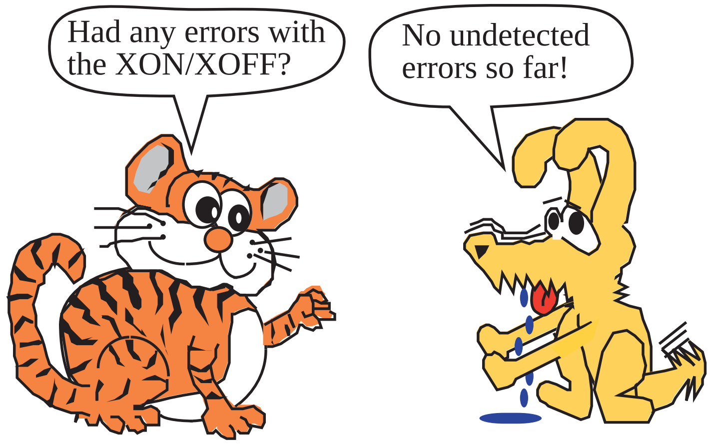


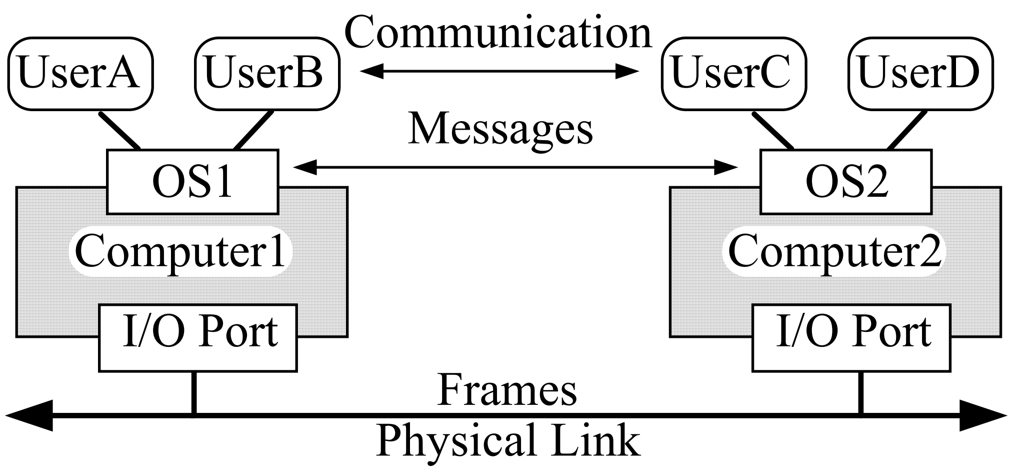
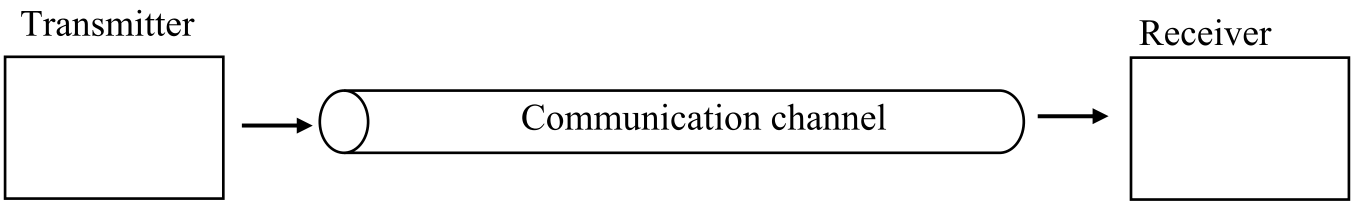




