens] 26

"Big" & "Small" Nodes

- **Some nodes can handle a lot of messages**
  - Many message mailboxes/filters
  - Fast processor

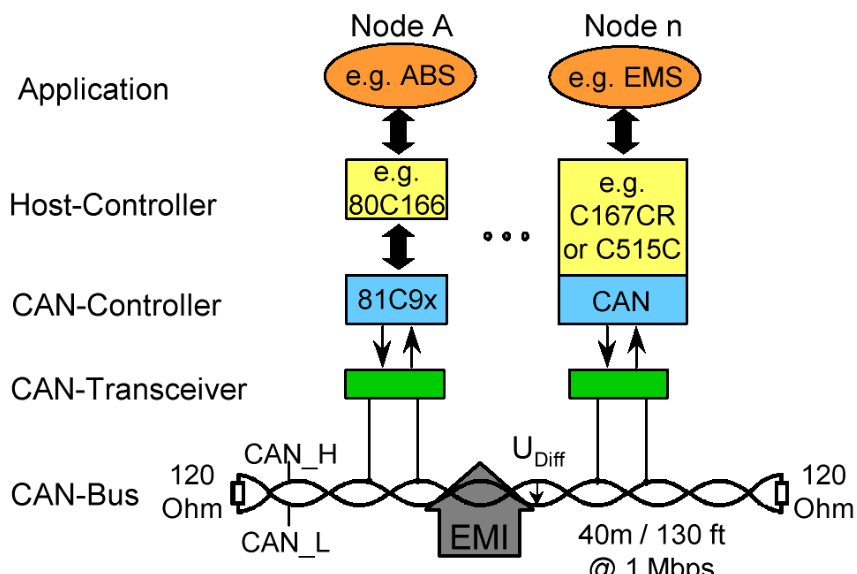
- **Some small nodes have limited capacity**
  - One or two mailboxes/filters
  - Slow processor

- **System designer has to prevent message over-run via one of:**
  - Dedicated mailbox per message (hardware ensures no data lost)
  - If mailbox shared, ensure messages to slow processors are spaced apart
    - Must be infrequent
    - Must ***ALSO*** not be clumped closer than receiver response time
    - This ends up being a constraint for real time scheduling (a later lecture)

27

Generic CAN Network Implementation

- **Signals usually sent differentially – CAN_H and CAN_L**



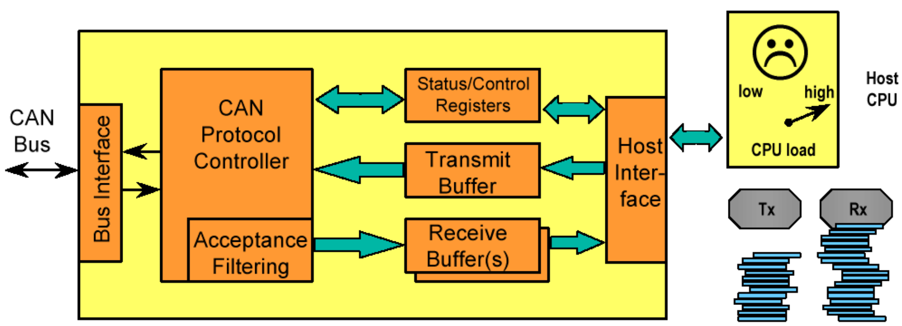
[Siemens] 28

Example CAN Microcontrollers

- **Motorola 68HC05 Family**
  - 11-bit headers; 1 Tx buffers; 2 Rx message buffers; 8-bit accept mask
  - 8-bit CPU; up to 32 KB on-chip ROM; 28- or 64-pin housing
  - (Also 68HC08 with 29-bit support and more buffers)
- **Motorola 68HC912 Family**
  - 11- & 29-bit headers; 3 Tx buffers; 2 Rx message buffers; 2 accept masks
  - 16-bit CPU; up to 128 KB on-chip Flash; 80- or 112-pin housing
- **Motorola 6837X Family**
  - 11- & 29-bit headers; 16 Tx/Rx buffers; 16 accept masks
  - 32-bit CPU; 256 KB on-chip Flash
- **Many other companies support CAN of course – these are just examples**

