Comunicação Série

7. CAN (**C**ontroller **A**rea **N**etwork)

CAN

- Bus série, multi-master, com capacidade tempo-real
- Concebido por engenheiros da Bosch no início da década de 1980.
- Ideia inicial- desenvolver um sistema de comunicação entre unidades eletrónicas de controlo (ECU)
- Desenvolvimento de um novo standard – nenhum dos protocolos de comunicação existentes satisfazia os requesitos de velocidade e fiabilidade necessários
- Protocolo otimizado para sistemas que precisam de transmitir pequenas quantidades de informação de um modo fiável (pequenas quando comparadas com Ethernet ou USB)

CAN

Descrição do CAN nas normas ISO:

- Um protocolo de comunicação série que suporta de um modo eficiente a distribuição de comandos em tempo real com um alto nível de segurança.
- Os seus domínios preferenciais de utilização são as aplicações de redes de alto débito com elevada fiabilidade de transmissão e ligações multiplexadas de baixo custo.

CAN

- Propriedade da Bosch; licenciado à maioria dos fabricantes de semicondutores
- Módulos CAN incluídos na maior parte das famílias atuais de microcontroladores (PIC, …)
- Atualmente o bus mais popular na indústria automóvel
- Competidores: J1850 (USA), VAN (França) e PALMNET (Japão)
- Muitas aplicações em automação e controlo para além da indústria automóvel (low level field bus)

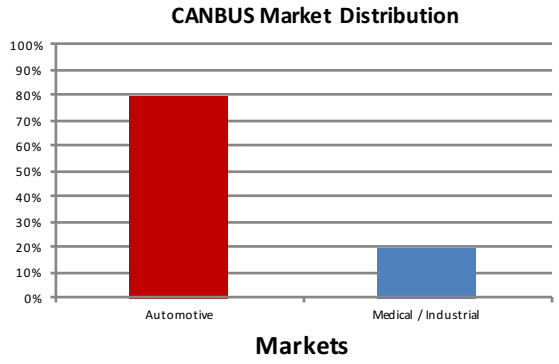
Desenvolvimento do bus CAN

- 1983: início de um projeto na Bosch, na Alemanha, para desenvolver uma rede de comunicação para veículos automóveis
- 1986: introdução do protocolo CAN
- 1987: primeiros controladores CAN da Intel (82526) e da Philips (82C200)
- 1991: especificação CAN 2.0A da Bosch
- 1992: Criação do grupo de fabricantes e utilizadores *CAN in Automation* (**CiA**) que publicam o CAN *Application Layer Protocol* (**CAL**)
- 1993: Standard ISO 11898-1 (Data Link Layer), ISO 11898-2 (Physical Layer for high-speed CAN) e ISO 11898-3 (Physical Layer for low-speed, fault-tolerant CAN).
- 1995: **CANopen** (application layer protocol) publicado pelo CiA
- 2000: Desenvolvimento de **TTCAN** (Time-Triggered Communication protocol for CAN)
- 2012 Bosch: CAN FD 1.0 (CAN with Flexible Data-Rate)

➢ **Presente : a maioria dos veículos automoveis usa o bus CAN**.

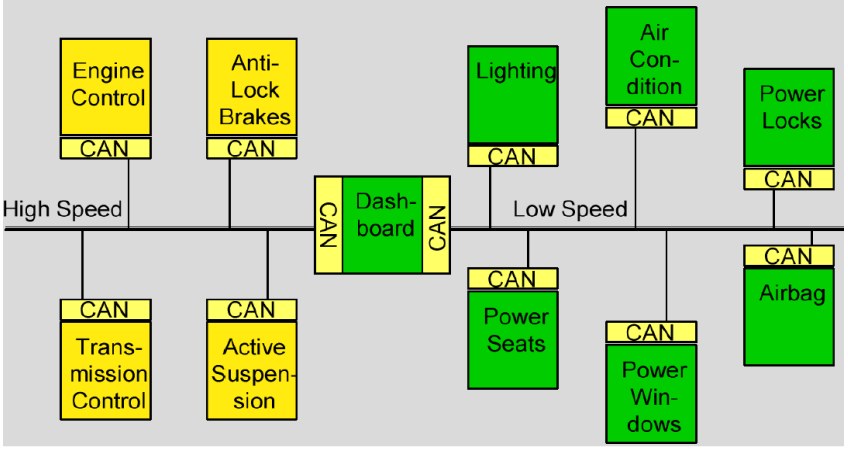
Quem usa o bus CAN?

