Versione N° 2.1

LaurTec

Il protocollo CAN

Autore : *Mauro Laurenti*

ID: AN4004-IT

INFORMATIVA

Come prescritto dall'art. 1, comma 1, della legge 21 maggio 2004 n.128, l'autore avvisa di aver assolto, per la seguente opera dell'ingegno, a tutti gli obblighi della legge 22 Aprile del 1941 n. 633, sulla tutela del diritto d'autore.

AVVERTENZE

I progetti presentati non hanno la certificazione CE, quindi non possono essere utilizzati per scopi commerciali nella Comunità Economica Europea.

Chiunque decida di far uso delle nozioni riportate nella seguente opera o decida di realizzare i circuiti proposti, è tenuto pertanto a prestare la massima attenzione in osservanza alle normative in vigore sulla sicurezza.

L'autore declina ogni responsabilità per eventuali danni causati a persone, animali o cose derivante dall'utilizzo diretto o indiretto del materiale, dei dispositivi o del software presentati nella seguente opera.
Si fa inoltre presente che quanto riportato viene fornito così com'è, a solo scopo didattico e formativo, senza garanzia alcuna della sua correttezza.

L'autore ringrazia anticipatamente per la segnalazione di ogni errore.

Indice

Introduzione....4
Il protocollo CAN, uno sguardo d'insieme....4
Introduzione alle specifiche CAN 2.0, Parte A....6
Data Frame....8
Remote Frame....11
Error Frame....11
Overload Frame....12
Interframe Space....12
Introduzione alle specifiche CAN 2.0, Parte B....13
Gestione degli errori....15
Confinamento degli Errori....16
Sincronizzazione e temporizzazioni....17
Considerazioni sul Physical Layer....19
Le specifiche CAN FD....22
Bibliografia....26
History....27

Introduzione

Il protocollo CAN è ormai utilizzato in applicazioni che vanno ben oltre gli scopi originali per cui venne ideato. La sua affidabilità lo rendono un protocollo ideale in applicazioni dove non si può lasciare alcun margine all'errore di trasmissione. In questo Tutorial, dopo una breve introduzione sulle applicazioni del protocollo CAN si introducono con un certo dettaglio le specifiche CAN 2.0 e le modifiche introdotte nell'estensione CAN FD.
Al termine del Tutorial il lettore dovrebbe essere in grado di utilizzare una qualunque libreria CAN o comunque avere un buon background con cui affrontare il datasheet di un microcontrollore per poter scrivere una propria libreria CAN.

Il seguente Tutorial non rappresenta una sostituzione delle specifiche CAN ufficiali. Seppure faccia rifermento alle specifiche CAN il lettore è incoraggiato sempre a far riferimento alla documentazione ufficiale per maggior dettagli e risoluzione di problemi.

Il protocollo CAN, uno sguardo d'insieme

Il protocollo CAN (Controller Area Network)[1] venne ideato nel 1986 dalla Bosch su richiesta della Mercedes. Il protocollo CAN nacque dunque per scopi *automotive* al fine di ridurre il cablaggio tra le varie apparecchiature elettroniche che cominciavano ad affollare gli autoveicoli. La semplicità del cablaggio discende dal fatto che il protocollo descrive una comunicazione seriale. Oltre che a questioni meramente pratiche il bus doveva rispettare stringenti vincoli di sicurezza ed affidabilità in termini di errori di trasmissione. La natura di questa esigenza derivava dal fatto che apparecchiature legate alla sicurezza del guidatore come per esempio l'air bag, il sistema antislittamento, avrebbero fatto uso di tale bus.
L'affidabilità che venne raggiunta fu tale per cui molte altre case automobiliste cominciarono ad utilizzare il protocollo CAN come protocollo di trasmissione all'interno dei loro autoveicoli.
Oggigiorno si sta avendo ancora un'ulteriore spinta verso applicazioni che vanno ben oltre quelle per le quali il bus venne inizialmente ideato. Il protocollo CAN viene attualmente utilizzato in applicazioni robotiche, medicali, domotiche[2] ed industriali.

La sua espansione risiede nel fatto che il suo utilizzo non risulta particolarmente complesso dal momento che, come si vedrà, gran parte della sua complessità è gestita in maniera intelligente a livello hardware. Molti microcontrollori odierni possiedono al loro interno un CAN controller senza che questo faccia lievitare i costi degli stessi.
Il protocollo CAN ha subito dalla sua ideazione alcune modifiche. Attualmente la versione di cui si fa utilizzo è la versione 2.0, la cui documentazione ufficiale venne rilasciata nel 1991[3]. La versione 2.0 è suddivisa in Parte A e Parte B (nominate anche 2.0A e 2.0B). Nei paragrafi successivi verrà maggiormente spiegata la Parte A mentre la trattazione della Parte B consisterà principalmente nell'evidenziare le differenze con la

[1] Per una migliore comprensione di quanto verrà esposto si raccomanda la lettura dei Tutorial "Bus I2C" e "Il Protocollo RS232" disponibili sul sito LaurTec.
[2] In generale per applicazioni domotiche si fa riferimento ad una rete fisica o meno, che connette vari sistemi domestici rendendo la casa più "intelligente" .
[3] Si faccia riferimento alla Bibliografia per maggiori informazioni.

Parte A. Tra le caratteristiche principali del protocollo CAN, di cui si parlerà a breve, si mettono in evidenza:

- **Semplicità di cablaggio**
  Su di un Bus CAN è possibile aggiungere o rimuovere periferiche (nodi) con grande semplicità. Questo discende dal fatto che la comunicazione avviene su un doppino elettrico e dal fatto che le periferiche non sono identificate da indirizzi[4]. Il protocollo CAN è infatti orientato ai messaggi (message oriented) e non orientato agli indirizzi (address oriented).

- **Tempi di risposta rigidi**
  Il Bus CAN permette di ottenere sistemi i cui tempi di risposta sono molto rigidi grazie a mirati artifici per evitare perdite di tempo.

- **Alta immunità ai disturbi**
  Lo standard ISO11898 prescrive che il protocollo CAN debba poter continuare a funzionare anche nel caso in cui si abbia l'interruzione di uno dei due fili o un corto circuito di una linea del bus verso l'alimentazione.

- **Elevata affidabilità**
  L'Hardware si accorge automaticamente di eventuali errori di trasmissione e provvede alla ritrasmissione dell'informazione senza intervento del software.

