Università degli Studi RomaTre

- Dipartimento di Informatica e Automazione -

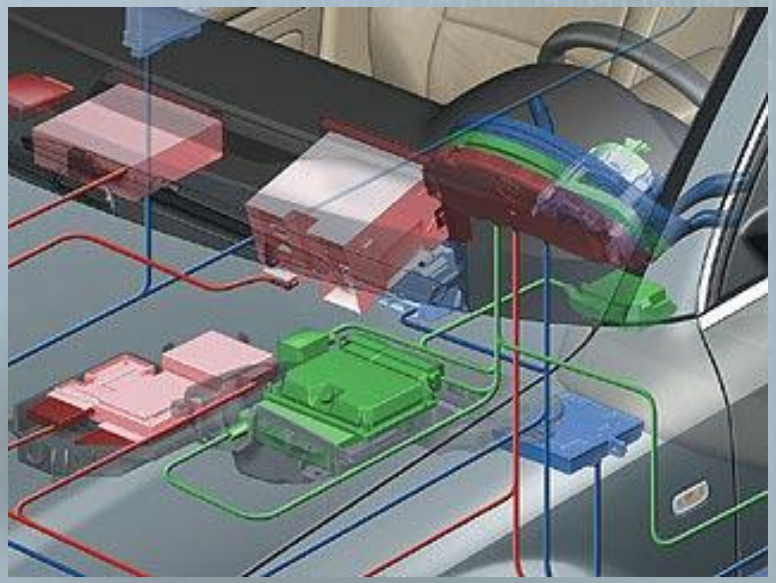
BOSCH's
CONTROLLER AREA NETWORK

Tesina di "Reti E Sistemi Per L'automazione" 2005 di Antonello Galanti

Introduzione

- Il protocollo CAN (controller area network) è un bus **seriale** di comunicazione digitale di tipo "**broadcast**". Esso permette il controllo real-time distribuito con un livello di **sicurezza** molto elevato.
- E' stato introdotto dalla **Bosch** nei primi anni '80 per applicazioni automobilistiche, per consentire cioè la comunicazione fra i **dispositivi elettronici intelligenti** montati su un autoveicolo, ma si è diffuso ormai in molti settori dell'industria.
- Sono sorti anche consorsi di aziende che promuovono questa diffusione

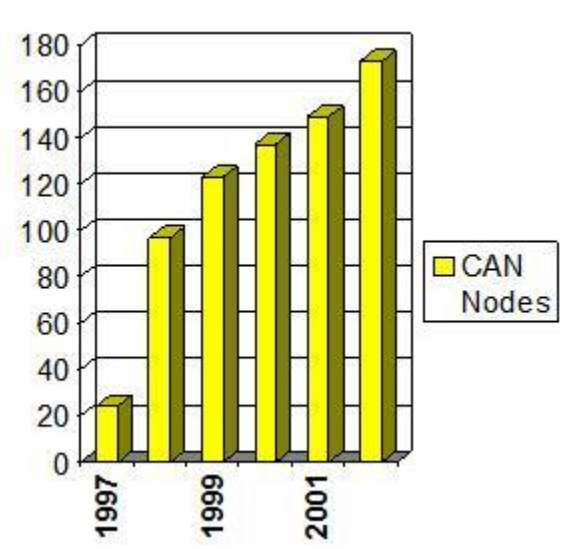
(CAN in Automation www.can-cia.de)

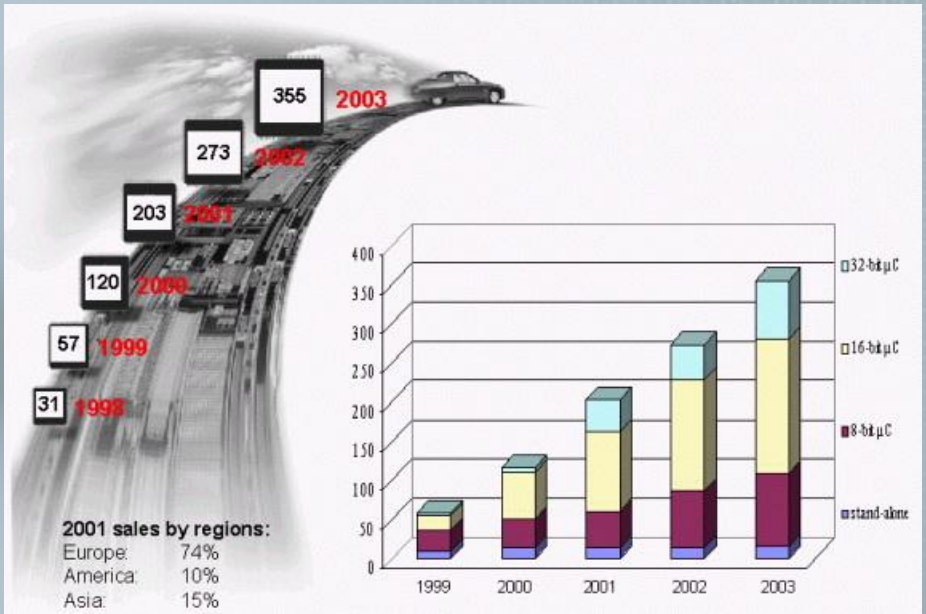


BOSCH

Introduzione

- Sono in molti a vedere nel Bus CAN lo standard dominante nelle reti industriali del prossimo **futuro**.





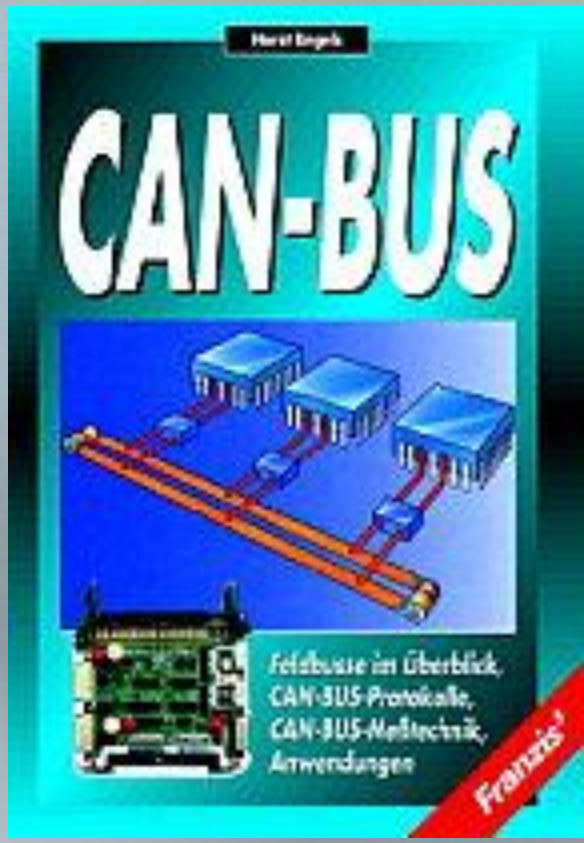
- Il suo **successo** è dovuto ai notevoli vantaggi tecnologici che offre:
  - **tempi di risposta rigidi**, specifica fondamentale nel controllo di processo: la tecnologia CAN prevede molti strumenti hardware e software e sistemi di sviluppo per protocolli ad alto livello (il bus CAN implementa solo i primi due livelli della pila ISO-OSI) che consentono di connettere un elevato numero di dispositivi mantenendo stringenti vincoli temporali.

Introduzione

- **semplicità e flessibilità del cablaggio**: CAN è un bus seriale tipicamente implementato su un doppino intrecciato (schermato o meno a seconda delle esigenze). I nodi non hanno un indirizzo che li identifichi e possono quindi essere aggiunti o rimossi senza dover riorganizzare il sistema o una sua parte.

- **alta immunità ai disturbi**: lo standard ISO11898 raccomanda che i chips di interfaccia possano continuare a comunicare anche in condizioni estreme, come l'interruzione di uno dei due fili o il cortocircuito di uno di essi con massa o con l'alimentazione.

- **elevata affidabilità:** la rilevazione degli errori e la richiesta di ritrasmissione viene gestita direttamente dall'hardware con cinque diversi metodi (due a livello di bit e tre a livello di messaggio).



Introduzione

- **confinamento degli errori**: ciascun nodo è in grado di rilevare il proprio malfunzionamento e di autoescludersi dal bus se questo è permanente. Questo è uno dei meccanismi che consentono alla tecnologia CAN di mantenere la rigidità delle temporizzazioni, impedendo che un solo nodo metta in crisi l'intero sistema.

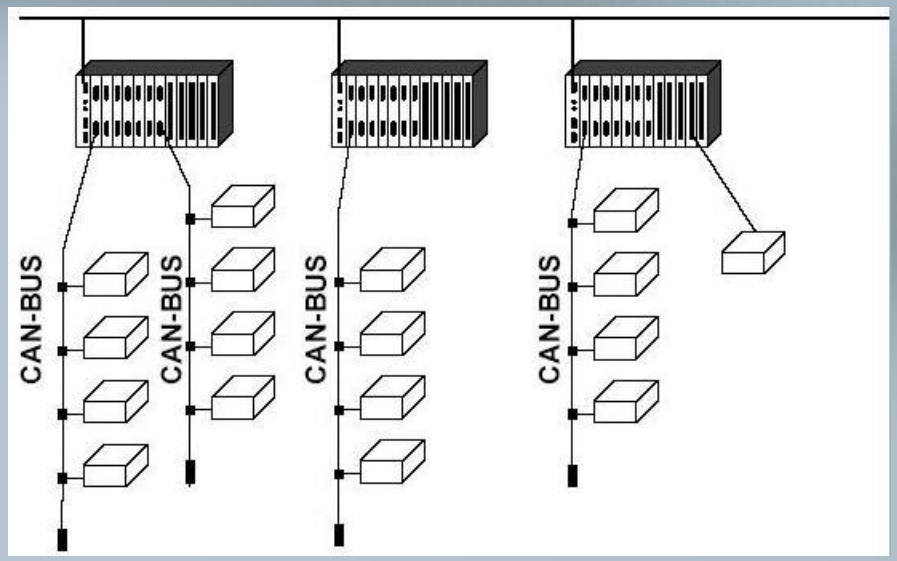
- **maturità dello standard**: la larga diffusione del protocollo CAN in questi venti anni ha determinato un'ampia disponibilità di chip rice-trasmettitori , di microcontrollori che integrano porte CAN, di tools di sviluppo, oltre che una sensibile diminuzione del costo di questi sistemi. Questo è molto importante per far sì che uno standard si affermi nell'ambito industriale.

ECU BOSH con supporto CAN

*E' grazie ai vantaggi che offre, che il can bus, trova grande impiego in applicazioni real-time per **auto**. Qui sopra c'è una **ECU** (Engine Control Unit) che comunica con il resto dei sensori/attuatori, mediante CAN.*

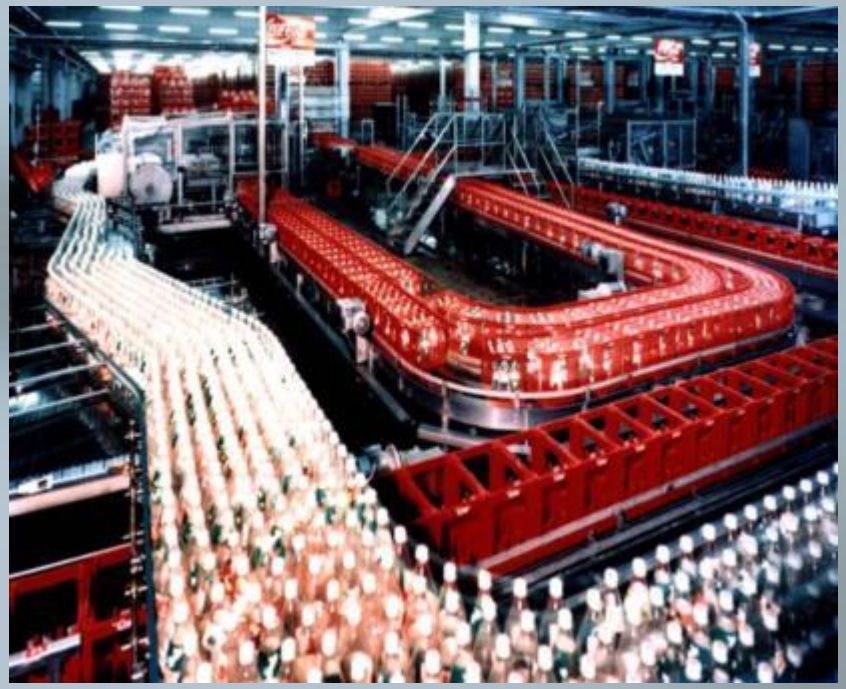
Introduzione

*Come si è già capito, stiamo parlando di uno **standard forte collaudato**. Facile da gestire e economico nella struttura, il CAN BUS si stà affermando sempre più in un'epoca dove **costi e sicurazza** sono due prerogative sempre in conflitto tra di loro.*



AFFIDABILITA’

- Il bus can ha un’incredibile capacità di riconoscere gli **errori**. La probabilità che un messaggio sia corrotto e non riconosciuto come tale, è praticamente nulla.



- E’ stato calcolato che una rete basata su CAN bus a 1 Mbit/s, con un’utilizzazione media del bus del 50%, una lunghezza media dei messaggi di 80 bit e un tempo di lavorazione di 8 ore al giorno per 365 giorni l’anno, avrà un errore non rilevato ogni 1000 anni. **Praticamente la rete non è soggetta ad errori per tutta la durata della sua vita**. Questo è il maggior punto di forza di questo bus.