- Desenhado originariamente para utilização em veículos automóveis, onde se tornou standard
- Atualmente – automação industrial / equipamento médico



CAN no automóvel: 2 buses

(Mercedes, Wolkswagen, BMW, Renault, Fiat, Volvo,…)



Classe C – 1 Mbit/s

Classe A – 10 Kbits/s

Microcontroladores num carro:

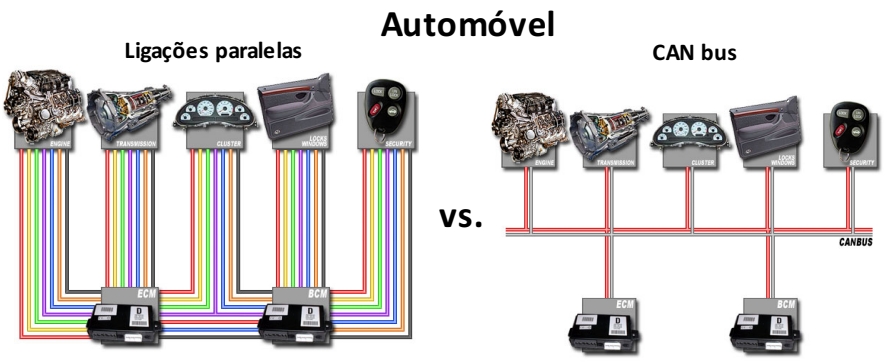
- ABS (1 + 4)
- keyless entry system(1)
- active wheel drive control (4)
- engine control (2)
- airbag sensor(6++)
- seat occupation sensors(4)
- automatic gearbox(1)
- electronic park brake(1)
- diagnostic computer(1)
- driver display unit(1)
- air conditioning system(1)
- adaptive cruise control(1)
- radio / CD-player(2)
- collision warning radar(2)
- rain/ice/snow sensor systems (1 each)
- dynamic drive control(4)
- active damping system (4)
- driver information system(1)
- GPS navigation system(3)

CAN – exigências

- Assegurar fiabilidade nas comunicações em aplicações críticas de sistemas de controle (p.ex. gestão do motor, sistema anti-blocagem das rodas,...)
- Aspetos importantes para a fiabilidade:
  - Disponibilidade do bus quando é necessário enviar dados importantes
  - Possibilidade dos bits numa mensagem serem corrompidos por ruído
  - Falhas elétricas ou mecânicas nas linhas do bus

CANBUS Physical Layer

- Meio Físico – 2 fios terminados em ambas as extremidades por resistências
- Sinal Diferencial (*twisted pair*) – melhor imunidade ao ruído.
- Vantagens de bus série:
  - Peso e custo reduzidos
  - Menos fios = maior fiabilidade



Vantagens do CAN

- Standard bem estabelecido
- Existência de muitos produtos e ferramentas CAN no mercado
- Protocolo implementado em Hardware
- Conjugação de tratamento de erros e confinamento dos erros com velocidades de transmissão elevadas
- Meio de Transmissão Simples
  - Par entrançado (twisted pair) standard, mas um unico fio tambem possivel
  - Outros meios disponiveis: ligações óticas ou via rádio
- Excelente tratamento de Erros
  - CRC para detetar erros
  - Confinamento dos erros: mecanismo para evitar que um nó avariado bloqueie o sistema
- Protocolo mais usado na indústria e em veículos de transporte
- A melhor relação Preço / Performance