- **Confinamento degli Errori**
  Ogni periferica connessa al Bus CAN è in grado di autodiagnosticare eventuali problemi hardware ed autoescludersi dal bus in caso di malfunzionamento, permettendo alle altre periferiche l'utilizzo del bus. In particolare la periferica riesce a distinguere il caso in cui l'errore è occasionale o sistematico. Nel caso in cui l'errore sia sistematico la periferica (nodo) è in grado di autoescludersi dal bus.

- **Priorità dei messaggi**
  I messaggi che vengono trasmessi da ogni nodo possiedono una priorità intrinseca legata alla struttura del messaggio stesso. La parte del messaggio a cui è legata la priorità è nominata identificatore (identifier). Per implementare la regola della priorità è definito anche un bit "dominante" e un bit "recessivo"[5]. Se un bit dominante è trasmesso in contemporanea ad un bit recessivo sul bus prevarrà il bit dominante.

- **Protocollo multimaster**
  Nel protocollo CAN ogni nodo può concorrere per l'utilizzo del bus. Ogni nodo può infatti prendere dominio del bus e trasmettere come master. In caso in cui più nodi dovessero tentare di trasmettere un messaggio la priorità del messaggio farà prevalere il nodo con identificatore a più alta priorità.

[4] Quando in un bus le periferiche sono identificate da indirizzi vuol dire che per comunicare con una determinata periferica se ne deve conoscere l'indirizzo. Aggiungere o togliere periferiche può compromettere l'integrità del sistema.

[5] In una rete wired AND il bit dominante è rappresentato dallo 0.

Introduzione alle specifiche CAN 2.0, Parte A

Come detto in precedenza sono presenti due specifiche per il protocollo CAN. La versione A e la versione B possono tra loro comunicare qualora non si faccia uso del formato esteso che verrà spiegato successivamente. Dunque tutto quello che si dirà per la versione A può considerarsi valido anche per la versione B. Il protocollo CAN è suddiviso in 3 differenti livelli (layers)

- CAN object layer
- CAN transfer layer
- CAN physical layer

L'object layer è il livello più vicino al software che viene scritto per la gestione del bus. Lo scopo di questo livello è quello di filtrare i messaggi che vengono ricevuti e preparare quelli che devono essere trasmessi. A questo livello fanno parte tutte le maschere che il programmatore deve impostare per filtrare i messaggi.

Il transfer layer racchiude le regole vere e proprie del protocollo CAN in particolare controlla la formattazione dei messaggi (frame), l'arbitraggio del bus, controllo degli errori, segnalazione degli errori e il confinamento della periferica in caso di errori.

Il livello physical layer rappresenta il mezzo fisico per mezzo del quale avviene la comunicazione. Questo livello non viene descritto dalle specifiche CAN, le quali lasciano libertà al progettista di adeguare il mezzo fisico del bus a seconda delle loro esigenze. Normalmente come mezzo fisico si utilizza un doppino elettrico schermato o meno, o una fibra ottica. Le specifiche elettriche del bus vengono a dipendere dal particolare mezzo fisico di cui si fa uso. Alcuni informazioni sul mezzo fisico sono riportate nel paragrafo “Considerazioni sul physical layer”. In Figura 1 è riportata una rappresentazioni a blocchi di quanto appena descritto. In aggiunta a quanto detto è stato aggiunto in cima al tutto, il livello Application Layer. Questo livello, in generale, è rappresentato dal Firmware che gestisce il microcontrollore.

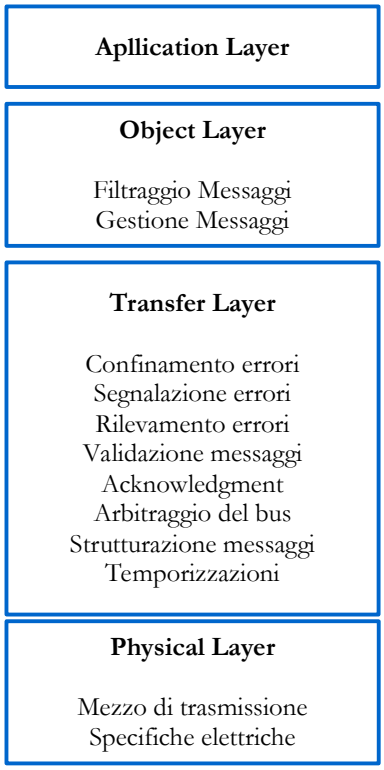

Figura 1: *Rappresentazione a blocchi dei livelli del protocollo CAN.*

I messaggi che vengono trasmessi per mezzo del protocollo CAN sono strutturati in maniera univoca. Ogni tipo di messaggio viene nominato Frame. In particolare sono presenti quattro tipi di strutture (Frame):

- **Data Frame**
  Il Data Frame è la struttura dati per mezzo della quale i nodi trasmettono l'informazione. Questa struttura è caratterizzata da un identificatore (identifier) per mezzo del quale è possibile identificare la tipologia del messaggio. Come detto il protocollo CAN non è orientato agli indirizzi. Infatti ogni nodo che vuole trasmettere un'informazione trasmette un Data Frame con un determinato identificatore, sul bus e non un indirizzo al quale far recapitare il messaggio. Tutti i nodi sul bus ricevono il messaggio e a seconda del tipo di identificatore decidono se elaborare o meno l'informazione ricevuta. Questa decisione avviene per mezzo di filtri e maschere senza l'intervento del software utente.

- **Remote Frame**
  Il remote Frame, come si vedrà a breve, è molto simile al Data Frame ma non contiene nessuna informazione utile; il Remote Frame contiene infatti l'identificatore ma non i dati. Il suo scopo, se trasmesso da un nodo, è quello di avvertire gli altri nodi dicendo che vuole le informazioni identificate dall'identificatore trasmesso con il Remote Frame. Il Remote Frame ha una priorità più bassa del Data Frame.

- **Error Frame**
  L'Error Frame viene inviato da tutti i nodi ogni qual volta venga rilevato un errore

di trasmissione. I possibili errori di trasmissione verranno illustrati a breve.