Basic CAN Controller (avoid this one if possible)

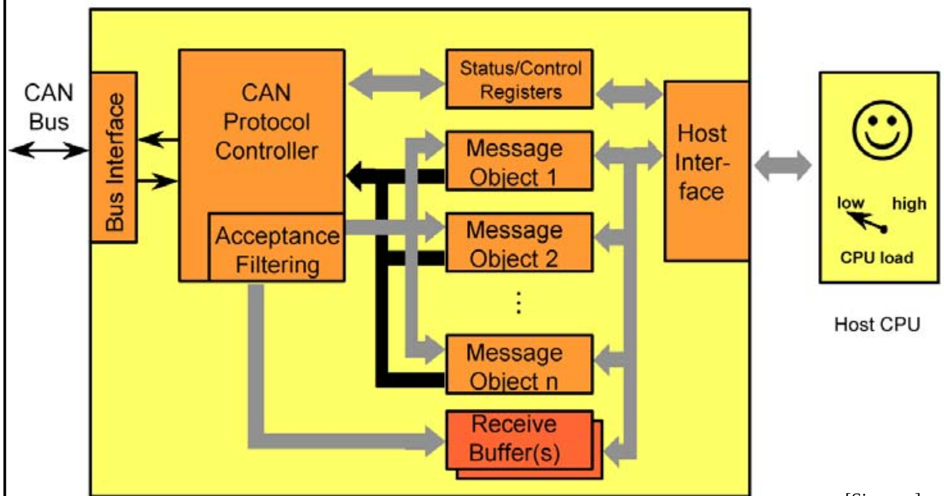
- **"Cheap" node**
  - Could get over-run with messages even if it didn't need them



[Siemens]

Full CAN Controller

- **Hardware message filters sort & filter messages without interrupting CPU**
  - Message object holds most recent message fo that type – not a queue!



[Siemens]

Mask Registers

- **Used to set up message filters**
  - Mask register selects bits to examine
  - Object Arbitration register selects bits that must match to be accepted
  - Map multiple messages into each message object “mailbox”

	10										0	
Mask Register (std ID)	1	1	1	1	1	1	1	0	1	1	0	(= 0x7f6)
Message Object Arbitration Register	1	0	0	1	0	0	1	0	0	0	0	(= 0x490)
Resulting ID matches (X = don't care)	1	0	0	1	0	0	1	X	0	0	X	
ID's received:	1	0	0	1	0	0	1	0	0	0	0	(= 0x490)
	1	0	0	1	0	0	1	0	0	0	1	(= 0x491)
	1	0	0	1	0	0	1	1	0	0	0	(= 0x498)
	1	0	0	1	0	0	1	1	0	0	1	(= 0x499)

[Siemens] 32

Mask Register Example

- **Mask Register:** **1 1 0 1 1 1 0 1 0 1 1**
- **Message Object Arbitration:** **1 0 0 0 1 0 0 0 0 0 1**
- **Effective Match Value:** **1 0 * 0 1 0 * 0 * 0 1**

- **Matches these message IDs:**

1 0 0	0 1 0 0	0 0 0 1
1 0 0	0 1 0 0	0 1 0 1
1 0 0	0 1 0 1	0 0 0 1
1 0 0	0 1 0 1	0 1 0 1
1 0 1	0 1 0 0	0 0 0 1
1 0 1	0 1 0 0	0 1 0 1
1 0 1	0 1 0 1	0 0 0 1
1 0 1	0 1 0 1	0 1 0 1

- **More likely, you mask a few bits next to each other**
  - See DeviceNet later in lecture