Caraterísticas

- Bus Multi-master assíncrono
- Inexistência de endereçamento dos nós; *broadcasting* das mensagens; (*content-addressing*)
- Mecanismos sofisticados de deteção e tratamento de erros
- Até 1 Mbit/s
- Frequências de funcionamento mais comuns: 1 MHz, 500 KHz e 125 KHz
- Comprimento máximo – dependente da frequência de funcionamento:
  - 1 MHz: até 40 m
  - 125 KHz: até 500 m
  - 50 KHz: até 1 km

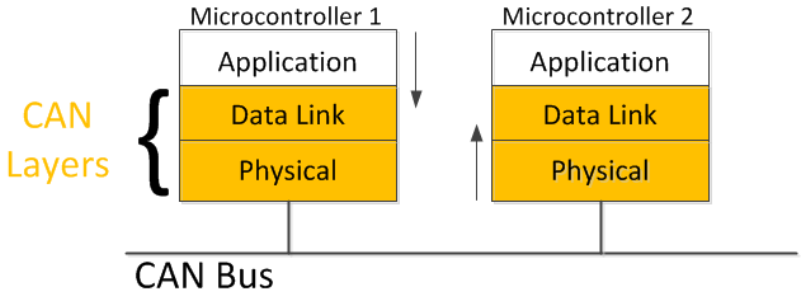
CAN e o modelo OSI

CAN é uma rede fechada
- desnecessário segurança ou logins.
- desnecessário interface com o utilizador.

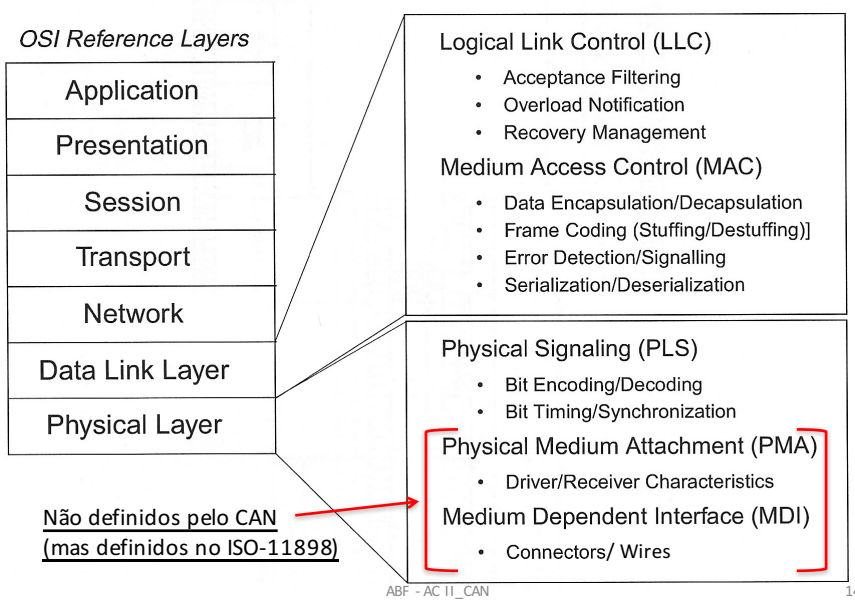
Physical and Data Link layers no silício.

Modelo OSI – 7 camadas (Physical, Data Link, Network, Transport, Session, Presentation, Application)

CAN:



Modelo de Referência OSI e as redes CAN



CAN Higher Level Protocols (HLPs)

HLPs correspondem aos níveis 3 a 7 do modelo OSI

São usados para:

1. Standardizar procedimentos de *startup* (incluindo *bit rates* usadas)
2. Distribuir endereços / tipos de mensagens entre os nós
3. Determinar a estrutura das mensagens
4. Fornecerem rotinas de processamento de erros ao nível do sistema

Application Layer standards

- CAN não define o nível de aplicação – cabe aos fornecedores desenvolver o respetivo software, originando soluções diferentes, fechadas, e exigindo um investimento significativo
- Esforços para a definição de standards ao nível da aplicação têm sido feitos desde a década de 1990.