COMUNICAZIONE

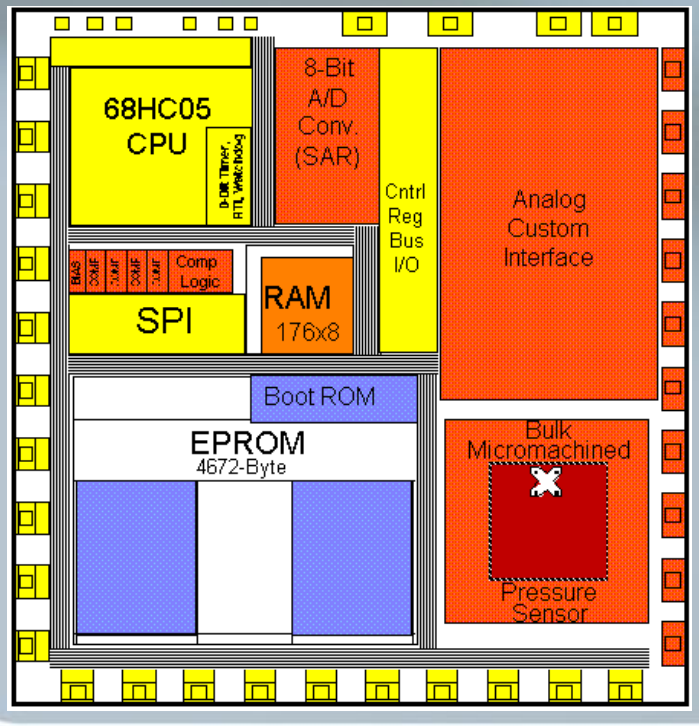
- La comunicazione, nel CAN BUS, avviene tramite **dispositivi intelligenti**, ovvero sensori o attuatori in grado di produrre dati autonomamente per poi immetterli sul BUS.
- Inoltre, questa tipologia di apparecchiature, è in grado di **richiedere e utilizzare i dati** prodotti da un altro dispositivo intelligente.

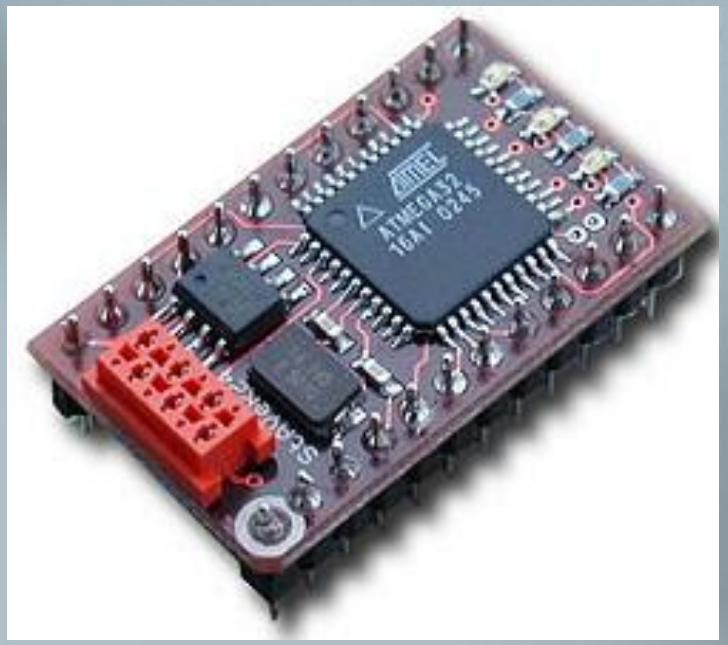
COMUNICAZIONE - Sensori

- I sensori intelligenti, prima di inviare un dato, svolgono determinati compiti:
  - **Amplificazione** del piccolo segnale d'uscita dal sensore vero e proprio
  - Traslazione die segnale in un range opportuno per la **conversione A/D**
  - **Elaborazione** dei dati
  - **Emissione** dei dati su bus

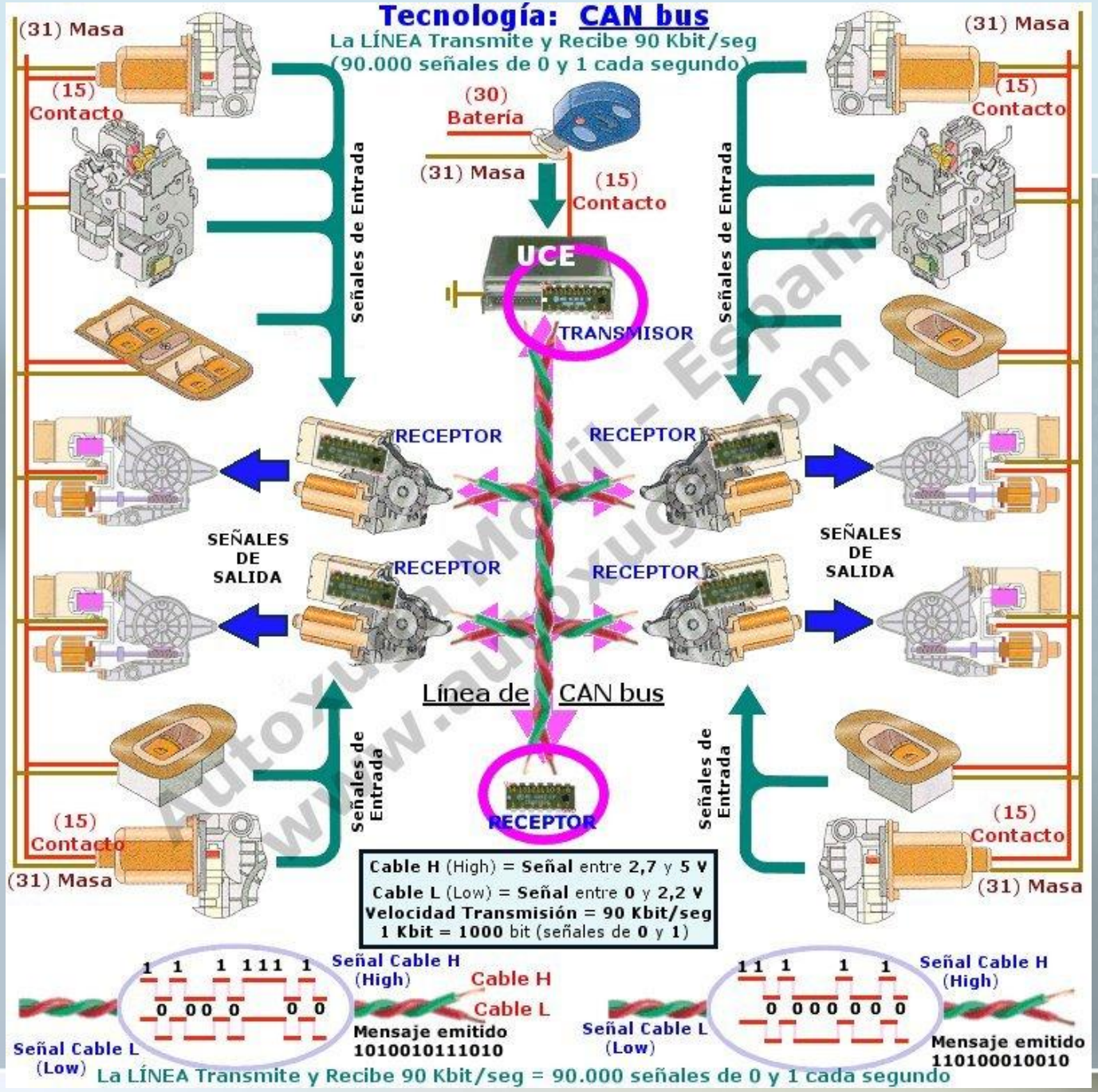
COMUNICAZIONE - Sensori

- Tipicamente un **nodo intelligente** ha un suo processore con una sua memoria. Esso è in grado di gestire ed elaborare autonomemente i dati. I singoli microcontrollori, sono provvisti di ram, eeporom, convertitore A/D e di un'interfacciamento con i dispositivi esterni. Oggi, questa tecnologia, è **economicamente** alla portata di tutti.



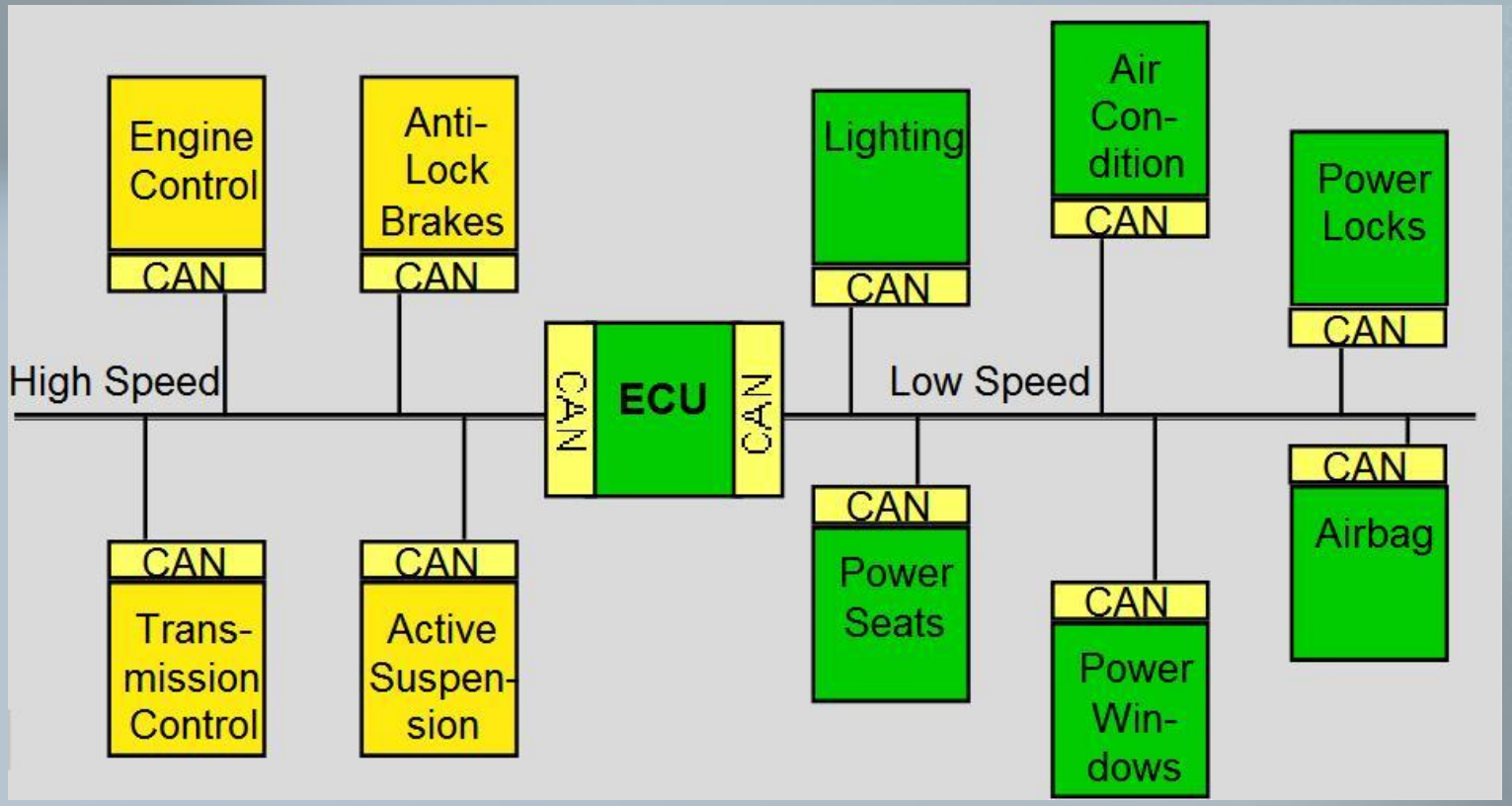


CAN IN AUTO


Tecnología: CAN bus
La LÍNEA Transmite y Recibe 90 Kbit/seg
(90.000 señales de 0 y 1 cada segundo)
(31) Masa
(15) Contacto
(30) Batería
UCE
TRANSMISOR
Señales de Entrada
RECEPTOR
SEÑALES DE SALIDA
Línea de CAN bus
Cable H (High) = Señal entre 2,7 y 5 V
Cable L (Low) = Señal entre 0 y 2,2 V
Velocidad Transmisión = 90 Kbit/seg
1 Kbit = 1000 bit (señales de 0 y 1)
Señal Cable H (High)
Cable H
Cable L
Señal Cable L (Low)
Mensaje emitido
1010010111010
Mensaje emitido
110100010010
La LÍNEA Transmite y Recibe 90 Kbit/seg = 90.000 señales de 0 y 1 cada segundo