*Figure 14.1. A layered approach to communication systems.*

1) Address information field
   - physical address specifying the destination/source computers
   - logical address specifying the destination/source processes (e.g., users)
2) Synchronization or handshake field
   - Physical synchronization like shared clock, start and stop bits
   - OS synchronization like request connection or acknowledge
   - Process synchronization like semaphores
3) Data field
   - ASCII text (raw or compressed)
   - Binary (raw or compressed)
4) Error detection and correction field
   - Vertical and horizontal parity
   - Checksum
   - Logical redundancy check (LRC)
   - Block correction codes (BCC)

- The general transmission system depicted below:
  - employs some transmission medium that permits some form of energy to be carried from the *transmitter* to the *receiver*
  - the energy may vary continuously with time or transition between discrete values
  - ultimately, the energy is employed to represent *information*
    - audio, video, image, text, abstract, etc.




- The issues along the way:
  - attenuation
    - energy is lost to the medium and surroundings
  - distortion
    - channel treats signals differently based upon frequency, intensity (amplitude), etc.
  - noise
    - energy is combined with the signal to produce a new signal
- How rapidly can information (bits) be communicated via a particular transmission system?
- That depends upon:
  - the amount of *energy* used in transmitting each signal
  - the distance between transmitter and receiver – *attenuation* and *distortion*
  - the amount of *noise* associated with the channel
  - the *bandwidth* of the channel
- *Shannon channel capacity*:

- $C = W \log_2 (1 + SNR)$ b/s

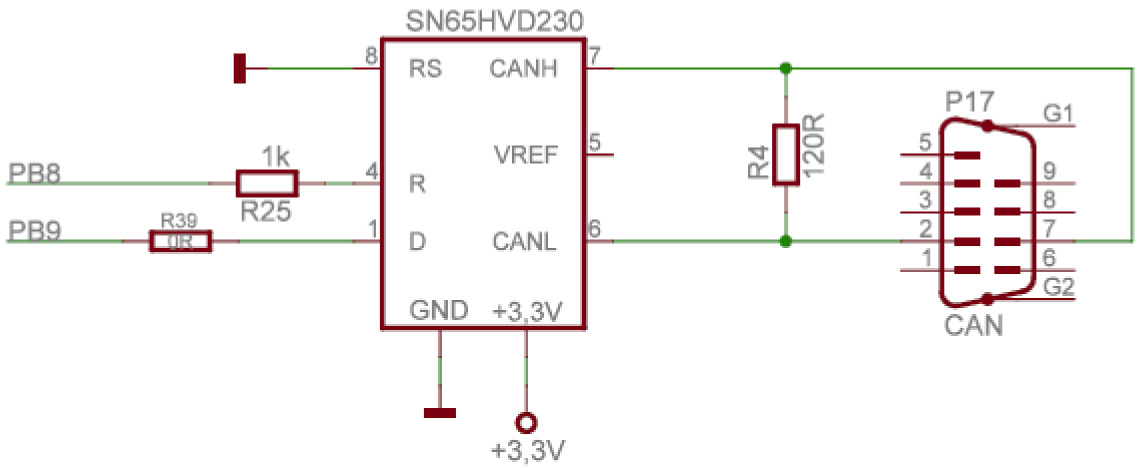
- Example: Telephone channel
  - W = 3.4 kHz and SNR ~ 38 dB => SNR ~ 6310
  - C = 3.4 kHz log2 (1 + 6310) = 3.4kHz x 12.62 b/s