  Impuseram-se 2 standards :
  - **DeviceNet** (Allen Bradley) otimizado para automação fabril
  - **CANopen** (Projeto ESPRIT) otimizado para máquinas industriais e outros dispositivos, sistemas médicos, …

CANopen

Caraterísticas

- CANopen é um subconjunto de CAL (CAN Application Layer) desenvolvido pela CiA (CAN in Automation)
- Auto configuração da rede
- Acesso fácil a todos os parâmetros dos dispositivos
- Sincronização dos dispositivos
- Transferência de dados cíclica e desencadeada por eventos
- Estabelecimento ou leitura de inputs, outputs ou parâmetros síncrona

Aplicações

- Automatização de Máquinas

Vantagens

- Permite a integração de pequenos sensores e atuadores
- Standard aberto e independente do vendedor
- Suporta a inter-operabilidade de diferentes dispositivos
- Capacidade de tempo-real de alta velocidade

CAN Physical Layer

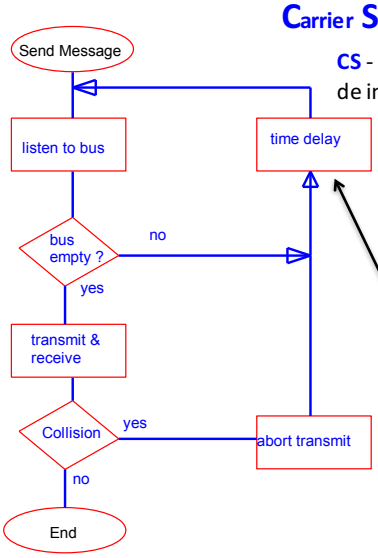
- O facto do protocolo não definir a camada física (PMA e MDI) permite usar diversas ligações físicas em redes CAN:
  - Ligação unifilar
  - Par diferencial (a solução mais comum)
  - Fibra ótica (redes CAN de alto débito)
    - Solução usada pela Mercedes

Message Oriented Transmission Protocol

- Cada nó é recetor e transmissor
- Cada mensagem é transmitida a todos os dispositivos ligados ao bus (***broadcast***)
- Todos os nós lêem a mensagem, e decidem se é relevante para eles (*filtering*)
- Todos os nós verificam a inexistência de erros na receção
- Os nós fazem o *acknowledge* da receção

- Transmissão usa código **NRZ** (Non-Return to Zero)

Acesso ao Bus – Ethernet: CSMA/CD



Carrier Sense Multiple Access with Collision Detection

CS - Cada nó monitora o bus para detetar um período de inatividade antes de tentar enviar uma mensagem

MA - Quando há um período de inatividade todos os nós têm igual oportunidade de enviar uma mensagem

CD – se 2 nós começam a transmitir ao mesmo tempo, os nós detetam a colisão e tomam as medidas apropriadas

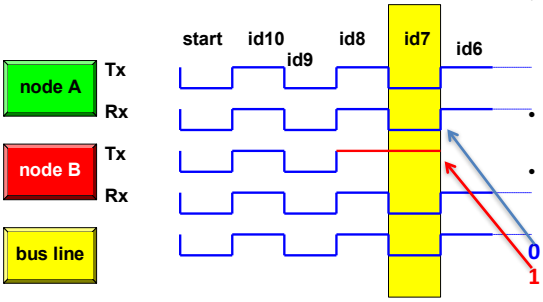
Ethernet – os nós abortam a transmissão e tentam de novo após um time delay

Incompatível com utilizações tempo-real
Não pode ser usada por CAN

CAN - Collision Avoidance (**CA**)
A arbitragem processa-se sem corromper ou atrasar a mensagem com prioridade mais alta

CAN: CSMA/CA

Carrier **S**ense **M**ultiple **A**ccess with **C**ollision **A**voidance



- Se há colisão ***"the winner takes the bus"*** **-** a mensagem com prioridade mais alta não é atrasada
- capacidade de tempo-real para mensagens com prioridade alta
- quanto menor o identificador, mais elevada é a prioridade

0 "***dominant level***" como ***wired-AND***
1 "***recessive level***"

1. A e B acedem simultâneamente ao bus
2. $id7_A = 0$, $id7_B = 1$; bus line = 0 $Rx_B \neq Tx_B \rightarrow$ B retira-se (desiste de transmitir)
3. A adquire o bus e prossegue transmissão da mensagem; B para transmissão.