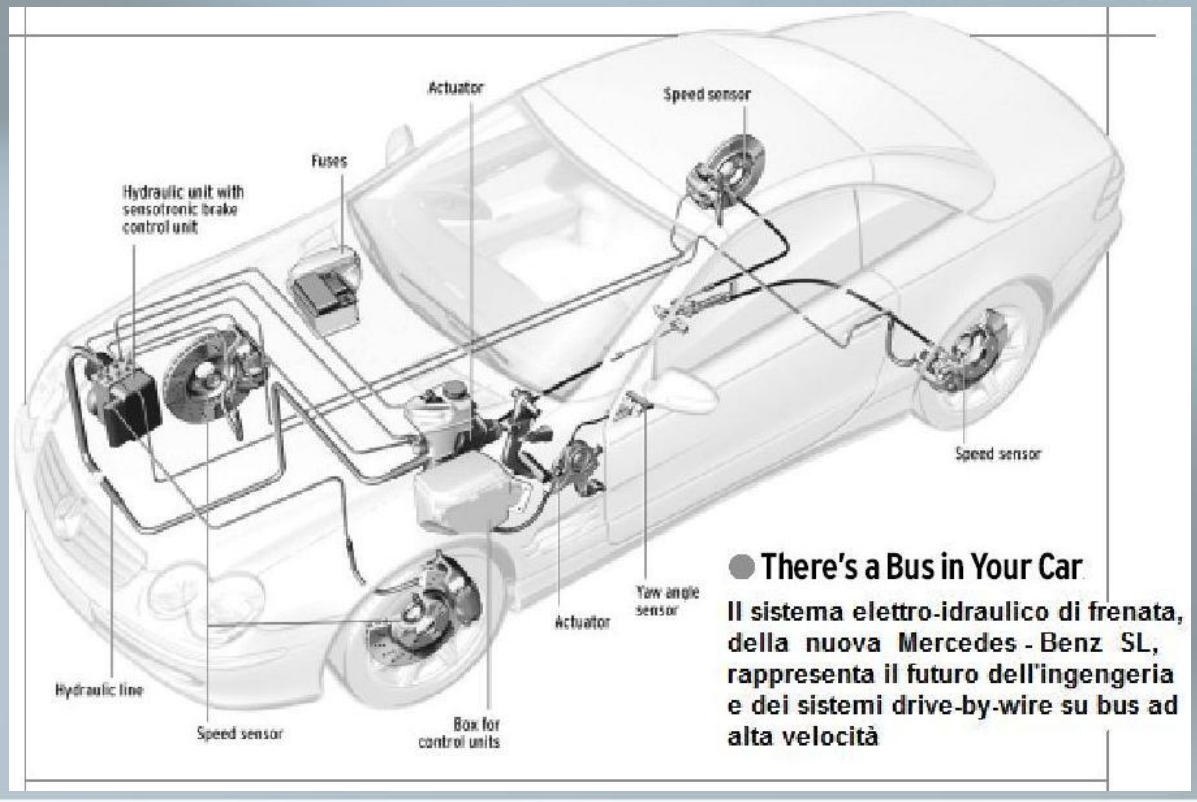
COMUNICAZIONE - Esempi

- Questi sono alcuni **esempi** degli apparati che il can bus è in grado di mettere in comunicazione tra loro. Questo esempio è applicato al mondo **automobilistico**.(notare come le **periferiche critiche** sia state collegate sul bus più **veloce**)



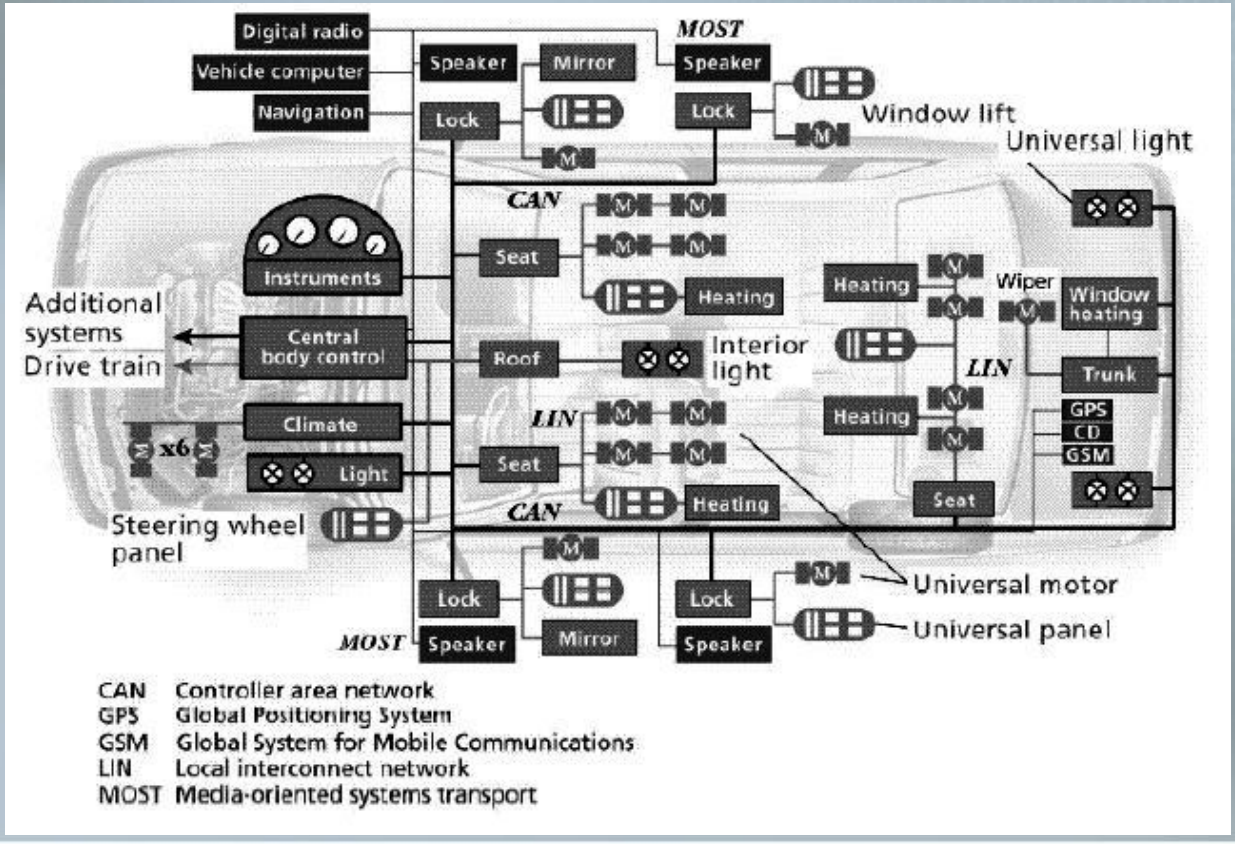
COMUNICAZIONE - Caratteristiche

- Questo BUS ha capacità "**Multi-Master**", ovvero tutti i nodi della rete posso trasmettere e più nodi della rete possono chiedere il canale trasmissivo contemporaneamente.
- I nodi di una rete CAN non sono caratterizzati da indirizzi di rete nel "senso convenzionale", ma i messaggi vengono instradati in base alla criticità del dispositivo mediante **un'identificativo**.



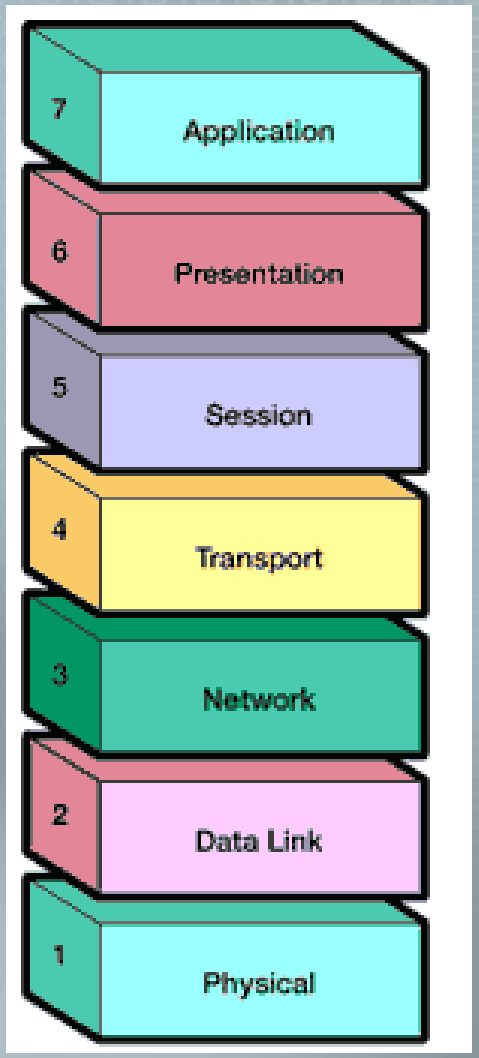
COMUNICAZIONE - Caratteristiche

- In base all'identificativo, il ricevente, può scegliere se **processare** il dato o meno.
- L'identificativo, inoltre, determina la **priorità** che il dispositivo ha nella competizione per l'accesso al bus.

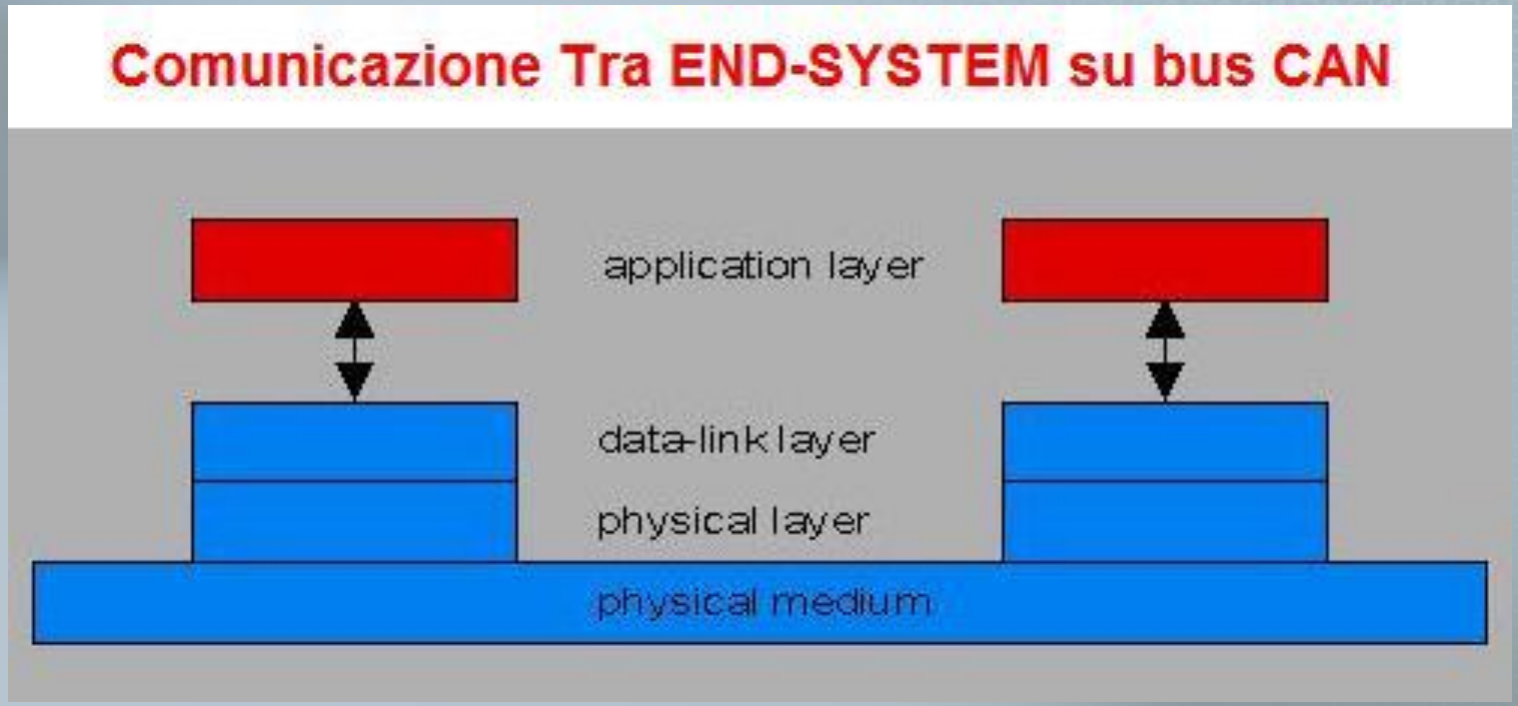


Il BUS CAN secondo il modello ISO-OSI

- Facendo riferimento alla schematizzazione in livelli definita dall'ISO (International Standard Organization) col progetto OSI (Open System Interconnection) si può ritenere che il Bus CAN implementi il **Physical Layer ed il Data Link Layer**, ovvero i **due livelli più bassi della pila**, come è logico per un protocollo più vicino ad un **Bus di Campo** che ad una rete informatica del tipo **Ethernet**.

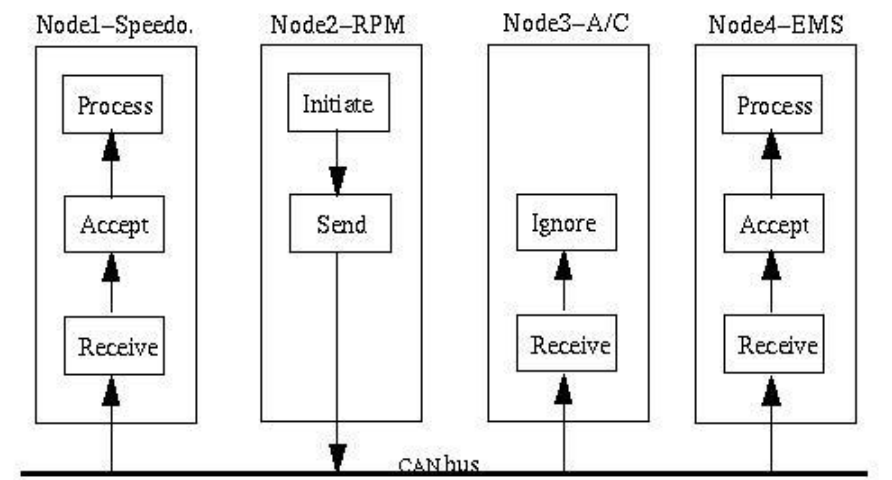


Il BUS CAN secondo il modello ISO-OSI



Il BUS CAN secondo il modello ISO-OSI

Il **Data Link Layer** è a sua volta implementato attraverso altri due livelli: **l'Object Layer ed il Transfer Layer** . Il primo si occupa del ***filtraggio dei messaggi*** *arrivati* . In una comunicazione broadcast tutti i nodi ricevono gli stessi pacchetti: nell'Object Layer si scartano quelli non rilevanti per il nodo considerato. Inoltre il livello di oggetto, si occupa della *della* ***gestione dei messaggi da trasmettere*** *e dell'interfaccia con l'Application Layer* .