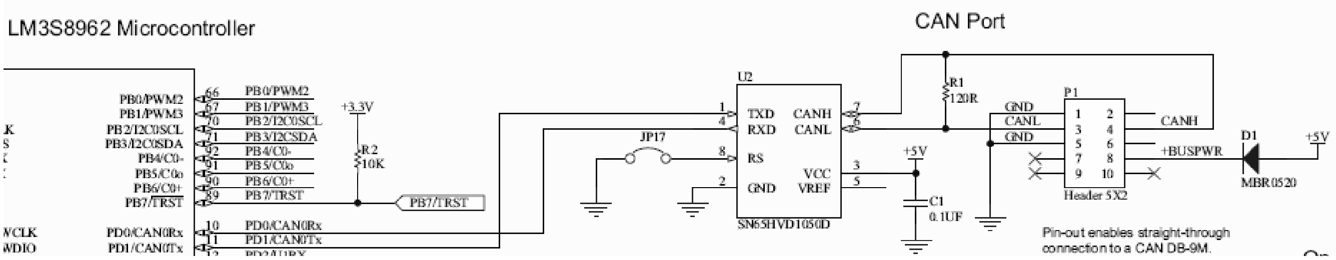
= 42.9 kb/s

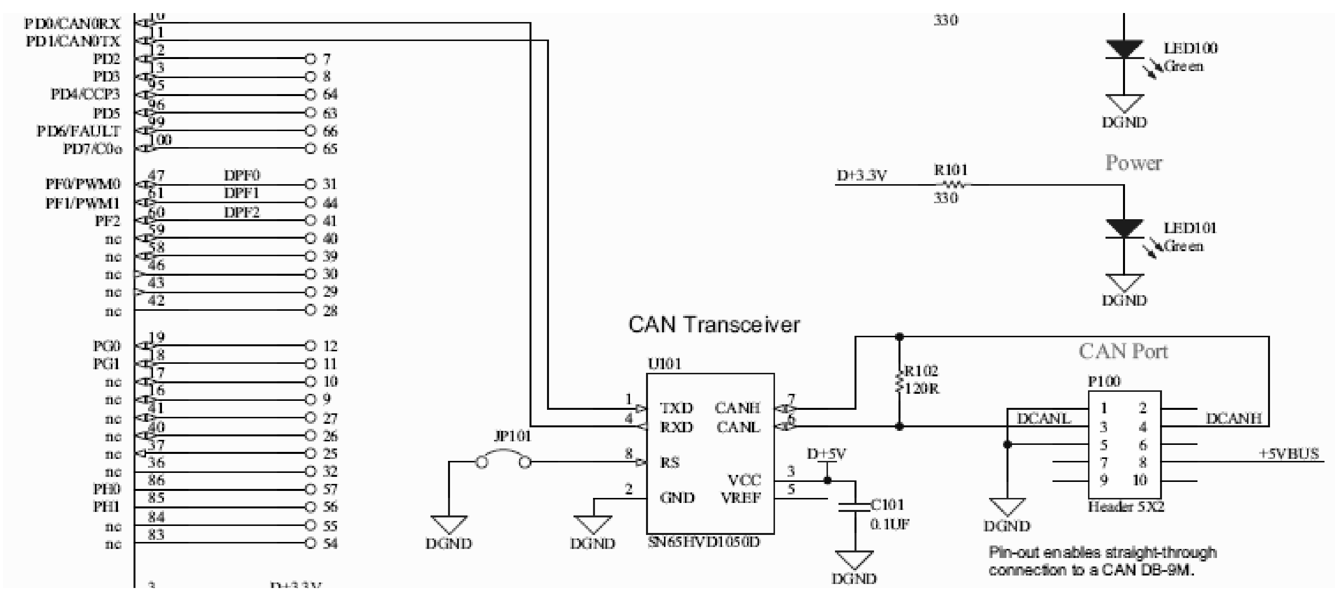
## Controller Area Network (CAN).

- High-integrity serial communications
- Real-time applications
- Up to 1 Mbits/second
- Originally for use in automobiles,
- Can have up to 112 nodes
- Half duplex (both directions, but only one direction at a time)