- **Overload Frame**
  L'Overload Frame viene inviato da tutti quei nodi che richiedono tempo per elaborare l'informazione di un messaggio ricevuto. Questo frame viene dunque inviato come sinonimo di "ho ricevuto un messaggio, lo sto elaborando, aspetta". Solo due Overload Frame consecutivi posso essere inviati da un nodo. Dunque ad un nodo è concesso dire: aspetta, aspetta ma non di più. Questo limite permette di garantire che informazioni importanti non siano trasmesse a causa di attese prolungate.

Vediamo ora in maggior dettaglio come sono strutturati i vari Frame del protocollo CAN.

Data Frame

Come detto il Data Frame rappresenta la struttura dati per mezzo della quale i vari nodi inviano i dati nel bus. Il Data Frame è composto da 7 campi:

- Start of Frame
- Arbitration Field
- Control Field
- Data Field
- CRC Field
- ACK Field
- End of Frame

In Figura 2 è riportata una rappresentazione grafica della struttura del Data Frame.

Start of Frame	Arbitration Field	Control Field	Data Field	CRC Field	ACK Field	End of Frame

Figura 2: *Struttura del Data Frame.*

Vediamo in dettaglio i vari campi:

Start of Frame

Questo frame consiste di un unico bit dominante, utilizzato per segnalare l'inizio del Data Frame. Il Data Frame è sempre preceduto da un Interframe che segnala ai nodi che il bus è libero. Lo Start of Frame viene inviato da ogni nodo che vuole inviare dei dati. Questo può avvenire solo se il bus è libero (bus idle). Lo Start of Frame permette a tutti i nodi in ricezione di sincronizzarsi con il nodo trasmittente[6].

Arbitration Field

L'Arbitration Field consiste nell'Identifier (identificatore) e il bit RTR (Remote Transmission Request). Questo campo è detto d'arbitraggio poiché l'identificatore del messaggio racchiude l'importanza del messaggio stesso. L'Identifier consiste di 11 bit[7] che vengono trasmessi dal più significativo al meno significativo. I 7 bit più significativi non

[6] Lo Start of Frame è paragonabile al bit di start utilizzato nel protocollo RS232.
[7] Nella modalità estesa che verrà spiegata successivamente si fa uso di 29 bit.

devono essere tutti contemporaneamente recessive. Il bit RTR è settato come dominante. Quando il bit RTR è settato come recessivo vuol dire che si ha un Remote Frame e non un Data Frame. Cerchiamo di capire meglio perché questo campo è detto d'arbitraggio.
Ogni nodo che trasmette sul bus controlla anche il valore presente sul bus, ovvero se trasmette un bit dominante si aspetterà che il bus abbia un bit dominante e viceversa in caso di bit recessive. Supponiamo che due nodi abbiano iniziato a trasmettere in contemporanea. I nodi iniziano con un start of frame ovvero trasmetteranno un bit dominante e controllando il bus leggeranno un bit dominante. Questo farà credere ai due nodi di essere i soli a trasmettere. Supponiamo che l'identificatore del messaggio del primo nodo sia:

d d r r r r d r r r r

mentre l'identificatore del secondo nodo sia

d r d r r r d r r r r

I due nodi dopo aver trasmesso lo Start of Frame inizieranno a trasmettere l'identificatore. Tutti e due i nodi hanno all'inizio un bit dominante, dunque leggendo il bus i due nodi trovano un bit dominante e crederanno di essere i soli a trasmettere. Quando però il secondo bit viene trasmesso, il primo nodo trasmette un bit dominante mentre il secondo trasmette un bit recessive. A questo punto il primo nodo leggerà sul bus un bit dominante e penserà di essere il solo a trasmettere sul bus, mentre il secondo nodo leggendo un bit dominante invece di un bit recessive capirà che c'è un altro nodo che sta trasmettendo sul bus. In particolare dal momento che l'altro nodo ha un bit dominante invece di recessive, vuol dire che il messaggio che deve trasmettere è più importante. Il secondo nodo capendo che un messaggio più importante deve essere trasmesso abbandona la trasmissione (perde l'arbitraggio) e tenterà di ritrasmettere una volta che il messaggio più importante è terminato[8].

Una cosa molto importante da notare è che l'utilizzo dei bit dominant e recessive permette ad un nodo di prevalere su di un altro senza che venga perso tempo prezioso, infatti il primo nodo non si è accorto che era presente anche un secondo nodo che cercava di trasmettere un messaggio. L'arbitraggio permette inoltre di dare la priorità ai messaggi; in questo esempio si potrebbe pensare che il primo nodo sta cercando di trasmettere informazioni al freno mentre il secondo cerca di trasmettere informazioni al tergicristallo!

Control Field

Il campo di controllo consiste di 6 bit che contengono rispettivamente l'informazione della lunghezza dei dati (Data Length Code), misurata in numeri di byte, e due bit riservati (trasmessi sempre come dominati) concepiti per possibili espansioni del protocollo[9]. In Figura 1 è riportata la struttura del Control Field.

[8] La ritrasmissione non è in realtà garantita al termine del messaggio più importante. Il nodo che ha perso l'arbitraggio dovrà infatti competere con gli altri nodi per poter ritrasmettere il messaggio. Questo processo è gestito automaticamente dal nodo senza intervento del software.

[9] Si tengano a mente questi due bit poiché verranno utilizzati proprio nelle specifiche 2.0B (un'espansione appunto).

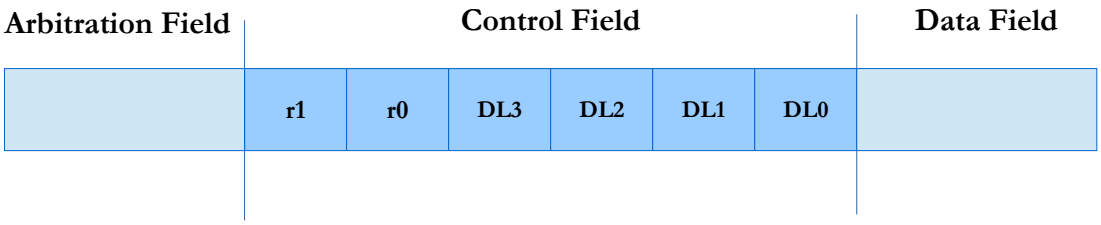

Figura 3: *Struttura del Control Field.*

Come visibile i primi due bit r1 ed r0 sono quelli riservati, mentre i restanti quattro definiscono la lunghezza dei dati. La lunghezza massima dei dati è di 8 byte, anche se potenzialmente in quattro bit si potrebbero scrivere numeri più grandi. Nel caso in cui si dovesse scrivere un numero più grande di 8 questo viene considerato sempre come 8. In Tabella 1 è riportata la sequenza di numeri validi per il Data Length Code.