Tipos de mensagem

O protocolo CAN define 5 tipos de mensagens (frames):

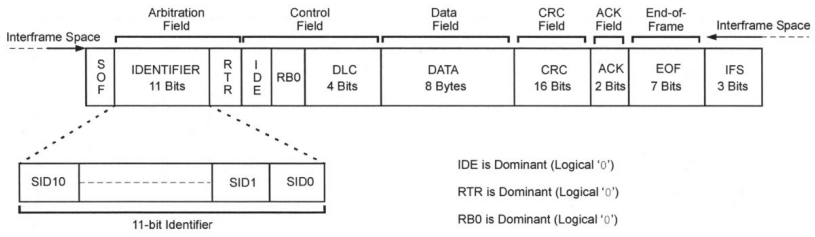
- ***Data Frame*** – usado quando um nó transmite informação aos outros nós da rede
- ***Remote Frame*** – basicamente uma data frame com o bit RTR (**R**emote **T**ransmit **R**equest) set (= 1), significando que se trata de um pedido de transmissão remoto (i.e. pedido para que outro nó envie infomação)
- ***Error Frame*** – geradas por nós que detetam um erro de protocolo
- ***Overload Frame*** - geradas por nós que precisam de mais tempo para processar as mensagens que já receberam
- ***Interframe*** – as *data frame* e *remote frame* são temporalmente separadas por uma *interframe*

Moldura das mensagens CAN

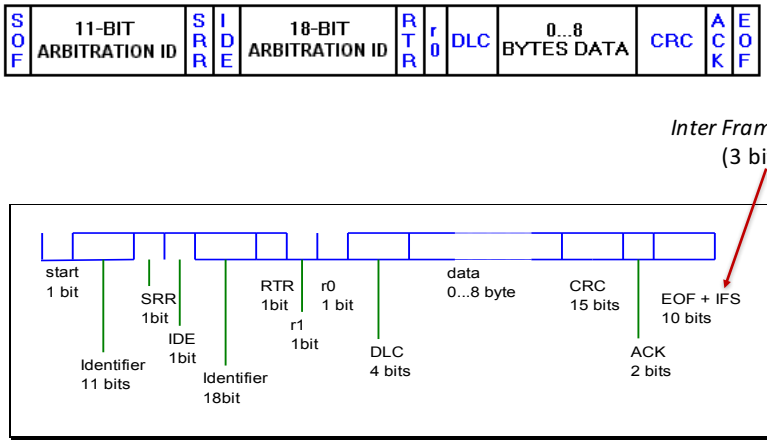
Cada trama de dados (*data frame*) tem os seguintes campos:

- SOF – Start Of Frame: 1 bit dominante
- Arbitration:
  - denota a prioridade da mensagem
  - Standard frame (CAN 2.0A): 11 bit-identifier + RTR bit
  - Extended frame (CAN 2.0B): 29 bit-identifier + RTR bit
- Control: 6 bits – IDE, RBO, DLC (Data Length Code – 4 bits)
- Data:
  - Até 8 bytes por mensagem
  - Mensagem com 0 byte permitida
- CRC:
  - Cyclic Redundancy Check ; contem uma *checksum* gerada por uma divisão polinomial (binária)
- EOF - End Of Frame:
  - contem *acknowledge* (2-bit), mensagens de erro, *end of message*