▪



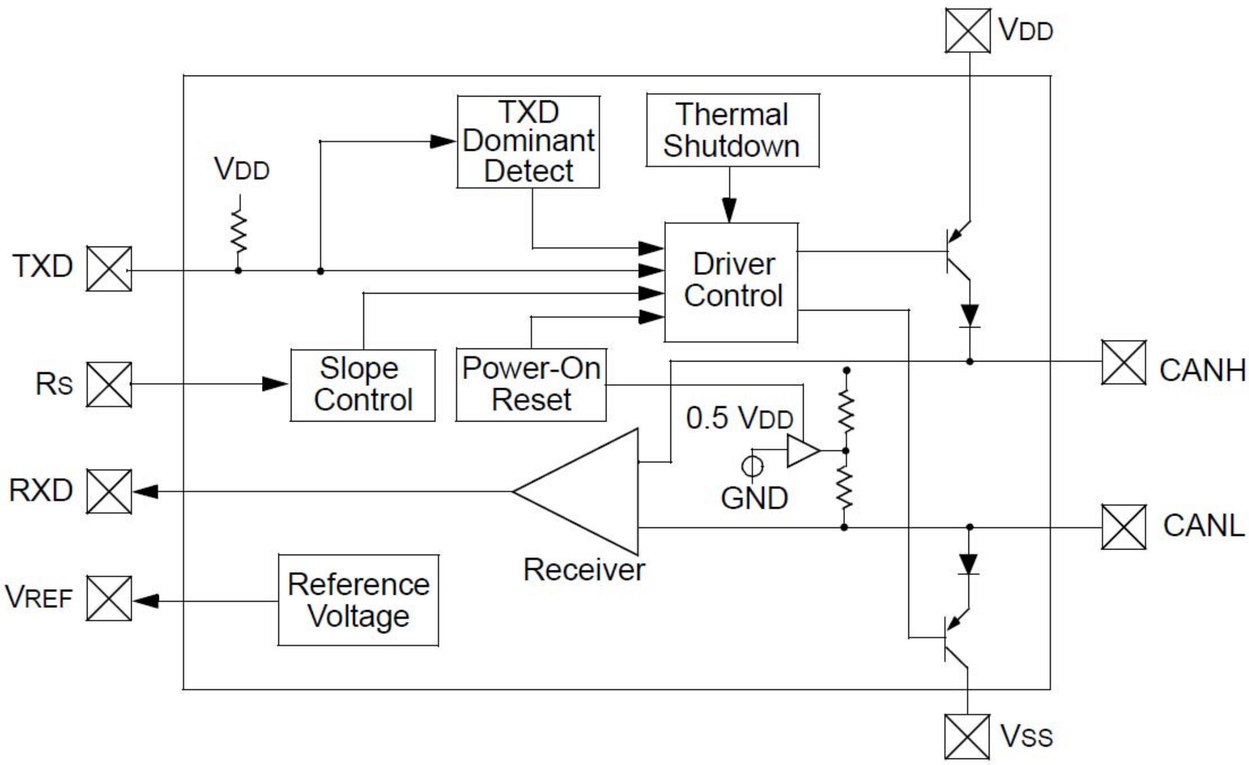




MCP2551

http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en010405

http://ww1.microchip.com/downloads/en/DeviceDoc/21667f.pdf



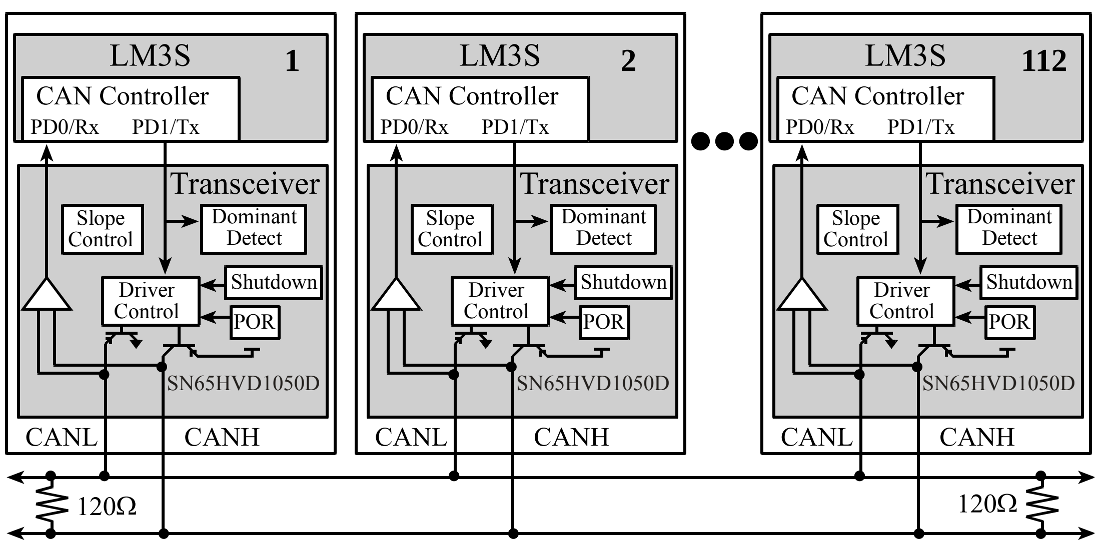




*Figure 9.3. Block Diagram of a CAN communication system (Rs=0V, Vdd=5V, Vref=nc)*

```
CANBitRateSet(CAN0_BASE, 80000000, CAN_BITRATE);
```

**There must be a 120Ω resistor on each end of the CAN cable, and no resistor on middle nodes.**

$$f \approx 1/\tau$$
$$v = \mathrm{VF}*\mathrm{c} = 2\cdot 10^8 \text{ m/s}$$
$$\lambda = v/f \approx v\ \tau$$