Number	DL3	DL2	DL1	DL0
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

Tabella 1: *Numerazione per il Data Length Code.*

Data Field

Il Data Field rappresenta il campo dove vengono trasmessi i byte che rappresentano i dati da trasmettere. Il numero di byte è legato a Control Field precedentemente spiegato. La sua lunghezza è variabile e dipende dal numero di byte da trasmettere (max. 8 byte).

CRC Field

Il campo CRC contiene un codice particolare che viene calcolato per ogni Data Frame. Il suo valore è contenuto in 15 bit. Il calcolo tiene conto dei dati contenuti nei campi precedentemente spiegati. Il codice CRC permette la rilevazione degli errori in sede di ricezione. Infatti ogni nodo ricevente, in base ai dati nel Data Frame, ricostruisce il CRC. Nel caso in cui non siano presenti errori in ricezione il CRC trasmesso dal nodo trasmittente e il CRC calcolato dal nodo ricevente, devono essere uguali. Il codice CRC è seguito da un bit recessive denominato CRC Delimiter. Si fa presente che il codice CRC è calcolato dal modulo CAN e non via software.

ACK Field

L'Acknowledgment Field consiste di due bit nominati ACK Slot e ACK Delimiter. Questi due bit sono trasmessi recessive. Durante la trasmissione del bit ACK Slot, tutti i nodi che hanno ricevuto correttamente il messaggio trasmesso, trasmettono un bit dominante. Si capisce che il trasmettitore avendo trasmesso un bit recessive può

comunque trovare sul bus un bit dominant. Se questo non dovesse avvenire viene segnalato un errore (gli errori verranno descritti a breve) poiché nessun nodo ha apparentemente ricevuto il messaggio.

End of Frame

Al termine dei campi precedentemente descritti viene inviata la segnalazione del temine Frame. L'End of Frame consiste di 7 bit recessive

Remote Frame

Il Remote Frame è strutturato in maniera quasi identica al Data Frame. La differenza tra i due è legata al fatto che il Remote Frame non possiede il campo dati, indipendentemente dal valore che assume il campo Data Length del Control Field. Il Remote Frame si distingue dal Data Frame dal valore del RTR bit. Nel Data Frame, come detto, RTR assume il valore del bit dominante mentre nel Remote Frame assume il valore del bit recessivo.
Il Remote Frame viene inviato da qualunque periferica interessata al messaggio identificato dall'identificatore presente nella struttura del Remote Frame. Il Remote Frame possedendo il bit RTR recessive ha meno priorità del Data Frame. Se ad esempio un nodo cerca d'inviare un messaggio con identificatore

d r d r r r d r r r r

mentre un secondo nodo cerca di richiedere la trasmissione dei dati identificati dall'identificatore

d r d r r r d r r r r

cioè lo stesso identificatore del nodo che sta trasmettendo l'informazione. Si capisce che dal momento che il secondo nodo sta chiedendo un messaggio che sta proprio per essere trasmesso, perderà l'arbitraggio. Questo permette di evitare di perdere tempo superfluo.

Error Frame

L'Error Frame viene trasmesso da tutte le periferiche ogni qual volta venga rilevato un errore in ricezione o in trasmissione. Sono presenti due tipi di Error Frame:

- Active Error Frame
- Passive Error Frame

I due tipi di Error Frame vengono inviati a seconda dello stato del nodo (più dettagli si vedranno in seguito). Se il nodo funziona correttamente invia un Active Error Frame mentre se il nodo sta funzionando con alcuni problemi invia un Passive Error Frame.
Un Active Error Frame consiste nella trasmissione di 6 bit dominanti consecutivi mentre un Passive Error Frame consiste nella trasmissione di 6 bit recessive. I sei bit dominati o meno, appartengono al campo nominato Error Flag. In ambe due i casi la trasmissione dei sei bit avviene senza bit stuffing[10]. Questo permette a tutti gli altri nodi di rilevare un

[10] Il bit stuffing consiste nell'inserire un bit di valore opposto, qualora siano presenti 5 bit consecutivi uguali. Quindi se si dovessero trasmettere 111111 la regola del bit stuffing farebbe inviare 1111101. Ogni nodo ricevente, conoscendo la regola dello stuffing effettuerà un destuffing per riottenere 111111. L'invio dei sei bit dell'Error Frame viola questa regola. Solo lo

errore e trasmettere a loro volta un Error Frame. Al termine dell'Error Frame è presente il campo Error Delimiter che consiste nell'invio di 8 bit recessive.

Overload Frame

L'Overload Frame consiste di due campi, l'Overload Flag e l'Overload Delimiter. L'Overload Flag consiste di 6 bit dominati mentre L'Overload Delimiter consiste di 8 bit recessive[11]. L'Overload Delimiter viene inviato da ogni nodo, che ricevuto un dato, richiede un certo tempo per poter elaborare l'informazione ricevuta ed essere nuovamente capace di ricevere altri dati. La sua funzione è dunque quella di porre un ritardo sul bus dicendo agli altri nodi di aspettare prima di trasmettere altre informazioni. Per ogni messaggio ogni nodo può inviare due soli Overload Frame.

Interframe Space

L'Interframe Space è un tipo particolare di frame che non è un frame! L'Interframe Space precede ogni Data Frame e Remote Frame. Al suo interno è presente il campo Intermission (3 bit recessive), durante il quale un nodo può trasmettere un Overload Frame[12] e il campo Bus Idle. Il campo Idle ha lunghezza indeterminata e indica che il bus è libero. Un Data Frame e un Remote Frame possono essere iniziati solo se il bus è in stato Idle.

Start of Frame, Arbitration Field, Control Field, il codice CRC sono soggetti alla regola dello stuffing. I rimanenti campi del Data Frame e del Remote Frame non sono soggetti allo stuffing, come anche l'Error Frame e l'Overload Frame. La violazione dello stuffing genera un errore di trasmissione/ricezione.

[11] Per ulteriori informazioni sull'Overload Frame far riferimento alla documentazione ufficiale delle specifiche CAN.