33

DeviceNet

- **One of several higher-level protocols**
  - Based on top of CAN
  - Used for industrial control (valves, motor starters, display panels, …)
    - Caterpillar is a member of ODVA as well (Open DeviceNet Vendors Assn.), but for factory automation.
- **Basic ideas:**
  - CAN is used in high volumes = cheaper network chips than competitors
  - Use structured approach to message formats to standardize operation
- **Does *NOT* standardize specific message contents**
  - But it does specify a hierarchy of message ID formats

34

DeviceNet Message ID Scheme

- **Each node on network “owns” a source node or message ID (or both)**

Message Identifier Bits

10	9	8	7	6	5	4	3	2	1	0	**Hex Range**	**Identity Usage**
0	Message ID				Source Node #						000 - 3ff	Group 1
1	0	Source Node #						Msg ID			400 - 5ff	Group 2
1	1	Msg ID (0..6)			Source Node #						600 - 7bf	Group 3
1	1	1	1	1	Message ID (0..2f)						7c0 - 7ef	Group 4
1	1	1	1	1	1	1	X	X	X	X	7f0 - 7ff	Invalid

- **Use message filters to only listen to messages you care about**
  - E.g., Use message object arbitration to subscribe to a particular message ID
  - E.g., Use mask object to accept that message ID from any source node #
  - Elevator example: message ID is button press; source node # tells which button
    - Single receiver mailbox then holds most recently received button press message
    - Message must be processed before next such message is received!

35

DeviceNet Group Strategy

- **Group 1**
  - Prioritized by Message ID / Node number
  - High priority messages with fairness to nodes
- **Group 2**
  - Prioritized by Node number / Message ID
  - Gives nodes priority
- **Group 3**
  - Essentially same as Group 1, but allows Group 2 to have higher priority
- **Group 4**
  - Global housekeeping messages / must be unique in system (no node number)

36

Other Approaches Are Possible

- **And, you can invent your own too…**
- **Variations include:**
  - Automatic assignment of node numbers (include hot-swap)
  - Automatic assignment of message numbers (include hot-swap)
  - Mixes of node-based vs. message-ID based headers
- **Can you have two transmitters using the same exact header field?**
  - No – that would produce a bus conflict
  - Unless you have middleware that ensures only one node can transmit at a time
    - For example use a low priority message as a token to emulate token-passing
- **Higher level protocols define message types**
  - For example, J1939 defines message ID meanings, mostly for trucks and buses

37

CAN Workloads – Spreadsheets

- **"SAE Standard Workload" (53 messages)** V/C = Vehicle Controller [Tindell]

Signal Number	*Signal Description*	*Size /bits*	*J /ms*	*T /ms*	*Periodic /Sporadic*	*D /ms*	*From*	*To*
1	Traction Battery Voltage	8	0.6	100.0	P	100.0	Battery	V/C
2	Traction Battery Current	8	0.7	100.0	P	100.0	Battery	V/C
3	Traction Battery Temp, Average	8	1.0	1000.0	P	1000.0	Battery	V/C
4	Auxiliary Battery Voltage	8	0.8	100.0	P	100.0	Battery	V/C
5	Traction Battery Temp, Max.	8	1.1	1000.0	P	1000.0	Battery	V/C
6	Auxiliary Battery Current	8	0.9	100.0	P	100.0	Battery	V/C
7	Accelerator Position	8	0.1	5.0	P	5.0	Driver	V/C
8	Brake Pressure, Master Cylinder	8	0.1	5.0	P	5.0	Brakes	V/C
9	Brake Pressure, Line	8	0.2	5.0	P	5.0	Brakes	V/C
10	Transaxle Lubrication Pressure	8	0.2	100.0	P	100.0	Trans	V/C
11	Transaction Clutch Line Pressure	8	0.1	5.0	P	5.0	Trans	V/C
12	Vehicle Speed	8	0.4	100.0	P	100.0	Brakes	V/C
13	Traction Battery Ground Fault	1	1.2	1000.0	P	1000.0	Battery	V/C
14	Hi&Lo Contactor Open/Close	4	0.1	50.0	S	5.0	Battery	V/C
15	Key Switch Run	1	0.2	50.0	S	20.0	Driver	V/C
16	Key Switch Start	1	0.3	50.0	S	20.0	Driver	V/C
17	Accelerator Switch	2	0.4	50.0	S	20.0	Driver	V/C
18	Brake Switch	1	0.3	20.0	S	20.0	Brakes	V/C
19	Emergency Brake	1	0.5	50.0	S	20.0	Driver	V/C
20	Shift Lever (PRNDL)	3	0.6	50.0	S	20.0	Driver	V/C
21	Motor/Trans Over Temperature	2	0.3	1000.0	P	1000.0	Trans	V/C
22	Speed Control	3	0.7	50.0	S	20.0	Driver	V/C
23	12V Power Ack Vehicle Control	1	0.2	50.0	S	20.0	Battery	V/C
24	12V Power Ack Inverter	1	0.3	50.0	S	20.0	Battery	V/C
25	12V Power Ack I/M Contr.	1	0.4	50.0	S	20.0	Battery	V/C
26	Brake Mode (Parallel/Split)	1	0.8	50.0	S	20.0	Driver	V/C