$$\text{a } \textit{transmission line} \text{ if } L > \lambda/4$$
$$\text{slew rate} = 25\text{V}/\mu\text{s}$$
$$1\text{V in } 40 \text{ ns}, \lambda = 2\cdot 10^8 \text{ m/s } 40*10^{-9}\text{s} = 8\text{m}$$
$$\lambda/4 = 2\text{m}$$

Similar to wire-or open collector logic

Dominant state is logic low

Recessive state is logic high

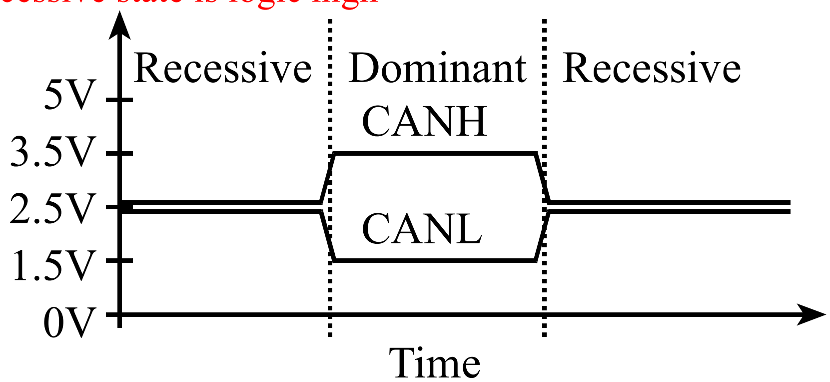



*Figure 14.6. Voltage specifications for the recessive and dominant states.*

Four message types or frames

- **Data Frame**,
- **Remote Frame**,
- **Error Frame**, and
- **Overload Frame**.

```
TxMessage.RTR = CAN_RTR_DATA;
```