[12] Durante la fase in cui viene trasmesso il campo Intermission né il Data Frame né il Remote Frame possono essere trasmessi.

Introduzione alle specifiche CAN 2.0, Parte B

Le specifiche descritte per il protocollo CAN 2.0B fanno si che due nodi che rispettino rispettivamente le specifiche 2.0A e 2.0B possano comunicare tra loro qualora non si faccia utilizzo della modalità estesa ovvero dell'Extended Frame.

Cominciamo col dire che con le specifiche 2.0B si sono riorganizzati i layer precedentemente descritti in Figura 1, la riorganizzazione è solo formale, per ulteriori chiarimenti, si rimanda alla documentazione ufficiale della Bosch. La vera differenza tra le specifiche 2.0A e 2.0B consiste nell'Extended Frame. Il nuovo frame introdotto nella specifica 2.0B permette di utilizzare indirizzi a 29 bit invece degli 11 bit descritti nella specifiche CAN 2.0A. L'utilizzo di 29 bit permette una maggior flessibilità nella scelta dell'identificatore permettendo di raggruppare famiglie di dati e facilitare espansioni future. Un nodo CAN che soddisfi le specifiche 2.0B non necessariamente deve supportare l'Extended Frame ma deve essere tale che se altri nodi nel bus fanno uso dell'Extended Frame il nodo che non lo supporta non deve generare errori in ricezione.

Da quanto fin ora detto si capisce che, poiché la differenza tra le specifiche CAN 2.0A e 2.0B consiste prevalentemente nel numero di bit dell'identificatore, le modifiche si riflettono sul Data Frame e il Remote Frame. Il resto della struttura dei Frame precedentemente descritti rimane praticamente invariato. La struttura del Data Frame in modalità extended è riportata in Figura 4 b) (per chiarezza, sono esclusi i bit CRC, ACK e End of Frame che seguono il Data Field).

SOF	11 bits Identifier	RTR	r1	r0	Data length	Data Field

a) Standard Data Frame/Remote Frame

SOF	11 bits Identifier	SRR	IDE	18 bits Identifier	RTR	r1	r0	Data length	Data Field

b) Extended Data Frame/Remote Frame

Figura 4: *Struttura della modalità Standard (a) ed Extended (b) a confronto.*

Dal confronto di Figura 4 a) e b) è possibile osservare che lo Start of Frame rimane invariato ed è seguito dagli 11 bit che precedentemente rappresentavano l'identificatore. Nella modalità extended questi bit rappresentano ancora l'identificatore ma solo una parte di quest'ultimo. In particolare gli 11 bit rappresentano i bit tra ID28 e ID18, trasmessi rispettivamente da ID28 a ID18 ovvero a partire dal bit più significativo. I restanti bit dell'identificatore vengono trasmessi nella seconda parte del Data Frame. In particolare il bit SRR (Substitute Remote Request) viene posto ad un valore recessive. Questo bit si trova nella stessa posizione del bit RTR del data frame descritto nelle specifiche 2.0A. Dal momento che il bit è recessive, qualora un altro nodo dovesse trasmettere con un identificatore a soli 11 bit proprio uguali a ID28-ID18, il nodo che cerca di trasmettere il Data Frame in modalità standard (come descritto per le specifiche 2.0A[13]) prevarrà sul

[13] Si ricorda che un nodo compatibile con le specifiche 2.0B può ancora trasmettere secondo le specifiche descritte per la

Data Frame in modalità Extended. Dopo il bit SRR viene trasmesso il bit IDE bit (Identifier Extension bit). Questo bit si trova nella stessa posizione prima ricoperta dal bit r1 (bit riservato per possibili estensioni). Normalmente il bit r1 era posto a valore dominante mentre nella modalità estesa viene posto a valore recessive.

Con la lettura del bit IDE tutti i nodi sono consapevoli che il Data Frame deve essere interpretato come Extended, dunque gli 11 bit precedenti rappresentano i bit ID28-ID18. La ragione per cui il bit SRR non può essere sufficiente ad indicare un Extended Data Frame è legato al fatto che un valore recessive per il bit SRR (ovvero il vecchio RTR) nella modalità standard viene utilizzato per segnalare il Remote Frame. Solo quando viene trasmesso il bit riservato r1 (supposto dominant nella modalità standard) e viene trovato al valore recessive, allora i nodi comprendo che si ha a che fare con un Extended Data Frame o Extended Remote Frame.

Dopo il bit IDE vengono trasmessi i restanti 18 bit dell'identificatore a 29 bit. Al termine dell'identificatore è presente il nuovo RTS bit, i nuovi r1 e r0 bit (riservati per possibili nuove estensioni) e il resto della struttura del Data Frame o Remote Frame. Il valore del nuovo bit RTR permette di impostare il frame trasmesso come un Extended Data Frame o un Extended Remote Frame, in particolare RTR è dominante per il Data Frame e recessive per il Remote Frame. Il resto della struttura è praticamente identica a quanto già descritto per le specifiche 2.0A.

versione 2.0A ma non deve generare errori qualora si ritrovi a ricevere Frame in modalità Extended.

Gestione degli errori

Come detto il protocollo CAN è stato progettato per ridurre al minimo la possibilità di errore. Questo non significa che trasmettendo un dato questo non possa essere trasmesso o ricevuto privo di errori. Bassa probabilità di errore deve essere vista nel senso che se si dovesse verificare un errore di trasmissione o ricezione, la probabilità che questo passi inosservato è molto bassa. Ogni nodo trasmettitore, in caso di errore ritrasmette automaticamente il dato che è stato compromesso da un errore. Il protocollo CAN prevede il rilevamento di 5 tipologie di errori, i quali possono avvenire anche in contemporanea ovvero non sono mutuamente esclusivi. La rilevazione degli errori viene segnalata dai vari nodi per mezzo della trasmissione dell'Error Frame. Le tipologie di errori sono:

- **Bit Error**
  L'errore di bit viene rilevato dal nodo trasmettitore al tempo di bit, ovvero quando il trasmettitore trasmette un bit sul bus. Il trasmettitore durante la trasmissione di un bit legge anche il valore del bus per vedere se il bus assume lo stesso valore del bit trasmesso. Se il bit trasmesso e quello letto sono differenti si ha un errore di bit, fanno però eccezione alcuni casi. Come si è già detto, durante la trasmissione dell'identificatore e dell'RTR bit è possibile che un nodo perda l'arbitraggio. La perdita di arbitraggio consiste proprio nella situazione in cui si trasmette un bit recessivo e si legge un bit dominante. In questa situazione non si verifica un bit Error ma solo la perdita dell'arbitraggio. La stessa cosa vale durante la trasmissione del ACK Slot. Altra eccezione si ha nella trasmissione di un Passive Error Flag. Infatti un nodo che trasmette tale messaggio deve dare priorità ad un Active Error Flag.