LIVELLO DI OGGETTO

- filtraggio dei messaggi
- interpretazione dei messaggi

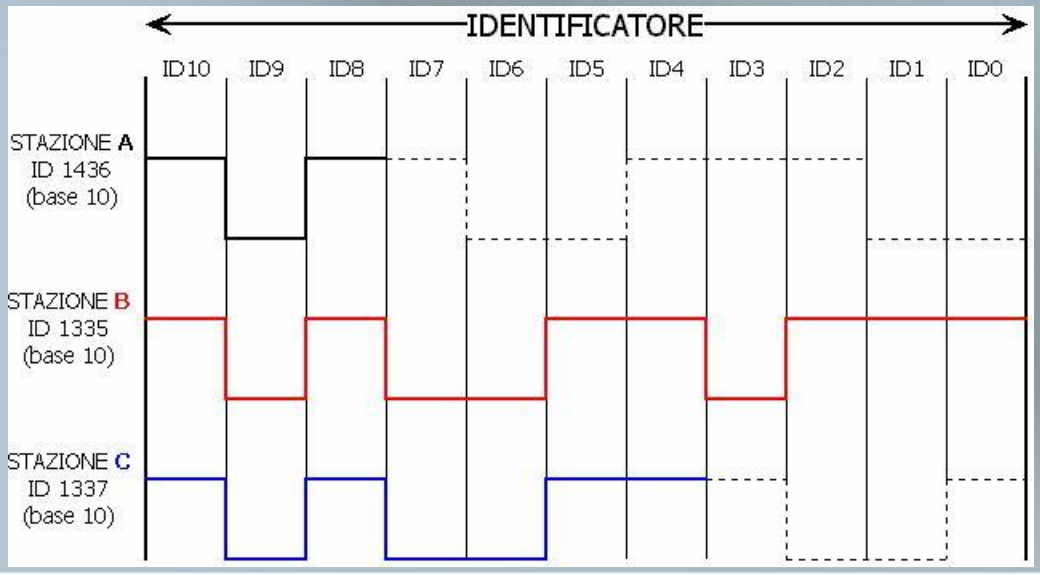
LIVELLO DI TRASFERIMENTO

- isolamento dei guasti
- rilevamento di errori
- segnalazione di errori
- acknowledgement
- arbitraggio
- suddivisione in pacchetti
- sincronizzazione

Il BUS CAN secondo il modello ISO-OSI

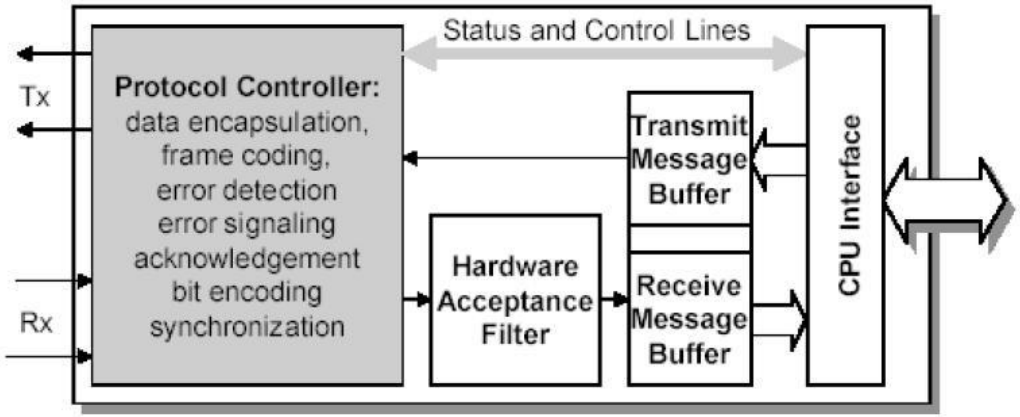
- Il **Data Link Layer**, ha l'importante compito **dell'arbitraggio nella competizione** per la contesa del canale trasmissivo, da parte dei vari nodi can che, contemporaneamente, ne richiedono l'utilzzo.
- Quando 2 o più nodi stanno trasmettendo contemporaneamente, il conflitto è risolto con un meccanismo di ARBITRAGGIO che non ci siano né perdite di informazioni, né di tempo.
- Durante ogni fase di arbitraggio, ogni stazione trasmittente, confronta il livello del bit trasmesso con il livello monitorato sul canale.

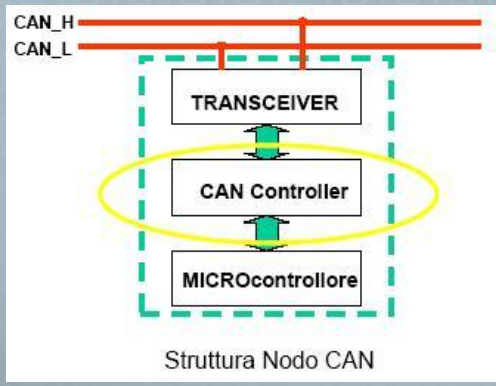
  Se i due bit coincidono il nodo continua la trasmissione. Quando il livello **associato al bit è recessivo, mentre sul canale si riscontra un livello dominante, l'unità interrompe immediatamente la trasmissione**



Livello oggetto e livello di trasferimento

- Le proprietà **dell'Object Layer** dipendono dal particolare hardware che lo implementa. **L'IC** che commercialmente va per la maggiore è il **SJA1000** della Philips Semiconductor (**CAN controller**). Le caratteristiche del **Transfer Layer** costituiscono invece il nucleo del protocollo CAN e sono quindi **rigidamente specificate.**





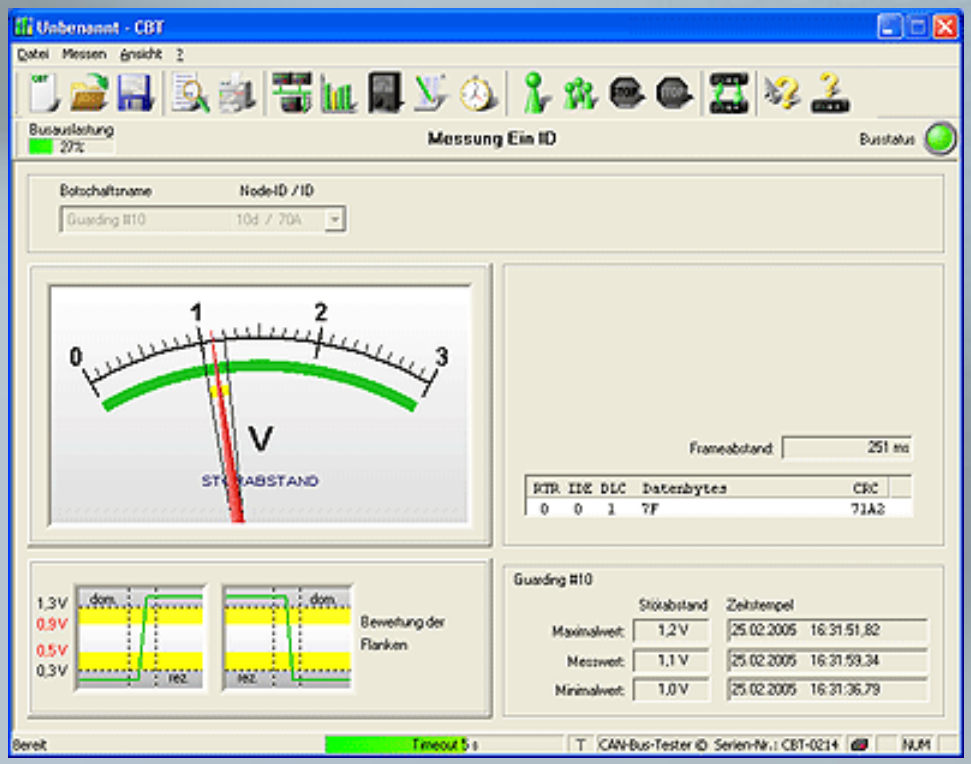
Struttura Nodo CAN

Philips SJA1000 Stand-Alone CAN Controller

Il **Transfer Layer** definisce le **modalità di trasferimento**: *formato dei messaggi, arbitraggio, segnalazione e correzione degli errori, esclusione dei nodi mal funzionanti, filtraggio messaggi.*

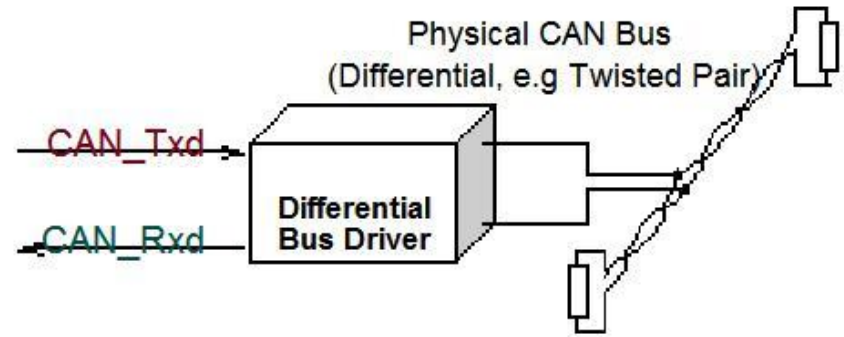
Livello applicazione

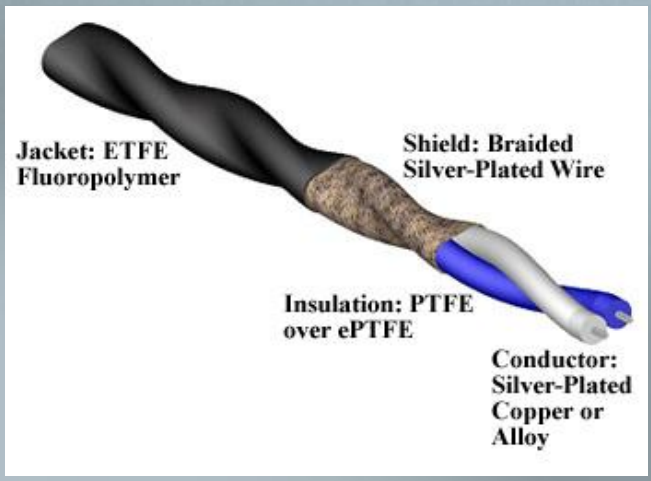
- La definizione **dell'Application Layer** è infine lasciata interamente al progettista (**non è standardizzato**), al quale spettano i dettagli dell'interfacciamento degli utenti verso il bus



Il Physical Layer

- Parliamo ora **dell'ultimo livello della pila ISO-OSI** riguardante il CAN. Esso è standardizzato in accordo con **ISO 11898**.
- Secondo il modello ISO-OSI (+ISO11898) il livello fisico deve specificare:
- **il mezzo trasmissivo:** nel bus CAN deve essere un singolo canale **bidirezionale**, che può essere di tipo **differenziale** o a **cavo singolo e terra** (meno usato del primo). Solitamente si usa un doppino intrecciato, **schermato** o meno a seconda della rumorosità (elettrica e magnetica) dell'ambiente.



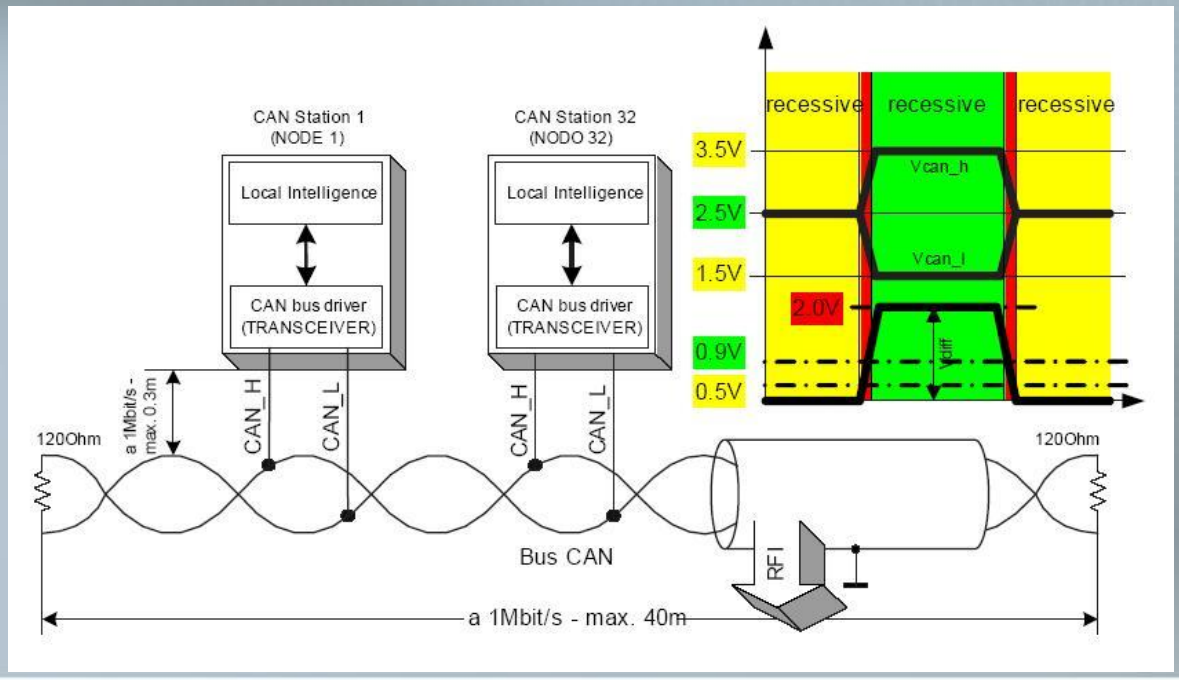


Il Physical Layer – (Differential Signal Communication)

- Un problema molto grande, nella trasmissione di segnali, sono i disturbi di tipo **RFI** (Radio Frequency Interferences). I disturbi RFI sono generati involontariamente (o volontariamente) dagli apparati tecnologici che oggi ci circondano. La prima linea di difesa contro le RFI è **schermare i cavi** con un conduttore metallico che poi viene messo a terra. Per compensare i limiti di questa tecnica, si utilizza la tecnica chiamata **DIFFERENCIAL**. Questa tecnica sfrutta due cavi, dove viene inviato rispettivamente il segnale trasmesso come positivo, e lo stesso segnale trasmesso in negativo. I disturbi, che penetrano il cavo, saranno presenti in entrambi i cavi e avranno valori uguali. Il ricevente non deve far altro che la differenza trai due segnali, in modo tale che la parte di rumore risulterà scomparsa.