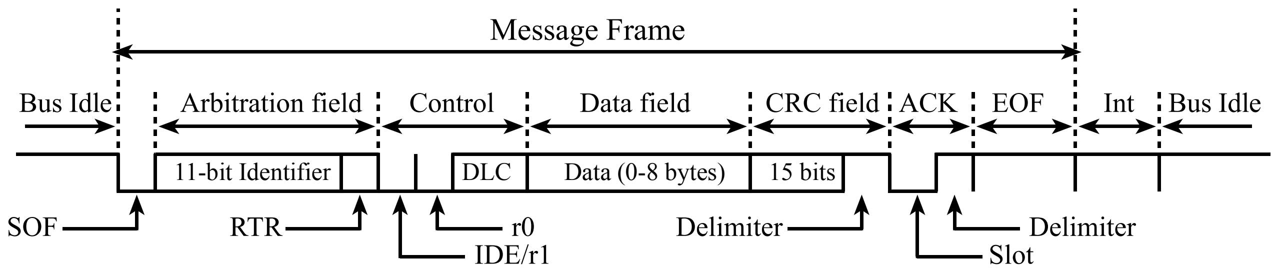



*Figure 14.7. CAN Standard Format Data Frame.*

**Arbitration Field**

11-bit identifier specifies data type (not address)

Ping,

IR, or

Touch sensor

priority handled by dominate wins over recessive

lower IDs are higher priority

RTR=IDE=0 means 11-bit standard format data frame

**Control Field**

DLC, which specifies the number of data bytes (0 to 8)

**Data Field**

contains zero to eight bytes of data.

**CRC Field**

15-bit checksum used for error detection.

$$\text{Bandwidth} = \frac{\text{number of information bits/frame}}{\text{total number of bits/frame}} \bullet \text{baud rate}$$

Number of bits in a CAN message frame.

ID (11 or 29 bits)

Data (0, 8, 16, 24, 32, 40, 48, 56, or 64 bits)

Remaining components (36 bits)

SOF (1)

RTR (1)

IDE/r1 (1)

r0 (1)

DLC (4)

CRC (15)

ACK/EOF/intermission (13)

```
TxMessage.StdId = id;          // message ID
TxMessage.IDE = CAN_ID_STD;    // 11-bit address
TxMessage.DLC = 2;             // 0 to 8
```

**How many bits in a frame:**

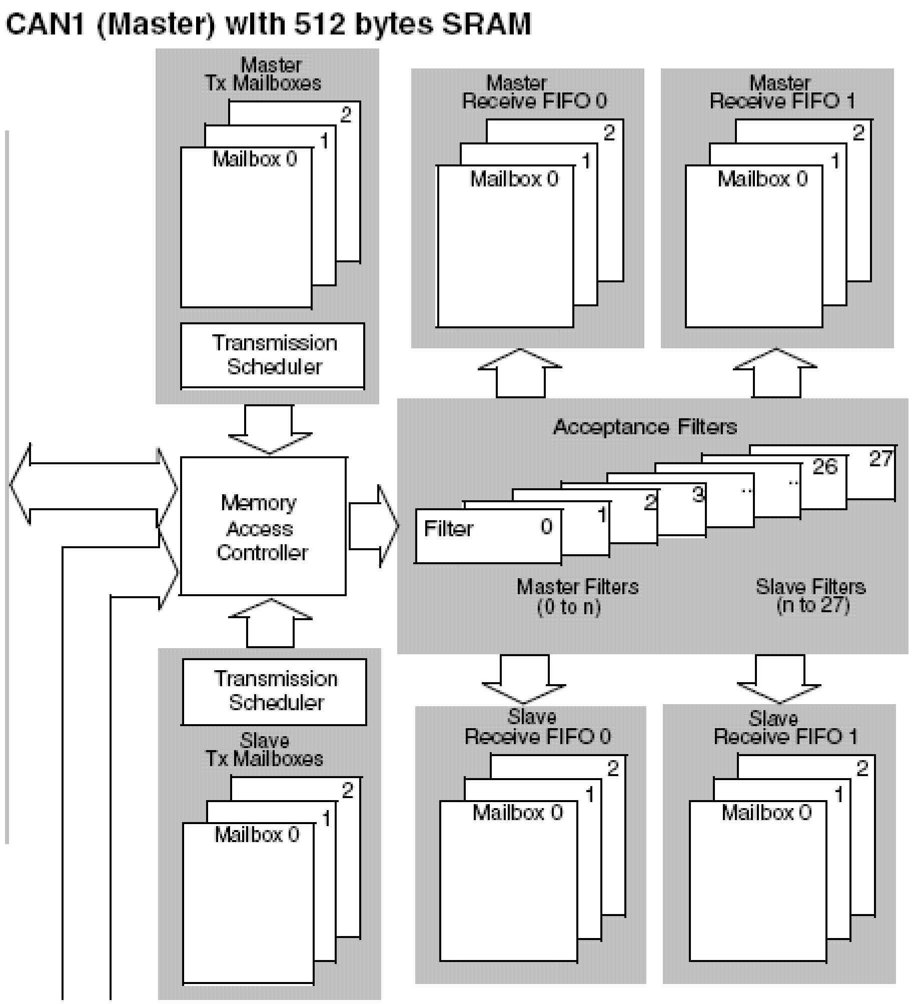
- Standard CAN 2.0A frame with 4 data bytes?
- Extended CAN 2.0B frame with 8 data bytes?