- **Stuff Error**
  Il protocollo CAN prevede la trasmissione dei bit per mezzo della codifica del bit stuffing. Questa consiste nell'inserire un bit di polarità inversa ogni qualvolta si dovessero avere 5 bit di ugual valore. Questa tecnica discende dal fatto che il clock di trasmissione non è trasmesso con i dati e i nodi devono potersi risincronizzare sulle transizioni dei bit. La regola dello stuffing viene violata se sono rilevati più di 5 bit di ugual valore. Come detto questa regola viene volontariamente violata durante la trasmissione di un Error Frame.

- **CRC Error**
  Tutti i campi del Data Frame o Remote Frame che precedono il campo CRC sono codificati attraverso un codice ridondante nominato CRC. Il codice CRC viene inviato dal nodo trasmettitore e ricostruito indipendentemente dal nodo ricevente. Il nodo ricevente confronta il CRC ricevuto con il CRC ricostruito per verificare se sono avvenuti degli errori durante la trasmissione/ricezione delle informazioni.

- **Form Error**
  Come visto ogni campo è strutturato in maniera fissa con un determinato numero di bit. Qualora la struttura di un Frame venga violata, si ha un Form Error.

- **Acknowledgment Error**
  Questo errore viene rilevato dal nodo trasmittente qualora durante la trasmissione del bit recessive ACK Slot, nessun nodo risponde con un bit dominante. Questo significa che nessun nodo ha ricevuto correttamente il dato trasmesso.

Confinamento degli Errori

La rilevazione degli errori e la trasmissione tra i nodi del rilevamento degli errori rilevati permette ai vari nodi di comprendere il loro stato di benessere. Un nodo che si ritiene "malato" e potenziale causa di errori sul bus si autoesclude dal bus stesso, permettendo agli altri nodi di trasmettere e ricevere con minor problemi. L'autodiagnosi del nodo avviene controllando il valore di due contatori particolari che vengono incrementati o decrementati a seconda del tipo di errore rilevato. L'incremento e il decremento può avvenire con valori maggiori di 1 a seconda della gravità dell'errore rilevato. I contatori utilizzati per l'autodiagnosi sono:

- Transmit Error Count
- Receive Error Count

I contatori vengono cambiati in accordo alle seguenti regole[14]:

- Ogni volta che il nodo ricevitore rileva un errore il Receive Error Count viene incrementato di 1.
- Quando il nodo trasmettitore invia un Error Flag il Transmit Error Count viene incrementato di 8.
- Quando il trasmettitore rileva un bit Error mentre trasmette un Active Error Flag o un Overload Flag, il Transmit Error Count viene incrementato di 8.
- Quando il ricevitore rileva un bit Error mentre trasmette un Active Error Flag o un Overload Flag, il Receive Error Count viene incrementato di 8.
- Dopo una trasmissione avvenuta con successo il Transmit Error Count viene decrementato di 1 (qualora non sia 0).

Come visibile a seconda della gravità dell'errore i contatori vengono incrementati di 1 o 8. Quando ambedue i contatori di un nodo sono inferiori a 127[15] il nodo è nello stato Error Active (funzionale al 100%). Se uno dei due contatori supera 127 il nodo entra nello stato Error Passive (partecipa alla comunicazione ma può inviare solo il Passive Error Flag). Un nodo entra nella stato Bus OFF qualora uno dei due contatori superi 255. Bus OFF significa che il nodo si ritiene "malato" e si disattiva dal Bus. Il nodo nello stato Bus Off può ancora ricevere informazioni dal Bus ma non partecipa né alla trasmissione d'informazione né alla trasmissioni di errori.
Il nodo che si trova nello stato Error Passive può tornare nello stato Error Active in

[14] Non tutte le regole sono riportate, ed in particolare le eccezioni agli errori non sono descritte. Per maggiori chiarimenti si faccia riferimento alla documentazione ufficiale della Bosch.

[15] Nonostante un nodo che abbia i contatori con valore inferiore a 128 sia ritenuto nello stato Error Active e funzionale al 100%, un valore dei contatori maggiore o uguale a 96 è indice di un bus particolarmente disturbato.

maniera automatica qualora i contatori diventino inferiori a 127. Un nodo che si trovi nello stato Bus OFF può tornare nello stato Error Active solo se riceve la sequenza di sblocco composta da 128 occorrenze di 11 bit recessive consecutivi. Questa sequenza viene in generale trasmessa da un operatore esterno per mezzo di strumenti di diagnosi ad hoc per il Bus CAN. Per mezzo di questi strumenti è possibile studiare il Bus CAN e verificare lo stato delle periferiche.

Molti microcontrollori possiedono degli interrupt associati ai vari cambi di stato del controller CAN. In particolare sono settati dei flag ed eventualmente abilitati degli interrupt, ogni qualvolta il sistema entri in stato Error Active, Error Passive o Bus OFF. In questo modo si evita di dover controllare continuamente i valori dei contatori.

Sincronizzazione e temporizzazioni

Il protocollo CAN 2.0 supporta velocità di trasmissioni fino a 1Mbit/s, questo permette di gestire in maniera real time molte tipologie di periferiche in cui non sia richiesto uno streaming dati eccessivo. Dal momento che il clock non è trasmesso assieme alla linea dati si capisce che ogni nodo deve ricostruire il clock internamente[16]. Questo non significa che il nodo ricevente deve indovinare la velocità del bus[17] ma che deve sincronizzare il proprio clock interno con il clock del nodo trasmittente. La necessità di sincronizzare la fase del clock interno con la fase del clock trasmittente discende dal fatto che se così non si facesse la lettura dei bit potrebbe essere falsata.