38

CAN Tradeoffs

- **Advantages**
  - High throughput under light loads
  - Local and global prioritization possible
  - Arbitration is part of the message - low overhead

- **Disadvantages**
  - Requires bit dominance (can't be used with transformer coupling)
  - Propagation delay limits bus length (2 t_{pd} bit length)
  - Unfair access - node with a high priority can "hog" the network
    - Can be reduced in severity with Message + Node # prioritization
    - Can, in principle, use a bus guardian to limit duty cycle of each node
  - Poor latency for low priority nodes
    - Starvation is possible

- **Optimized for:**
  - Moderately large number of message types
  - Arbitration overhead is constant
  - Global prioritization (*but* limited mechanisms for fairness)

39

A COMPARISON OF VARIOUS AUTOMOBILE NETWORKING STANDARDS

Name	Protocol specification	Interface	Type	Speed (kbits/s)	Comment
J1850	Yes	1 wire	Control	41.6	Proprietary implementations.
CAN	Yes	No	Control	Variable	General protocol.
CAN-A	CAN	2 wire	Control	33, 83	Used in U.S.
CAN-B	CAN	2 wire	Control	250	Used in U.S.
CAN-C	CAN	2 wire	Control	1000	Used in U.S.
SAE J2284	CAN	2 wire	Control	500	Used for power-train control.
SAE J1939	CAN	2 wire	Control	125	Recommended Practice for Serial Control and Communications Vehicle Network Class C by the SAE Truck & Bus Control and Communications Network Subcommittee of the Truck & Bus Electrical Committee.
SAE J2411	CAN	1 wire	Control	24	Unique to General Motors.
LIN	Yes	1 wire ISO 9141	Control	20	Master/slave. Doesn't require crystal.
TTP/A	Yes	1 wire ISO 9141	Control	20	Master/slave. Supports hot plug-and-play. Also supports higher speeds and fiber.
TTP/C	Yes	2 wire	X-by-wire	2000	Higher speeds possible.
Flexray	Yes	2 x 2 wire	X-by-wire	5000	Developed by Phillips.
IDB-C	CAN	2 wire	Multimedia	250	Developed by the IDB Forum.
IEEE 1394	Yes	2 wire	Multimedia	300,000	Being adapted to the automotive environment.
Smartwire	Yes	2 wire	Multimedia	22,000	Ring topology.
MOST	Yes	Fiber	Multimedia	25,000	Multiple master. Up to 64 devices.

[Electronic Design; Jan 8, 2001, pg. 66]

40

Review

- **Controller Area Network**
  - Binary-countdown arbitration
  - Standard used in automotive & industrial control

- **CAN Tradeoffs**
  - Good at global priority (but difficult to be “fair”)
  - Efficient use of bandwidth
  - Requires bit-dominance in physical layer
  - Message filters are required to keep small nodes from being overloaded
    - Only works if small node can read data before next data in that mailbox arrives

41