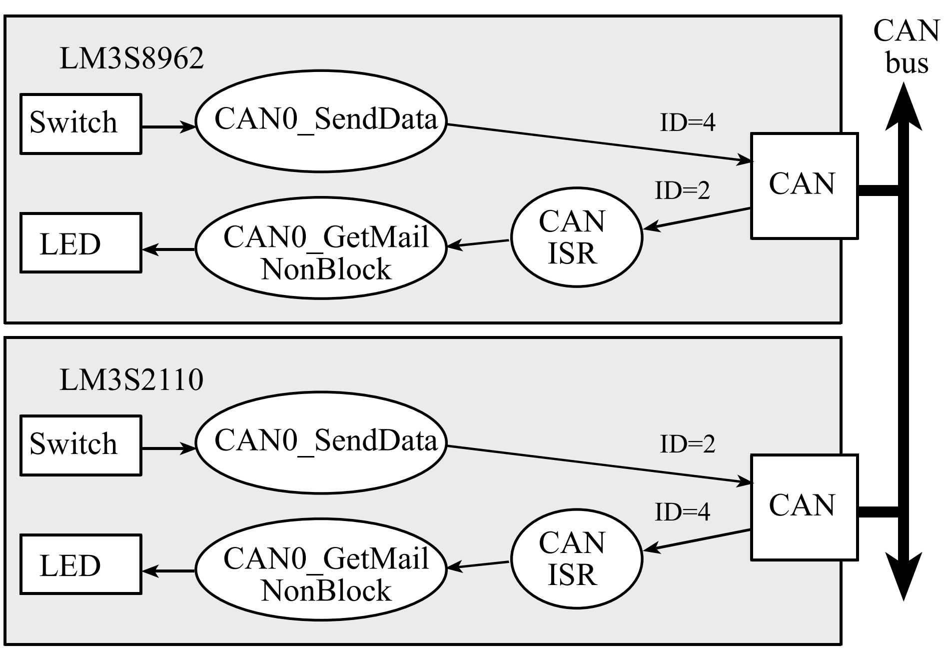
**Bit Stuffing**

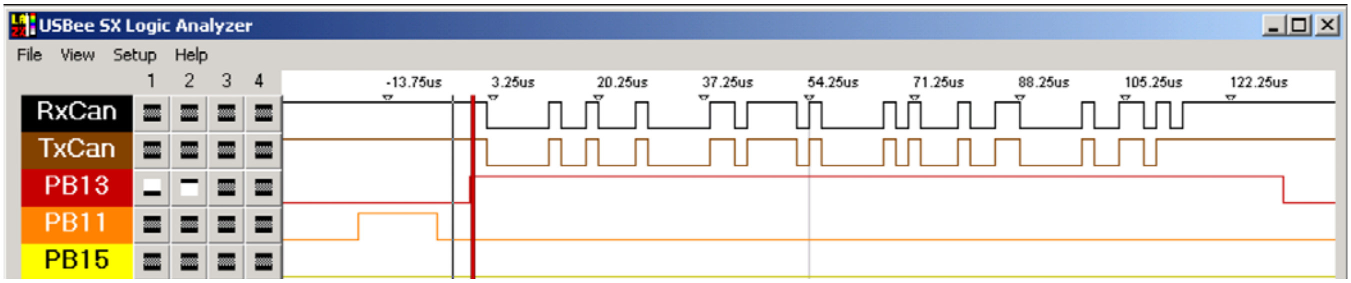
Where is the clock? (Answer: in the data)
Data line needs edges so the receiver can synchronize
A long sequence of 0's or a long sequence of 1's,
Insert a complementary bit after five bits of equal value.
CAN 2.0A may add 3+**n** stuff bits (**n** is number of bytes)
CAN 2.0B may add 5+**n** stuff bits.
Receiver has to un-stuff