CAN Standard Data Frame



CAN Extended Data Frame



Formato das mensagens

- **Start bit (1 bit - dominant):** marca o início da mensagem; a transição **1 -> 0** sincroniza todos os transmissores
- **Identifier (*Arbitration Field* - 11 bit) :** nome da mensagem e respetiva prioridade; quanto menor o valor mais alta a prioridade
- **RTR (1 bit):** Remote Transmission Request ; se **RTR=1 (recessive)** a moldura não contem dados válidos – é um pedido aos recetores para enviarem as suas mensagens
- **IDE (1 bit): ID**entifier **E**xtension; se IDE=1 então ***extended CAN-frame***
- **R0 (1 bit):** Reservado
- **DLC (4 bit): D**ata **L**ength **C**ode, indica o numero de bytes de dados incluídos na mensagem
- **Data (0..8 byte):** os dados da mensagem
- **CRC (15 bit): C**yclic **R**edundancy **C**ode; deteção de erros (não correção!); deteta até 5 erros single bit
- **ACK (2 bit): ACK**nowledge; cada nó que recebe a mensagem sem erros (inclui CRC) tem de transmitir um acknowledge-bit neste slot
- **EOF (7 bit = 1, recessive): E**nd **O**f **F**rame; violação intencional da regra de bit-stuffing (depois de 5 recessive bits 1 stuff-bit é automaticamente adicionado
- **IFS (3 bit = 1 recessive): I**nter **F**rame **S**pace; tempo para copiar a mensagem recebida do *bus-handler* para o buffer

Extended Frame:
- **SRR (1 bit = recessive): S**ubstitute **R**emote **R**equest; substitui o bit RTR das standard frames
- **R1 (1 bit):** reservado

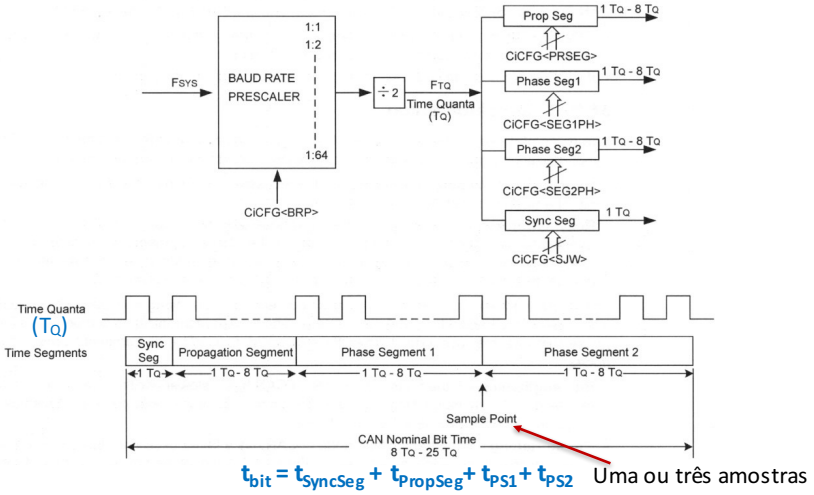
Bit-stuffing

- Código NRZ – se há muitos bits consecutivos iguais não há mudanças de tensão na linha – impossível resincronizar relógio local dos nós

➢ Solução: *Bit-stuffing* - depois de 5 bits consecutivos iguais, transmitir um bit extra de valor diferente para permitir ressincronização

  - Nota: o bit "*stuffed*" tem de ser descartado do fluxo de dados na receção

CAN Bit Time



$t_{bit} = t_{SyncSeg} + t_{PropSeg} + t_{PS1} + t_{PS2}$ Uma ou três amostras

$t_{PropSeg}$ - tempo de propagação da informação no bus

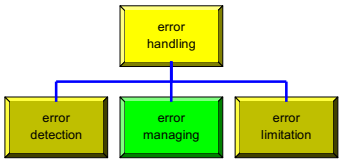
Exemplo de uma transação no bus

	ID	Data
1. Painel de instrumentos: "pode alguem dizer-me qual a temperatura do bloco motor?" (*remote frame*)	400	
2. Bloco motor vê a mensagem e emite a mensagem: "temperatura do bloco 76 graus" (*data frame*)	400	076
3. Painel de instrumentos vê a mensagem e visualiza a temperatura no painel		