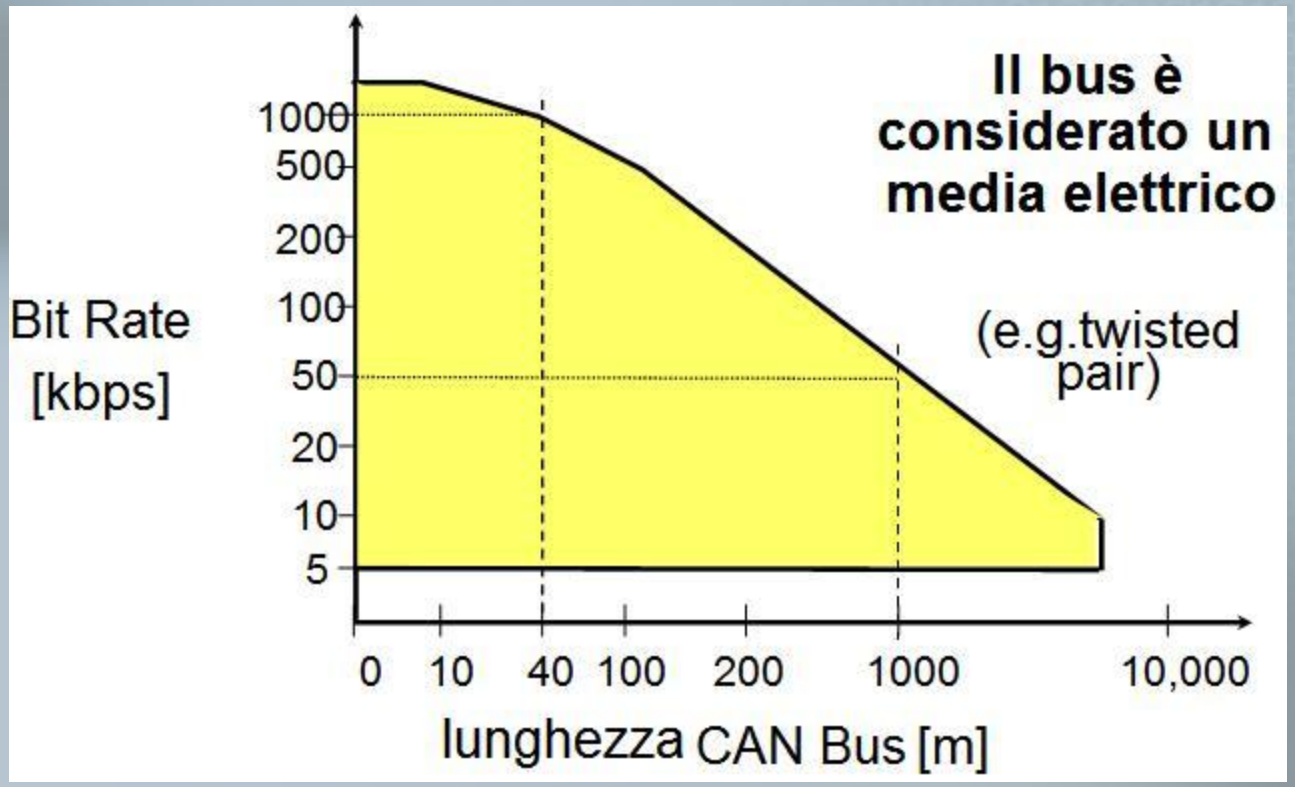
- **Questo metodo funziona bene sole se sui due cavi c'è lo stesso disturbo.** Per avere questo bisogna che i cavi occupino lo **stesso spazio**. Per arrivare a una legittima approssimazione, si avvolgono i due cavi insieme (**Twisted Pair**).

- **ESEMPIO:** Il doppino è composto da due linee, una denominate H e una L (high e low). Per rappresentare il bit 0, entrambe le linee restano a 2,5V (la loro diferenza è di 0 volts).Il valore logico 1 viene rappresentato portando la linea H a 3,5V (rispetto a massa) e la linea L a 1,5V (Con d.d.p. tra le due linee di 2 volts).



Il Physical Layer - Caratteristiche

- La **lunghezza** massima del bus dipende dalla **velocità** usata per la trasmissione (e vice-versa).

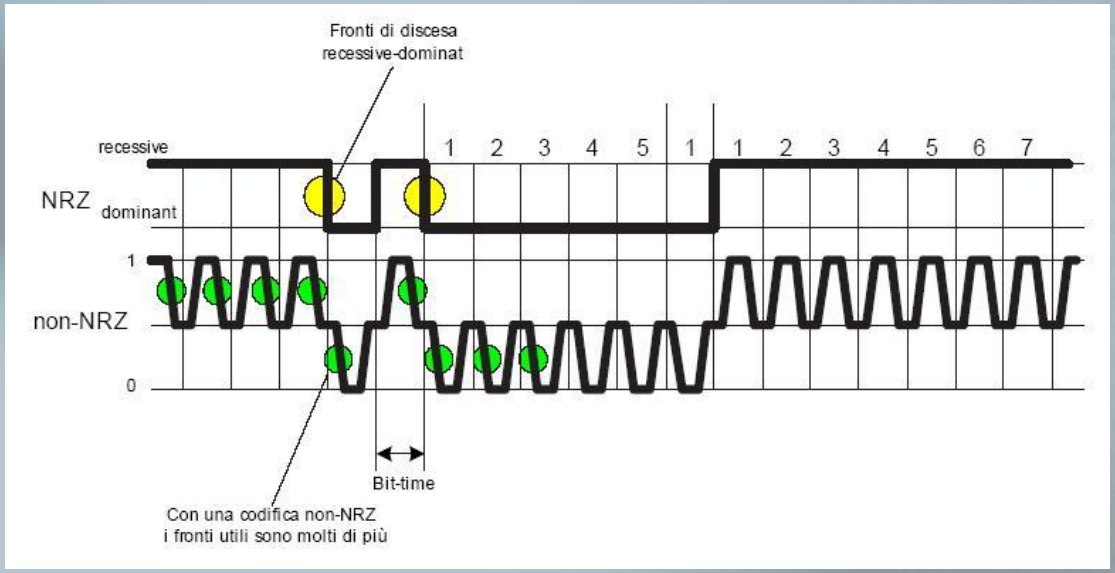


Oltre 1Mbit/sec entro 40m.

Il Physical Layer - Caratteristiche

La rappresentazione dei bit ed i livelli del segnale:

- il flusso di bit è codificato con il metodo **NRZ** (Non Return to Zero, ovvero i bit **non sono separati**) e questo porta ad un **problema di sincronizzazione** che esplicheremo più avanti).

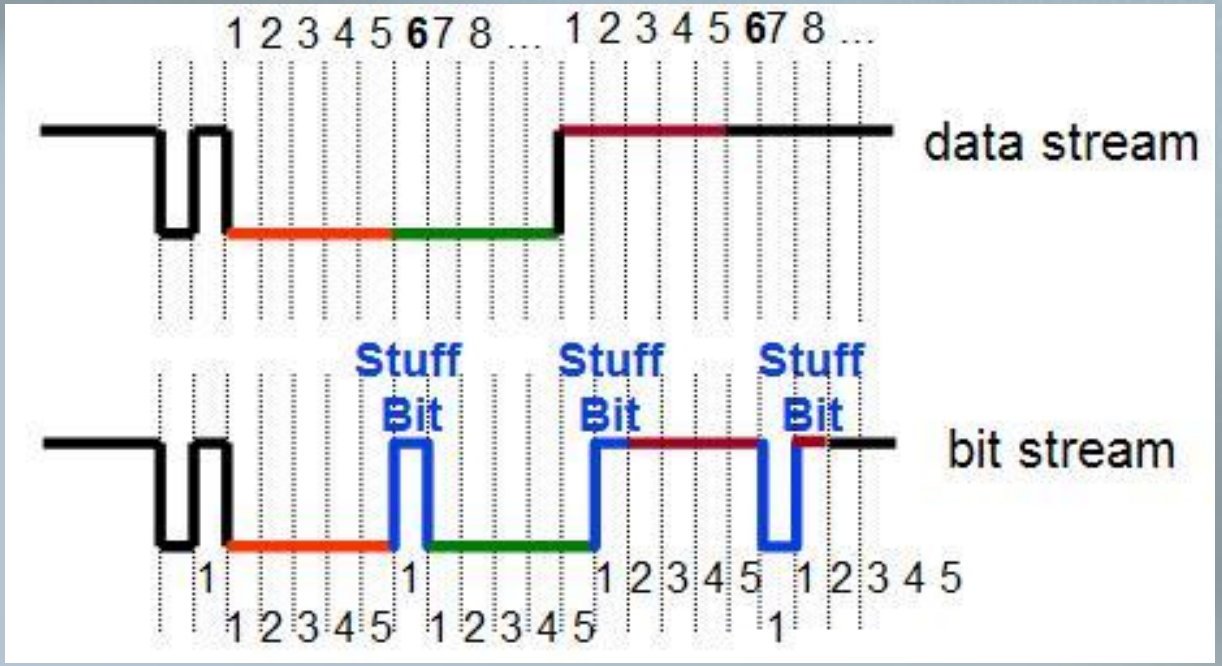


- I due livelli utilizzati sul bus sono detti "**dominante**" (d) e "**recessivo**" (r); questa definizione riguarda la possibilità di trasmissione contemporanea di un bit 'r' e di un bit 'd' : il valore che deve essere **presente sul bus in questo caso è '**d'. La corrispondenza fra 'd' ed 'r' ed i livelli logici '0' ed '1' **dipende dal tipo di cablaggio**, come riassunto nella seguente tabella:

Cablaggio wired-AND	'd'='0'	'r'='1'
Cablaggio wired-OR	'd'='1'	'r'='0'

Il Physical Layer - PLS

- Il **Physical Layer si divide** in tre sottogruppi:
  - **PLS** (Physical Signaling): Si occupa della **temporizzazione e sincronizzazione** dei segnali sul bus. Per la sincronizzazione utilizza la tecnica del **"Bit Stuffing"** che consiste nell'aggiungere, ogni volta che si hanno **5 bit consecutivi**, un bit di polarità inversa (Bit di Stuff). Prima di esere **scartato** il ricezione, viene utilizzato per la sincronizzazione.