La frequenza con cui trasmettono i vari nodi può essere anche inferiore a 1Mbit/s ma in ogni modo è fissa ed in generale nota a priori. La velocità di trasmissione di cui fa uso il protocollo CAN richiede che il clock debba essere generato per mezzo di un quarzo in modo da mantenere una certa stabilità della frequenza nel tempo. Ciononostante per frequenze inferiori a 125Kbits/s è possibile utilizzare oscillatori ceramici. La sincronizzazione del clock interno dei nodi riceventi avviene come per il protocollo RS232 nel momento in cui viene rilevato lo Start of Frame. Questo tipo di sincronizzazione viene detta Hard Synchronization. La sincronizzazione del clock viene mantenuta nel tempo anche grazie agli stuff bit e alle transizioni tra bit[18]. Ogni bit trasmesso può essere pensato come composizione di vari segmenti temporali. In particolare si definisce:

$$Nominal\ Bit\ Time = 1 \div Nominal\ Bit\ Rate$$

Il Nominal Bit Time, ovvero la durata del bit è cosi suddiviso:

- **Synchronization Segment** (SYNC_SEG)
  Il SYNC_SEG è sfruttato dai vari nodi per effettuare la sincronizzazione della fase del clock interno.

[16] In particolare il CAN utilizza la trasmissione NRZ per mezzo della quale non è possibile estrarre l'informazione del clock.

[17] Sono in realtà presenti semplici tecniche per far indovinare al nodo la velocità con cui vengono trasmessi i dati nel bus. Una volta trovata la frequenza il nodo mantiene l'informazione e successivamente deve solo sincronizzare la fase del clock interno con il nodo trasmittente.

[18] La sincronizzazione deve rispettare alcune regole particolari, per ulteriori informazioni si faccia riferimento alla documentazione ufficiale della Bosch.

- **Propagation Time Segment** (PROP_SEG)
  Il PROP_SEG è sfruttato per compensare eventuali ritardi di propagazione sulla linea di trasmissione e dei bus driver.

- **Phase Buffer Segment 1** (PHASE_SEG1)
  Il PHASE_SEG1 come anche PHASE_SEG2 possono essere allungati o accorciati durante la sincronizzazione del clock interno del nodo ricevente. Al termine del PHASE_SEG1 avviene il campionamento del bit ricevuto.

- **Phase Buffer Segment 2** (PHASE_SEG2)
  Si veda PHASE_SEG1.

Una rappresentazione grafica della struttura temporale di un bit è riportata in Figura 5.

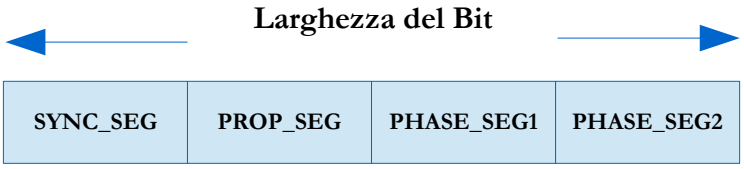

Figura 5: *Struttura temporale del singolo bit.*

Un'altra grandezza fondamentale definita nel protocollo CAN è il concetto di Quanto Temporale. Il quanto rappresenta l'unità temporale elementare. Il quanto viene derivato direttamente dall'oscillatore di sistema[19]. Ogni bit deve essere composto da un numero di quanti che varia da 8 a 25. La ripartizione dei quanti nella struttura temporale del bit precedentemente descritta è:

- Synchronization Segment (SYNC_SEG) : 1 quanto
- Propagation Time Segment (PROP_SEG) : 1-8 quanti
- Phase Buffer Segment 1 (PHASE_SEG1) : 1-8 quanti
- Phase Buffer Segment 2 (PHASE_SEG2) : è il massimo tra PHASE_SEG1 e IPT (Information Processing Time)
- Information Processing Time (IPT): massimo 2 quanti

L'Information Processing Time è il tempo che intercorre dal punto del campionamento e il rilevamento del valore del bit.

Durante la fase di sincronizzazione la durata del PHASE_SEG1 e del PHASE_SEG2 può essere allungata o accorciata a seconda dell'esigenza. Il valore massimo per cui è possibile accorciare o allungare tali intervalli temporali è associato al valore di Resynchronization

[19] Nel caso di un microcontrollore il Quanto viene derivato dal quarzo interno o esterno al microcontrollore facendo uso di prescaler.

Jump Bit che può assumere un valore compreso tra 1 e il minimo tra 4 e PHASE_SEG1.

La sincronizzazione del clock avviene alla transizione di ogni bit da recessivo a dominante.

Considerazioni sul Physical Layer

Quanto finora spiegato rappresenta il protocollo CAN 2.0. Come detto il physical layer non è descritto dalle specifiche Bosch, in maniera da lasciare libertà al progettista ed adeguare il mezzo trasmissivo alle esigenze del progetto. Sebbene sia presente una certa libertà, sono stati introdotti diversi standard ISO (International Standards Organization) al fine di garantire una facile interoperabilità tra moduli in applicazioni più sensibili. In particolare in ambito automobilistico, oltre agli standard ISO sono stati introdotti anche degli standard da parte della Society of Automotive Engineers (SAE). Tra i principali standard (forniti a pagamento dai relativi organi di competenza) si ricordano:

- ISO11898-1-2003: Standard ISO che descrive le specifiche CAN, escluso il physical layer.
- ISO11898-1:2015: Specifiche ISO11898 aggiornate con le specifiche CAN FD.
- ISO11898-2-2003: Standard ISO che descrive il physical layer ad alta velocità fino a 1Mb/s. Lo standard, descrivendo il physical layer, non appartiene alle specifiche CAN.
- ISO11898-3-2006: Standard ISO che descrive il physical layer a bassa velocità, compresa tra 40-125Kb/s. Lo standard, descrivendo il physical layer, non appartiene alle specifiche CAN.
- J1939 (elaborato dalla SAE) il cui scopo è quello di standardizzare le reti CAN sugli autobus e veicoli pesanti.

Alcuni degli standard estendono le specifiche della Bosch, definendo le specifiche elettriche che deve soddisfare il Bus CAN e le relative condizioni operative a cui possono essere sottoposti i relativi transceiver. Normalmente il mezzo fisico utilizzato per la trasmissione dei segnali differenziali è un doppino elettrico intrecciato (twisted pair), con o senza schermo. In alcune applicazioni si fa anche uso di segnali ottici piuttosto che segnali elettrici, sfruttando le fibre ottiche.