Reconhecimento de Erros em CAN

- **Bit-Error**
  - O bit transmitido não é lido com o mesmo valor (exceto nos slots de arbitragem e *acknowledge*)
- **Bit-Stuff-Error**
  - Mais de 5 bits sucessivos lidos com o mesmo valor (exceto '*end of frame*' da mensagem)
- **CRC-Error**
  - A soma CRC recebida não coincide com a calculada
- **Format-Error**
  - Violação do formato da mensagem, e.g. CRC-delimiter não é *recessive* ou violação do campo '*end -of-frame*'
- **Acknowledgement-Error**
  - transmissor não recebe *dominant bit* durante o *acknowledgement slot*, i.e. a mensagem não foi recebida por nenhum nó.

CAN Error Sequence

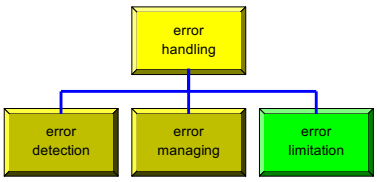


Após a deteção de um erro por um nó, todos os outros nós recebem uma frame específica:
Error–Frame - viola a regra do *stuff-bit* (transmissão de não mais de 5 bits seguidos iguais)
passive error frame – 6 bits dominantes
active error frame – 12 bits dominantes
A Error-Frame leva todos os outros nós a reconhecer um ***Error Status*** no bus.

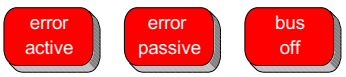
Error Management Sequence

1. Detetado um erro
2. *Error-Frame* transmitida por todos os nós que detetaram o erro
3. A última mensagem recebida é cancelada por todos os nós
4. Os contadores de erro (internos, h/w) são incrementados
5. A mensagem original é transmitida de novo.

CAN Error Status



* **Objetivo:** evitar perturbações persistentes do CAN desligando os nós defeituosos
* **3 Error States :**



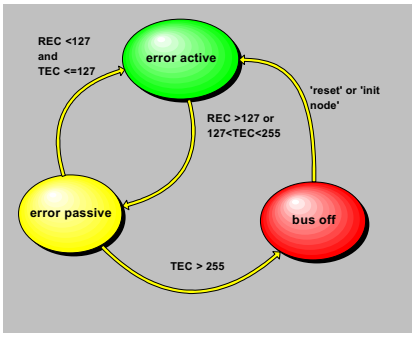
Error Active : modo normal, de operação, permite que o nó transmita e receba mensagens sem restrições. Em caso de erro é transmitida uma ***active error frame***

Error Passive : após a deteção de um número fixo de erros, o nó passa a este estado. As mensagens serão recebidas e transmitidas. Em caso de erro é transmitida uma ***passive error frame*****.**

Bus Off : quando o contador de erros de transmissão excede 255 o nó é separado da rede CAN; nem a transmissão nem a receção de mensagens é permitida, o nó não pode transmitir uma error frame.
A saída deste estado só é possível através de ***reset*** **(*"fault confinement"* –** assegura que nenhum nó pode bloquear a rede, garantindo sempre a transmissão das mensagens essenciais.**)**

CAN: Contador de Erros

Diagrama de Estados



- Transições entre estados feitas automaticamente pelo módulo CAN
- As transições entre estados são determinadas por 2 contadores de erros:
  Receive **E**rror **C**ounter **(REC)**
  Transmit **E**rror **C**ounter **(TEC)**
- **Situações Possíveis:**