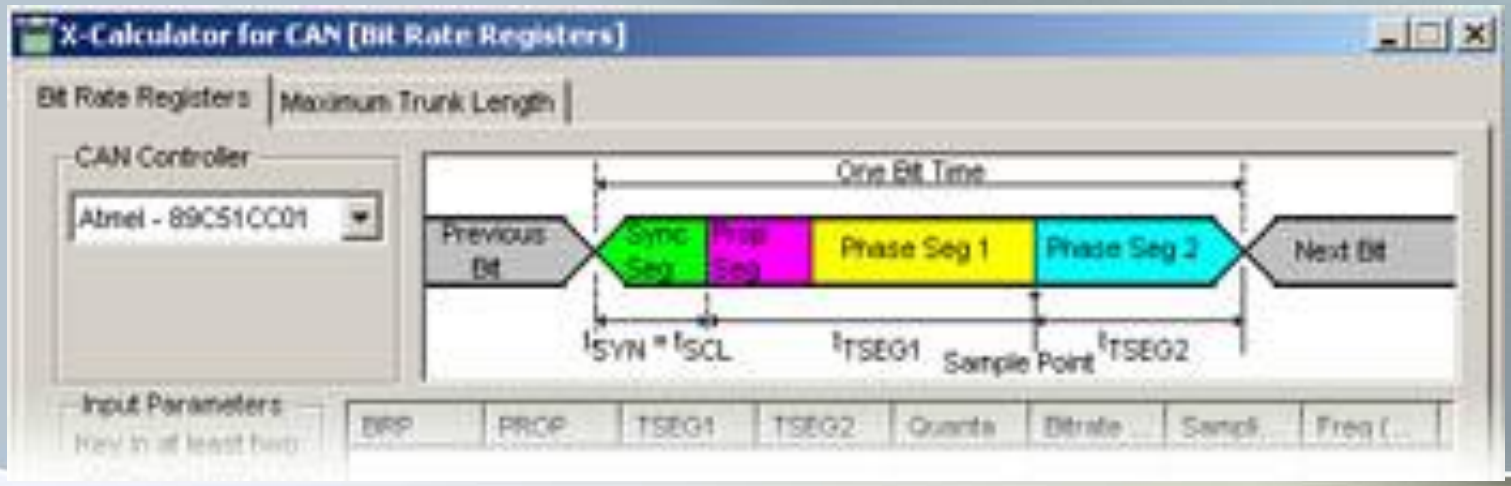


Il Physical Layer – PLS

- Il Flusso dei dati sul bus è determinata dal **Nominal Bit Rate** (NBR), definito come il numero di **bit trasmessi al secondo** in una trasmissione ideale; il suo **inverso è il Nominal Bit Timing**, che rappresenta il **tempo occupato da un singolo** bit ed è il reciproco del NBR:

$$NBT = \frac{1}{NBR}$$

- Il **NBT** può essere diviso in **4 intervalli temporali di durata regolabile**, espressa come **multiplo di un quanto di tempo** (TIME QUANTUM). Il **TIME QUANTUM è a sua volta regolabile come multiplo dell'intervallo di un oscillatore locale, assunto come quanto minimo di tempo**: TIME QUANTUM = m * MINIMUM TIME QUANTUM:

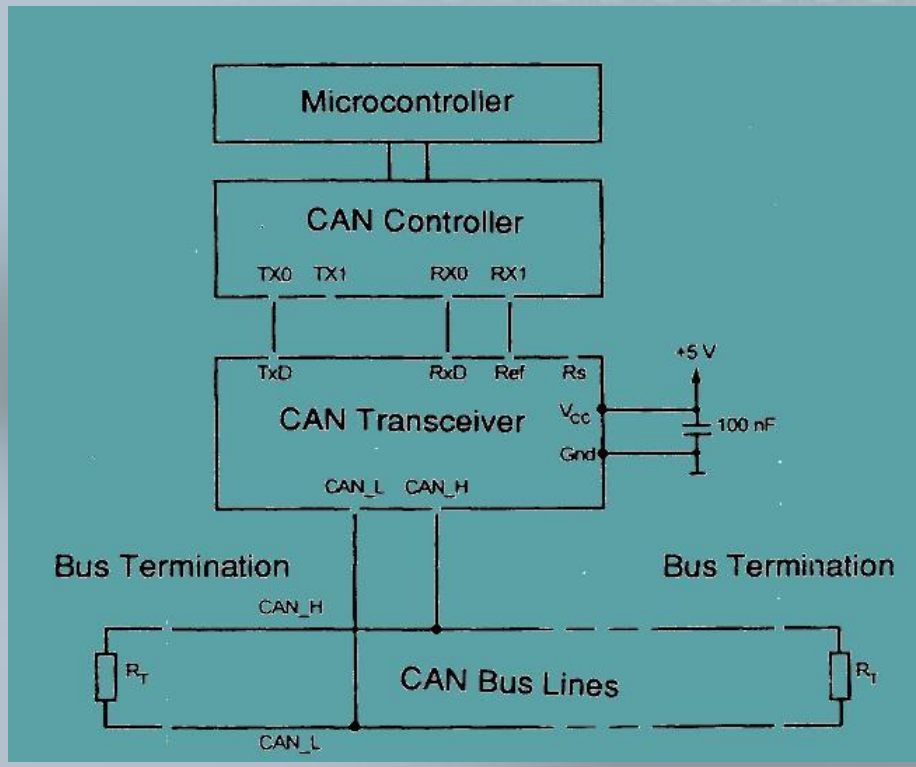


Il Physical Layer – PLS

- Qui è riportata la suddivisione del Bit Time:
  - **Synchronization Segment** (Sync_Seg): è la prima parte del bit time, che viene utilizzata per **sincronizzare** i vari nodi. Ha durata fissa e pari a 1 TIME QUANTUM.
  - **Propagation Time Segment** (Prop_Seg): viene introdotto per **compensare i ritardi** nella trasmissione e nella ricezione dei bit. Può durare da **1 a 8 TIME QUANTUM.**
  - **Phase Segment 1 e 2** (Phase_Seg1 e Phase_Seg2): sono i due segmenti terminali del bit-time e vengono regolati in modo che la transizione fra di essi, sulla quale i nodi leggono il livello del Bus (**Punto Di Campionamento**), rispetti le **esigenze di tutto l'hardware** connesso al bus.
    Phase_Seg1 può durare da 1 a 8 TIME QUANTUM, mentre Phase_Seg2 è deve avere la stessa durata di Phase_Seg1.

Il Physical Layer – PMA

- **PMA** (Physical Mediaum Attachment): Detta la **struttura** di ogni nodo CAN e come è connesso al Bus. Il nodo can è composto da **tre parti principali**. (vedi di seguito)
- **Transceiver**: Rileva lo stato del bus valutando la differenza di tensione tra CAN_H e CAN_L.
- **Can Controller**: Trasmette e riceve dati seriali dal microprocessore al bus (e viceversa).
- **Microcontroller**: E' il cuore del nodo can e gestisce tutte le operazioni che la periferica deve svolgere.



Il Physical Layer - MDI

- **MDI** (Medium Dependent Interface): Stabilisce le **caratterstiche elettriche** e fisiche standardizzate che devono avere:
- -Il cavo del bus: **Tipo di media**, l'impedenza che termina il bus (deve essere di **120Ω**), ritardo della linea, ecc…
- -**Il connettore** che collega il nodo (transceiver) al bus

Il Data Link Layer - Transfer Layer

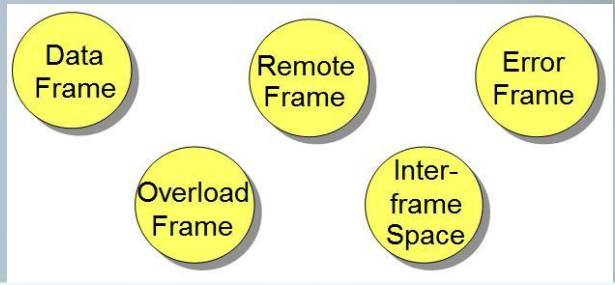
Il Transfer Layer si occupa di definire:

- il **formato** dei messaggi
- la gestione degli **errori**
- **l'arbitraggio**
- il **confinamento** dei nodi mal funzionanti
- la **validazione** dei messaggi

Questi punti verranno trattati nelle diapositive seguenti

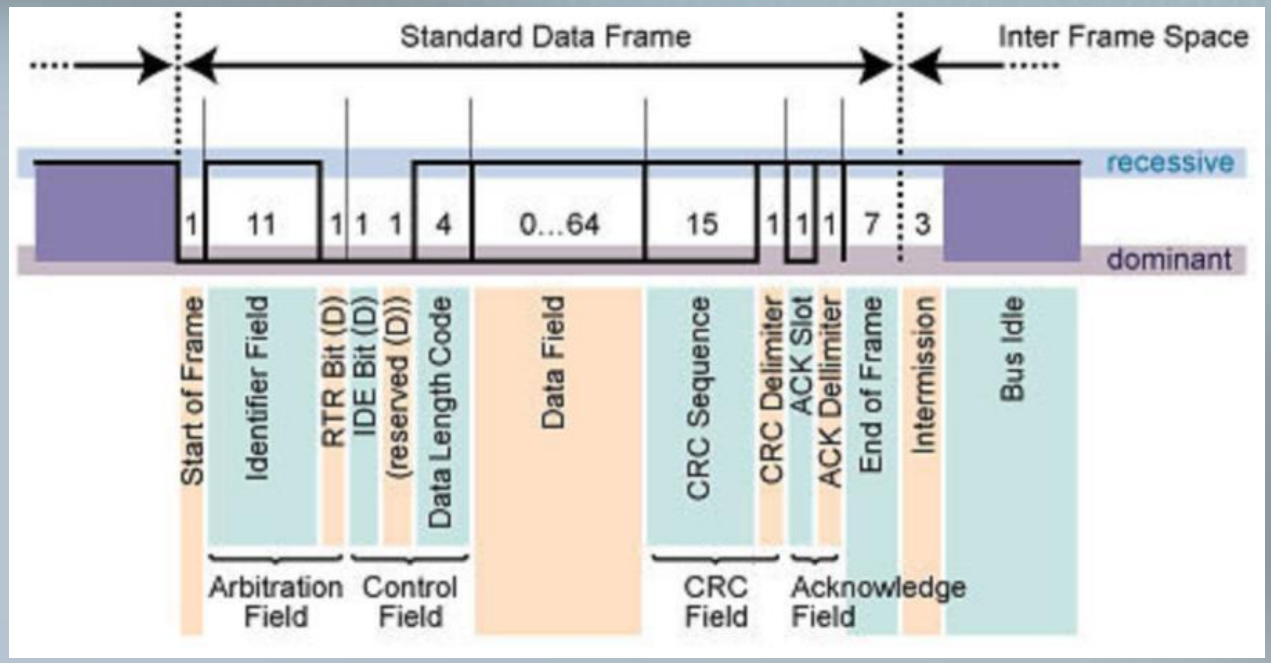
Formato dei messaggi

- **Nel protocollo CAN esistono cinque differenti strutture di messaggi:**
- **Data Frame** (D.F.) **:** è il tipo di messaggio più **diffuso** e permette la trasmissione dei dati da un nodo trasmettitore (TX) a tutti gli altri , che si comportano quindi come ricevitori (RX); ciascun nodo decide separatamente se ritenere rilevanti i dati ricevuti o se scartarli.
- **Remote Frame:** ha una struttura **simile al Data Frame**, ma è privo del campo dati; serve a sollecitare l'invio di un determinato D.F. da parte del nodo interrogato.
- **Error Frame:** viene inviato da un nodo che rivela un errore e **provoca la ritrasmissione** del messaggio da parte del nodo trasmettitore; poiché è sufficiente che un solo nodo segnali un errore per avere la ritrasmissione, il protocollo CAN prevede che ciascun nodo **monitorizzi il proprio stato di salute, autoescludendosi** in caso di tasso di errore elevato (fault confinement).
- **Overload Frame**: viene inviato da un nodo che risulta "**busy**" per ritardare la trasmissione del pacchetto successivo.
- **Interframe Space**: precede ogni Data e Remote Frame e ha una funzione separatrice.



Data frame

- **E' costituito dai seguenti 7 campi:**
  - **Start of Frame** (SoF): è costituito da un solo bit dominante e segnala l'inizio di un messaggio di tipo D.F.o di tipo R.F.. Ha anche una funzione di sincronizzazione per tutti gli altri nodi che riconoscono l'inizio della trasmissione.
  - **Arbitration Field**: è costituito **dall'identificatore** del **contenuto** del messaggio più un bit **RTR** (Remote Transmission Request) **che distingue fra D.F.e R.F.**. L'identificatore è di **11 bit** nel protocollo CAN 2.0A ("Standard CAN") e di **29 bit** nel CAN 2.0B ("Extended CAN"), come indicato in figura. Il bit RTR è dominante nel caso di D.F.e recessivo nel caso di R.F.: in questo modo se si ha contemporaneamente la richiesta di un dato e la trasmissione dello stesso, la richiesta viene abortita.



Data frame

- **Control Field**: è costituito da 6 bit, di cui 4 servono a **specificare il numero di byte** di cui è composto il messaggio vero e proprio (Data Field) e 2 sono riservati per future espansioni del protocollo. La codifica del **DLC** (Data Lenght Code) è indicata nella seguente **tabella**:

Numero di bytes nel DATA FIELDS	DATA LENGHT CODE			
	DLC3	DLC2	DLC1	DLC0
0	D	D	D	D
1	D	D	D	R
2	D	D	R	D
3	D	D	R	R
4	D	R	D	D
5	D	R	D	R
6	D	R	R	D
7	D	R	R	R
8	R	D	D	D

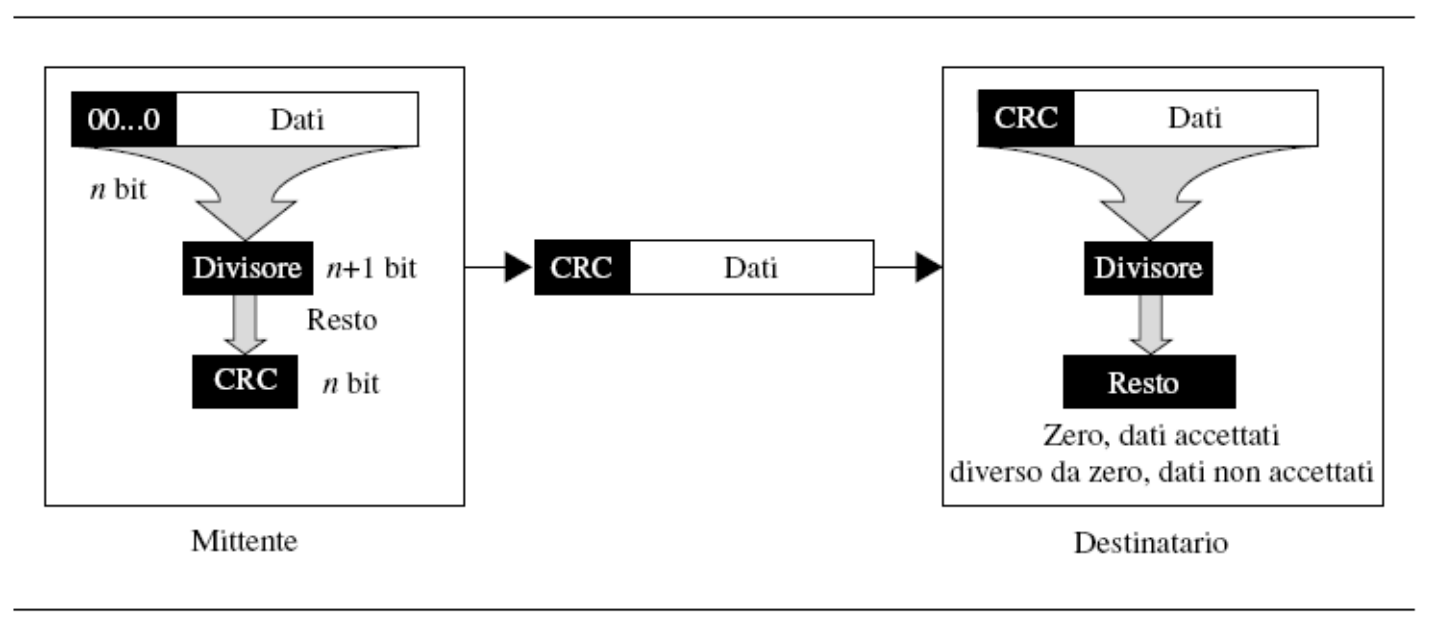
N.B. I valori di "D" ed "R" dipende dal tipo di cablaggio utilizzato (wired-AND o wired-OR)