**Filter on receive messages**

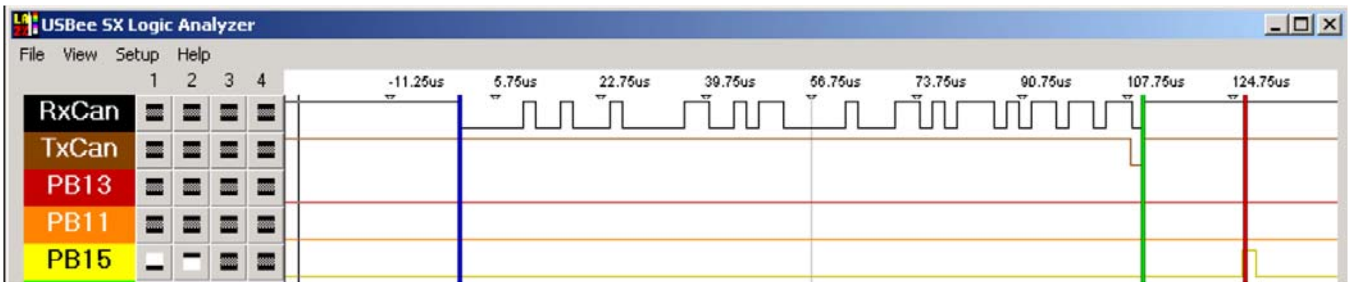
Which IDs to accept?
  **`IdMsg`** is the ID of the incoming message
  **`IdRule`** is the ID setup in the filter rule
14 filters: rules accept if ID matches
  Accept **`if (IdMsg&Mask)==(IdRule&Mask)`**
  Accept **`if IdMsg is in the list`**
Into which FIFO to put message?
ID+MASK (0 don't care, 1 bit must match) **`CAN_FilterMode_IdMask`**
list of IDs **`CAN_FilterMode_IdList`**


CAN1 (Master) with 512 bytes SRAM
Master
Tx Mailboxes
Mailbox 0
1
2
Transmission
Scheduler
Master
Receive FIFO 0
Master
Receive FIFO 1
Acceptance Filters
Memory
Access
Controller
Filter
0
1
2
3
..
26
27
Master Filters
(0 to n)
Slave Filters
(n to 27)
Slave
Tx Mailboxes
Slave
Receive FIFO 0
Slave
Receive FIFO 1


LM3S8962
Switch
CAN0_SendData
ID=4
CAN
ID=2
CAN
ISR
LED
CAN0_GetMail
NonBlock
CAN
bus
LM3S2110
Switch
CAN0_SendData
ID=2
CAN
ID=4
CAN
ISR
LED
CAN0_GetMail
NonBlock




Why did it take 130µs to execute **`CAN_Transmit`**?
Why does the RxCan have more stuff than the TxCan?




What is the total number of bits in this frame?
Why did it take 110µs to complete an entire frame?
What is that blimp on TxCan?
Where is the end of the frame?
What is the bandwidth?

## Synchronization issues

How to connect transmitter/receiver threads?
- How to start, handshake
- Race conditions

How to prevent streaming data from stalling?
- Priority, buffer size