a) um transmissor reconhece um erro
TEC:=TEC + 8

b) um recetor vê um erro: **REC:=REC + 1**

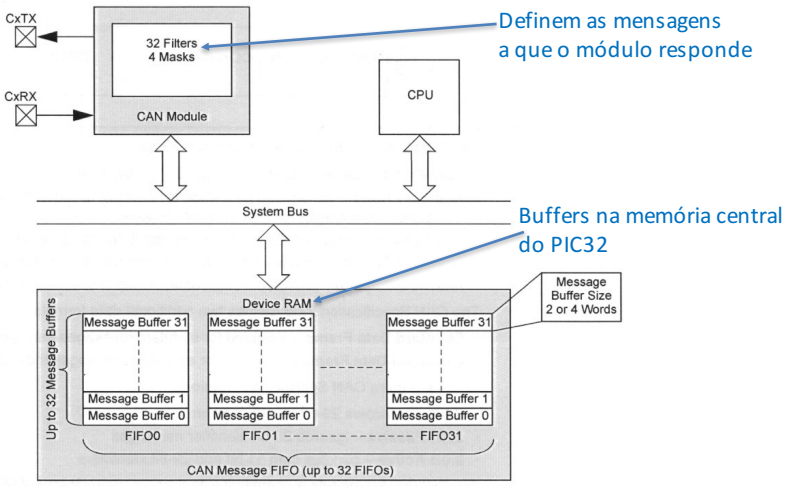
c) um recetor vê um erro, depois de transmitir uma error frame: **REC:=REC + 8**

d) Se um nó '*error active*' deteta um erro *bit-stuff* durante a transmissão de uma *error frame*.
TEC:=TEC+ 1

e) Transmissão bem succedida: **TEC:=TEC - 1**

f) Receção bem succedida: **REC:=REC - 1**

Módulos CAN do PIC32
(implementa CAN 2.0B)



Bibliografia CAN

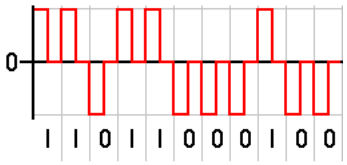
- "Controller Area Network (CAN) Basics", Application Note AN713, Microchip
- "Understanding Microchip's CAN Nodule Bit Timing", Application Note AN754
- "Le bus CAN", D. Paret, Dunod, 1996

Comunicação Série

APENDICE:
CÓDIGOS DE TRANSMISSÃO

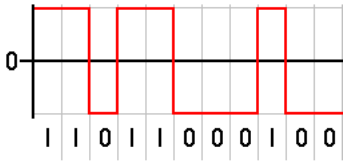
Códigos de comunicação

RZ



Sinal regressa a 0 entre a transmissão de 2 bits sucessivos – ***self clocking***

NRZ



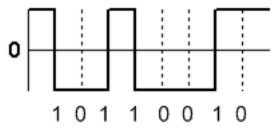
O valor do sinal só muda quando muda o valor do bit a transmitir. CAN usa NRZ

bit stuffing

– **0** acrescentado quando são transmitidos cinco 1 consecutivos

- **1** acrescentado quando são transmitidos cinco 0 consecutivos

NRZI

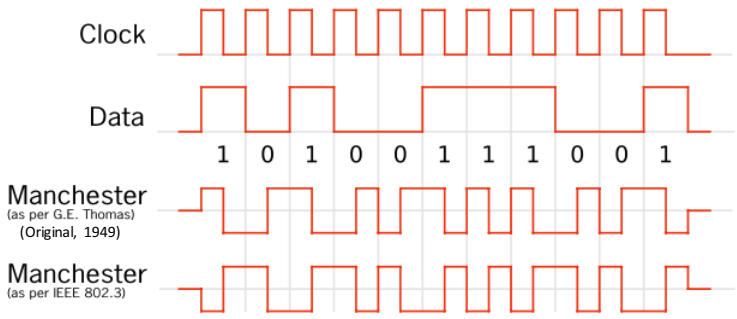


O valor do sinal muda na transição positiva do relógio se bit a transmitir é 0, não muda se é 1
USB 2.0 usa NRZI

bit stuffing – **0** acrescentado quando são transmitidos seis 1 consecutivos

Manchester or Phase Encoding (PE)

Self-clocking



IEEE 802.3

original data	=	clock	XOR	Manchester value
0		0		0
0		1		1
1		0		1
1		1		0

Cada bit é transmitido num intervalo de tempo fixo (o "período").
0 é expresso como uma transição *low-to-high*,
1 por uma transição *high-to-low*

No IEEE 802.3 a convenção é a inversa

Ocorre sempre uma transição a meio do período.
Transições no início do período só servem para colocar o sinal no nível adequado