Data frame

- **Data Field**: contiene i **dati** veri e propri, che vanno da un **massimo di 8 bytes (64bit)** ad un minimo di 0 , come indicato nel DLC. I bytes vengono inviati dal più significativo al meno significativo.
- **CRC Field**: è costituito da **16 bits**, di cui i primi 15 contengono la sequenza di controllo (cyclic redundancy check) mentre l'ultimo è un bit recessivo di delimitazione. Se il codice di ridondanza ciclica non rivela la presenza di un errore, il nodo mette un bit recessivo nel campo ACK del Data Frame attuale.
- **ACK Field**: è costituito da un bit detto ACK Slot ed uno di delimitazione (ACK Delimiter). Sono entrambi inviati come **recessivi** ma ACK Slot viene **sovrascritto** come **dominante** da ogni nodo che **riceve il messaggio in maniera corretta**. In questo modo il TX sa che almeno un nodo ha ricevuto il messaggio corretto (MA Ciò NON SIGNIFICA CHE SIA ARRIVATO AL DIRETTO INTERESSATO).
- **End of Frame** (EoF): è costituito da 7 bits recessivi che indicano la fine del Frame.

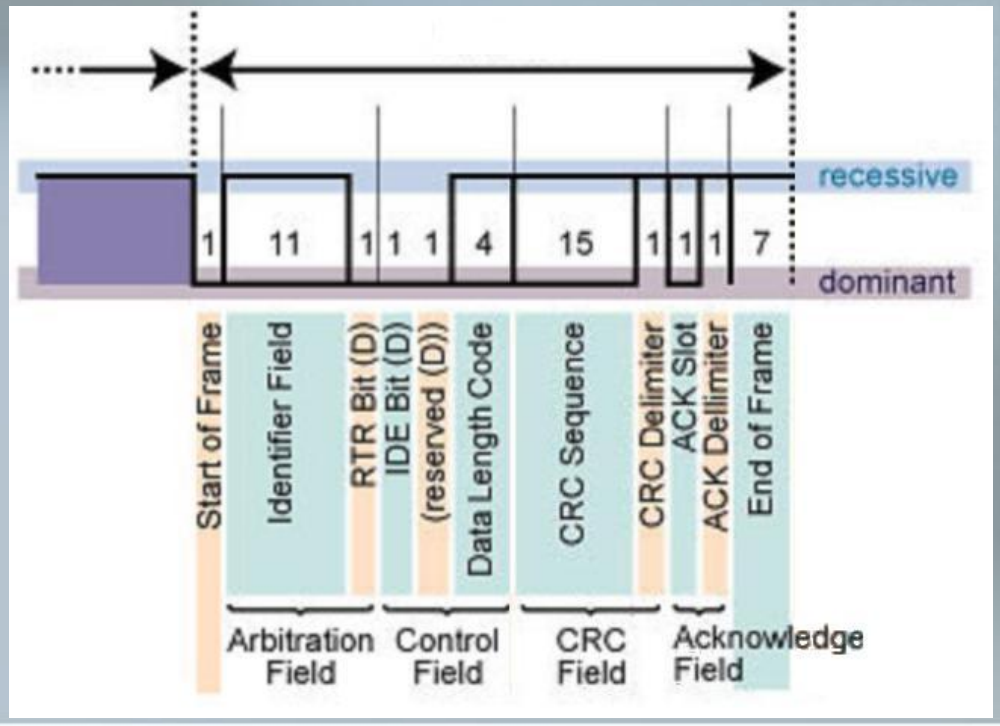
Data frame - CRC

- Il campo **CRC** usa la seguente tecnice per rilevare errori nei Frame:
  - Una **sequenza** di bit ridondanti, detta CRC, viene **aggiunta** alla fine dell'unità dati in modo tale che l'intera sequenza costituisca un numero binario **esattamente divisibile** per un altro numero binario prefissato. Il destinatario divide la sequenza binaria ricevuta per il numero binario prefissato e accetta i dati in caso **di resto nullo**, mentre li rigetta nel caso in cui il resto risulti diverso da zero.



Remote frame

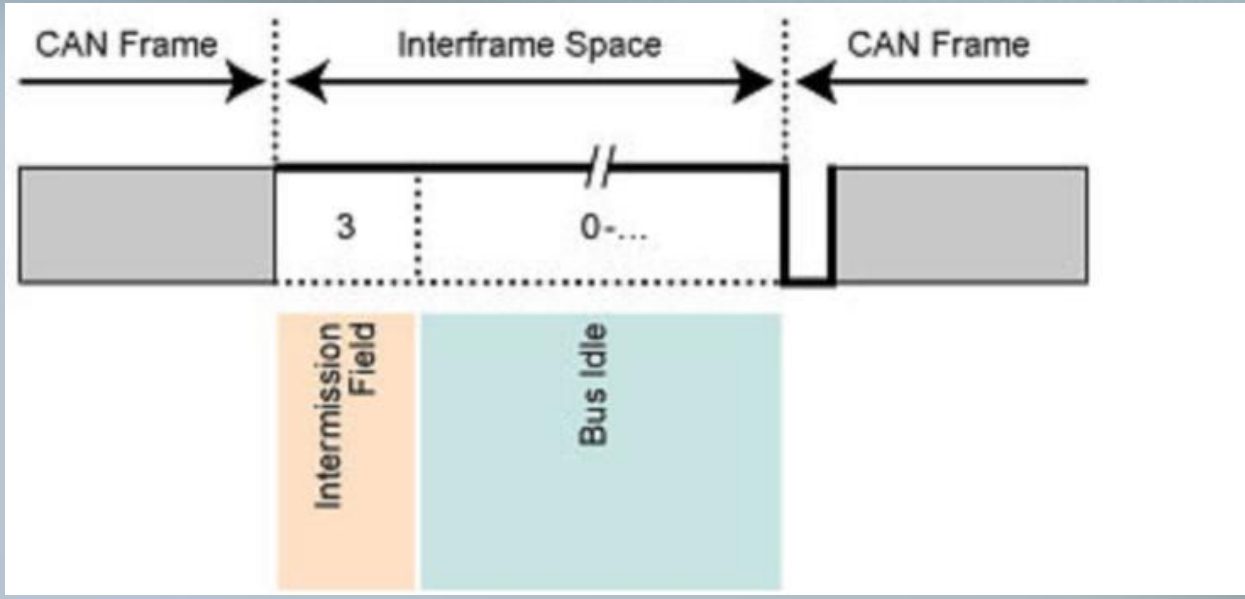
- Il secondo formato di messaggi, per il protocolla CAN, è il "**Remote Frame**".
- Il Remote Frame viene inviato da un nodo ricevitore che **necessiti di un dato**. **Il tipo di dato** desiderato viene specificato nel campo "**identificatore**". Il nodo trasmettitore interessato risponderà con un Data Frame contenente l'informazione richiesta nel campo Data Field.
  La struttura del messaggio è molto simile a quella di un Data Frame: i campi si riducono dai 7 a 6 per **l'assenza del Data Field** e il bit RTR (che distingue il Data Frame dal Remote Frame) è recessivo anziché dominante.



Interframe Space

*Il terzo tipo di messaggio per il CAN BUS è il "**Interframe Space**".*

- L'interframe Space Serve a **separare** due D.F. o R.F.s, mentre non è necessario nel caso di Overload Frame o Error Frame, che possono quindi anche essere inviati consecutivamente sul bus. In termini di struttura non si tratta di un vero e proprio messaggio, ma di una **sequenza indefinita di bit recessivi**, che viene comunque suddivisa in campi per specificare gli eventi che sono ammessi nei rispettivi bit-time:



Interframe Space

- **Intermission**: sono i primi tre bit recessivi, che possono essere **interrotti** solo da un **Overload Frame**; in questo caso i bit recessivi del campo seguente non sono trasmessi e si passa al campo Overload Flag di un Overload Frame. Non è consentita invece l'interruzione da parte di un D.F.o di un Overload Frame.

- **Bus Idle**: **è una sequenza di bit recessivi** di lunghezza non specificata, **che permane finché non si ha lo Start of Frame** di un nuovo D.F.o R.F.. Questa trama viene infatti interpretata dai nodi come un segnale di bus libero, per cui ciascuno può iniziare la trasmissione quando lo ritiene necessario.

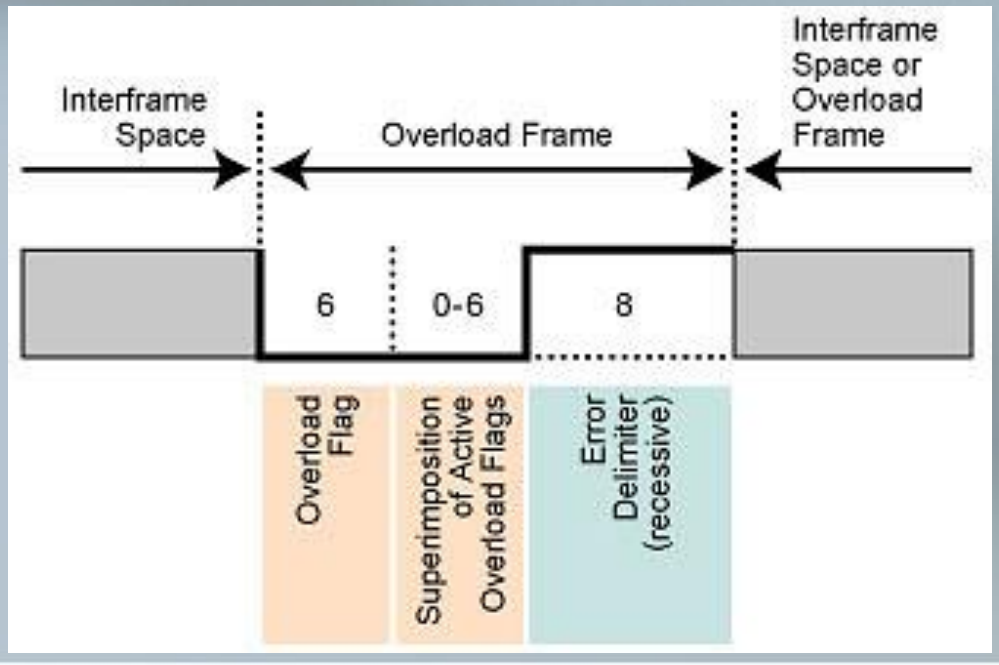
- **Suspend Trasmission:** si tratta di una diversa interpretazione degli 8 bit recessivi seguenti l'Intermission da parte di un nodo error passive. Se infatti un nodo è **frequentemente soggetto ad errori** deve **aspettare** un tempo pari ad **8 bit** prima di prendere possesso del bus per una nuova trasmissione.

Overload Frame

- Viene inviato in una delle seguenti situazioni e serve a **ritardare l'invio del D.F.o del R.F**. successivo:
- *- condizione interna di un nodo che rileva di essere "**busy**" (occorre più tempo per processare i dati prima di riceverne altri) e non può quindi ricevere correttamente il Frame successivo.*
- *- ricezione di un bit dominante in una fase di interframe : essendo un Interframe Space costituito da tutti bit recessivi, questo evento viene interpretato come la condizione di overload di un altro nodo, cui rispondere con un Overload Frame.*
- Poiché è sufficiente che un nodo invii un Overload Frame perché la comunicazione di tutti gli altri venga automaticamente ritardata, anche per questo tipo di messaggi il protocollo CAN prevede un meccanismo di **confinamento** dei **nodi** più **lenti**: **non possono infatti essere inviati più di 2 Overload Frame successivi** da parte dello stesso ricevitore. Inoltre i moderni chip di interfaccia non prevedono la possibilità di inviare questo tipo di messaggi, che restano una prerogativa dei chip **obsoleti**.

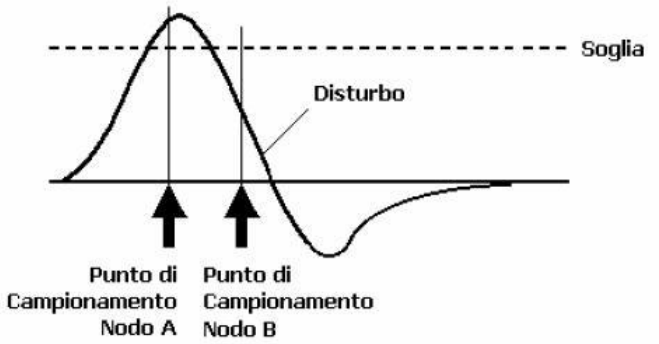
Overload Frame

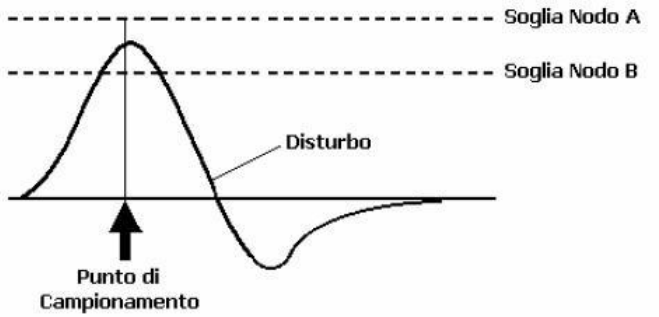
- La struttura di un Overload Frame è molto semplice, in modo da renderla facilmente riconoscibile; sono presenti due soli campi :
  - Overload Flag: è costituito da **6 bit dominanti** e può "sovrascrivere" un **Interframe Space** se lo interrompe entro il campo Intermission (primi tre bit): se invece un bit dominante arriva dopo questo termine viene interpretato come Start of Frame di un D.F. o di un R.F.
  - Overload Delimiter: è costituito da **8 bit recessivi**, cioè lo stesso formato del campo Error Delimiter di un Error Frame.

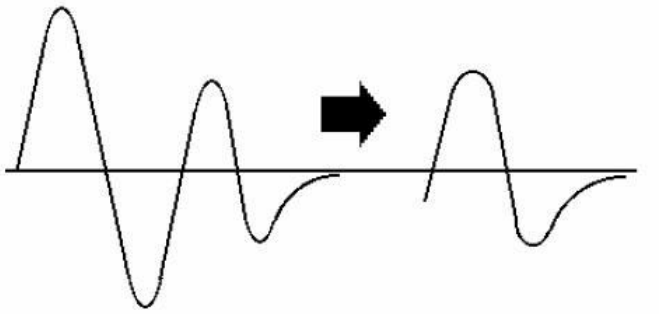


Error Frame – Errori nella TX/RC

- Lungo il bus ci possono essere diverse cause per le quali si verifica un'erroredi trasmissione o di ricezione. Le tre principali sono:
  - Ogni nodo può avere un **diverso punto di campionamento** e quindi a causa di un distrurbo può verificars che due nodi leggano due livelli logici diversi.
  - Due diversi nodi possono avere **diverse soglie di riconoscimento dei livelli logici**.
  - Il segnale lungo la linea si **attenua** e quindi (su tratti lunghi) si può avere una **degradazione** tale da generare un'errore in lettura.







Error Frame

- Viene **inviato da un nodo** che rivela la presenza di un errore in un **Remote Frame o in un Data Frame**. E' composto da due campi:

- **Error flag:** è costituito da 6 bit che possono essere dominanti o recessivi, dando luogo a due tipi di Error Flag (questo frame **viola volontariamente la regola di bit stuffing**, in modo tale che tutte le stazioni rilevino un'errore e **spediscano** anch'esse un error flag.). Tipi di error Flag:

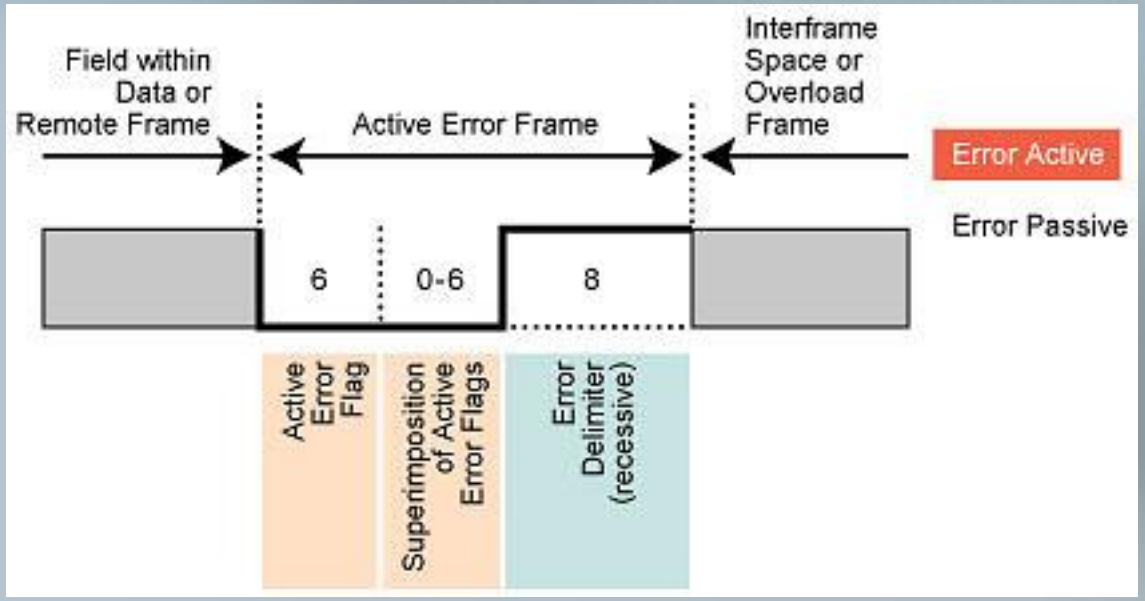
  - *Active Error Flag*, **costituito da 6 bit dominanti** e inviato se il sistema di confinamento dei guasti assegna al nodo uno stato **Error Active.** I bit dominanti possono quindi "**distruggere" il traffico** presente sul bus, cioè impedire la terminazione di un D.F. o di un R.F. rilevati come **errati**.

  - *Passive Error Flag*, **costituito da 6 bit recessivi** possono quindi essere sovrascritti da quelli di un Active Error Flag ma non possono sovrascrivere i campi ACK ed EOF di un D.F. Questo impedisce a questo tipo di nodi di "distruggere" il traffico presente sul bus in caso di un loro errore.

Error Frame

- Per la **contemporanea** trasmissione di tutti gli **Error Flag** si può arrivare ad un **massimo** di **12 bit** dominanti consecutivi (un nodo si accorge dell'error flag sul bus, solo al suo ultimo bit): la transizione di un bit recessivo viene interpretata come termine della trasmissione degli Error Flag da parte di tutti i nodi ed il primo bit recessivo è già riconosciuto come Error Delimiter.

- Error Delimiter: è costituito da 8 bit recessivi



GESTIONE DEGLI ERRORI

- Il procedimento di error detecting e il conseguente **isolamenteo** dei guasti rendono Can molto **affidabile**. L'error detecting si articola in 5 fasi:
  - Il controller can di un nodo **rileva** un'errore (in TX o RX)
  - **Trasmette** immediatamente un Error Frame
  - Il messaggio corrotto è **ignorato** da tutti i nodi della rete
  - Il can controller **aggiorna** il suo stato
  - Il messaggio viene **ritrasmesso** (eventualmente dovrà competere, per l'accesso al bus, con altri messaggi.

GESTIONE DEGLI ERRORI

- Il protocollo CAN definisce **5 differenti tipi di errore**, di cui **3 a livello di bit e 2 a livello di messaggio**, rilevati attraverso le seguenti tecniche:
  - **monitoraggio**: ciascun nodo **confronta i bit che invia con quelli effettivamente presenti** sul bus e in caso di discordanza si ha un *Bit error*; fa eccezione a questo monitoraggio il campo identificatore di un D.F o di un R.F. : la presenza di un bit diverso da quello inviato (un bit 'd' anziché un bit 'r' e non viceversa) viene infatti interpretata come perdita dell'arbitraggio e il nodo si pone in modalità di ricezione.
  - **bit stuffing**: è una tecnica (già illustrata in precedenza) che consiste nel trasmettere dopo ogni sequenza di 5 bit uguali un sesto bit complementare, che viene automaticamente ignorato dai nodi ricevitori. Questa tecnica viene utilizzata nei messaggi di tipo D.F. e R.F. mentre un E.F. viola questa regola. Se un ricevitore rileva 6 bit consecutivi dello stesso tipo si uno *Stuff Error.*

GESTIONE DEGLI ERRORI

- ***Controllo ciclico di ridondanza*** *(già visto in precedenza): ciascun nodo ricevitore calcola la sequenza CRC corrispondente al messaggio ricevuto e la confronta con quella che il trasmettitore ha accodato al messaggio stesso. In caso di differenza fra le due sequenze si ha un CRC error.*

- **Controllo dei Frame**: se il ricevitore rileva che non **sono state rispettate le 5 possibili strutture dei frame CAN**, genera un *Form Error*.

- **Invio dell'acknowledgement bit**: ciascun nodo che riceve correttamente un D.F. o un R.F. è tenuto a **inviare un bit dominante nel bit-time del campo ACK di questo frame**. Se la sovrascrittura non avviene il bit ACK-Slot **resta recessivo** e il trasmettitore invia un messaggio di errore: si ha **Acknowledgement Error** soltanto se nessuno degli altri nodi invia un bit dominante.

ISOLAMENTO DEI GUASTI - Contatori

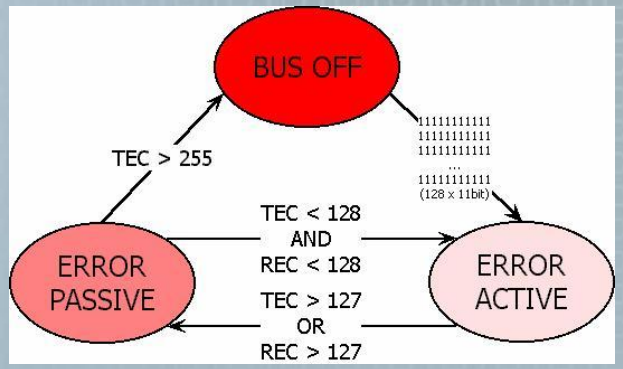
- A ciascuno dei 5 tipi di errore, indicati nelle diapositive precedenti, è assegnata una **determinata gravità** che si quantifica nell'incremento da 1 a 8 dei due contatori presenti in ciascun nodo CAN **(+1 per un errore in ricezione, +8 per un errore in trasmissione**):
  - Receive Error Counter (**REC**): tiene conto degli errori rilevati dal nodo quando è in modalità ricevitore.
  - Transmit Error Counter (**TEC**): tiene conto degli errori sui messaggi che il nodo trasmette, rilevati dal nodo stesso o segnalati dagli altri nodi con E.F.

ISOLAMENTO DEI GUASTI - Contatori

- In base al valore raggiunto da questi due contatori, ciascun nodo si pone in uno dei tre stati:
  - **Error Active, se entrambi i contatori sono inferiori a 128.** La frequenza e la gravità degli errori sono quindi trascurabili e il nodo può partecipare normalmente alle attività del bus, inviando anche Active Error Flag in caso di rilevazione degli errori.
  - **Error Passive, se almeno uno dei due contatori è maggiore di 128**. La frequenza e la gravità degli errori fanno ritenere il nodo poco affidabile, perciò la segnalazione degli errori può avvenire solo con un Passive Error Flag (che non provoca automaticamente la ritrasmissione) e per la trasmissione deve attendere un tempo superiore ai nodi attivi (l'interframe space è di 8 bit-time anziché di 3).
  - **Bus Off: se uno dei due contatori raggiunge il valore 256**. Il nodo è gravemente soggetto ad errori e non partecipa quindi alle attività del bus; resta però in ascolto per rilevare una sequenza di 11 bit recessivi consecutivi, che segnala la sua riammissione.

ISOLAMENTO DEI GUASTI - Riattivazione

- Un nodo in stato di **"bus off"** può ritornare "error active" (con entrambi i contatori a zero) Dopo che sono state rilevate sul canale 128 occorrenze di bit recessivi consecutivi.
- Questa **sequenza** può essere inviata da un **monitor** esterno, gestito da un operatore, per riabilitare un determinato nodo. (per esempio il monitor esterno collegato a una autovettura mediante presa **OBD**, può riabilitare una singola periferica esclusa dal Transfer Layer)



Sequenza di riabilitazione

- Con questo piccolo terminale **(application layer)** si può controllare lo stato delle periferiche sul bus, e riabilitare, se necessario, un nodo con la "sequenza di riabilitazione" vista sopra.

Conclusione

- Oggi ci troviamo già nella situazione in cui ogni operazione compiuta dal conducente, si tramuta in segnali elettrici elaborati da uno o più calcolatori, che provvederanno alla giusta attuazione del comando. Ci troviamo di fronte ad una grande **affidabilità e una grande velocità** (non grande in assoluto, ma grande in confronto alla **piccola mole di dati** che basta per gestire un bus come il CAN).

- Con il tempo si diffonderanno sempre di più i sistemi "**Drive-By-Wire**" e diminuiranno sempre di più i cablaggi nelle auto. Questo sviluppo verrà sempre alimentato dalla ricerca e sviluppo delle tecnologie per avionica.

- Se diamo un'occhiata al passato ci possiamo accorgere che il transistor è stato inventato non molto tempo fa. Questa, come moltissime altre tecnologie, sono solo agli albori. **Il loro sviluppo non è lineare, ma esponenziale,** dato che le nuove tecnologie scoperte sono la base per sviluppare le successive in modo migliore e più rapido.