Le specifiche che trattano il physical layer non descrivono gli aspetti meccanici del protocollo, lasciando libertà ai progettisti di utilizzare i connettori più adatti alle proprie esigenze. Ciononostante nel tempo si è andato standardizzando l'utilizzo del connettore DB9 maschio come possibile standard in applicazioni industriali. In particolare il pin-out utilizzato è:

- pin 2: CAN LOW (CAN -).
- pin 3: GND.
- pin 7: CAN HIGH (CAN +).
- pin 9: CAN V+ (Alimentazione).

Normalmente, visti i diversi standard che è possibile utilizzare, i microcontrollori che possiedono un modulo CAN interno, non soddisfano in generale nessuno degli standard sopra enunciati. Le uscite del modulo sono semplicemente rappresentate da una linea di trasmissione (CANTX) e una linea di ricezione (CANRX). Sta al progettista decidere, in base alle proprie esigenze, di connettere le uscite del microcontrollore ad un CAN transceiver piuttosto che ad un altro. Tra i moduli CAN transceiver più noti si ricordano[20]:

- TCAN1044V della Texas Instruments, CAN FD con protezione contro tensione ±58V. Supporto fino a 8Mbit/s. Supporto logica digitale 1.8V. ISO 11898-2
- TCAN337 della Texas Instruments, 3.3V CAN PHY ISO 11898-2, fino a 5Mbit/s.
- SN65HVD233 della Texas Instruments, soddisfa le specifiche ISO11898 fino ad 1Mbit/s.
- SN65HVD265 della Texas Instruments, soddisfa le specifiche ISO11898-1-2015 (CAN FD).
- ISO1042 della Texas Instruments, CAN PHY isolato con protezione linea fino a 70V, supporto FD fino a 5Mbit/s, ISO 11898-2
- MCP2551 della Microchip, soddisfa le specifiche ISO11898 fino ad 1Mbit/s.
- L9615 della ST Microelectronics, ISO11898 fino a 500Kbit/s

Tutti gli integrati possiedono un'uscita differenziale. La terminologia utilizzata per identificare i due terminali del bus sono CANH e CANL. Una ipotetica rete multi nodo è rappresentata in Figura 6.

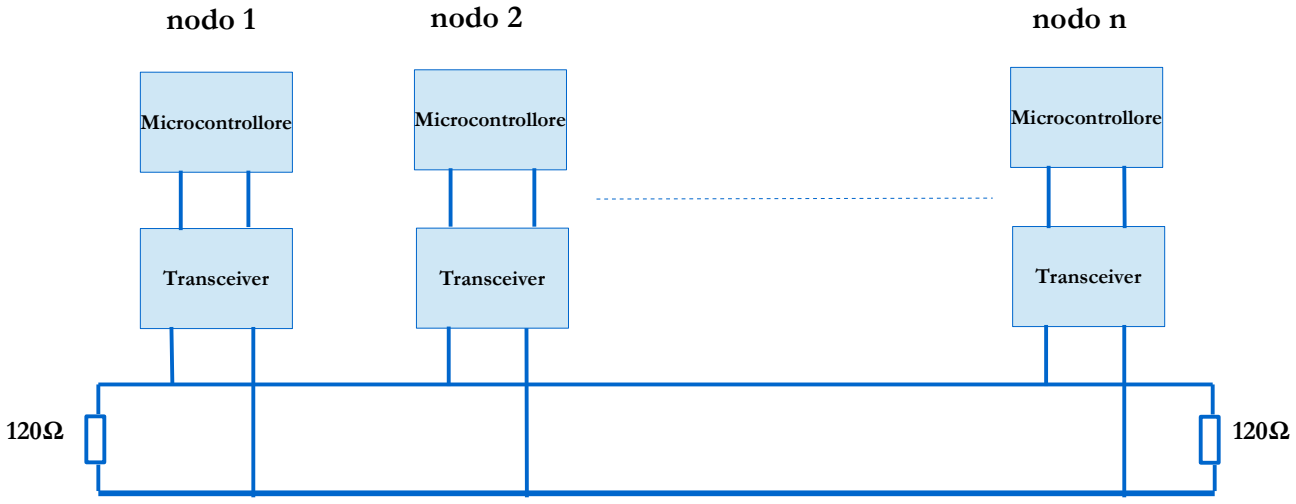

Figura 6: *Connessione di n nodi sul bus CAN.*

Il numero massimo di nodi che è possibile collegare sul Bus è teoricamente infinito ma praticamente questo viene limitato dalla lunghezza della linea, dalla massima velocità di trasmissione e dalla capacità parassita della rete. Il limite massimo di nodi è legato anche al tipo di transceiver che viene utilizzato. Per esempio nel datasheet dell'MCP2551 è specificato che il numero massimo dei nodi è 112 escludendo i limiti derivanti dalla rete stessa. I transceiver sopra descritti hanno la caratteristica di richiedere che i due nodi agli

[20] Per ulteriori informazioni sui transceiver descritti si faccia riferimento ai relativi datasheet delle case costruttrici.

estremi del bus possiedano una resistenza di terminazione da 120Ω; questo significa che la linea di trasmissione ha un'impedenza di 60Ω visto che le due resistenze sono collegate in parallelo[21].

Il protocollo CAN può essere utilizzato per trasmissioni a lunga distanza ma a seconda della distanza il valore massimo della frequenza possibile tende a diminuire.
Al fine di garantire una trasmissione robusta, non è insolito che il Bus possieda protezioni ESD (Electrostatic Discharge), common mode choke e VTS (Voltage Transient Suppressor) oltre alle protezioni elettriche spesso contenute nei transceiver stessi, richieste dalle relative norme ISO.

In Figura 7 è riportato un andamento tipico della massima lunghezza della linea di trasmissione e la massima frequenza di trasmissione utilizzabile.
Come è possibile capire, viste le elevate distanze raggiungibili, il protocollo CAN può essere utilizzato in applicazioni che vanno ben oltre gli spazi automobilistici. Vista l'elevata affidabilità del protocollo CAN, tale standard viene utilizzato in applicazioni industriali, per il controllo dei macchinari e in applicazioni domotiche, affiancandosi o sostituendo protocolli basati su interfaccia RS485.

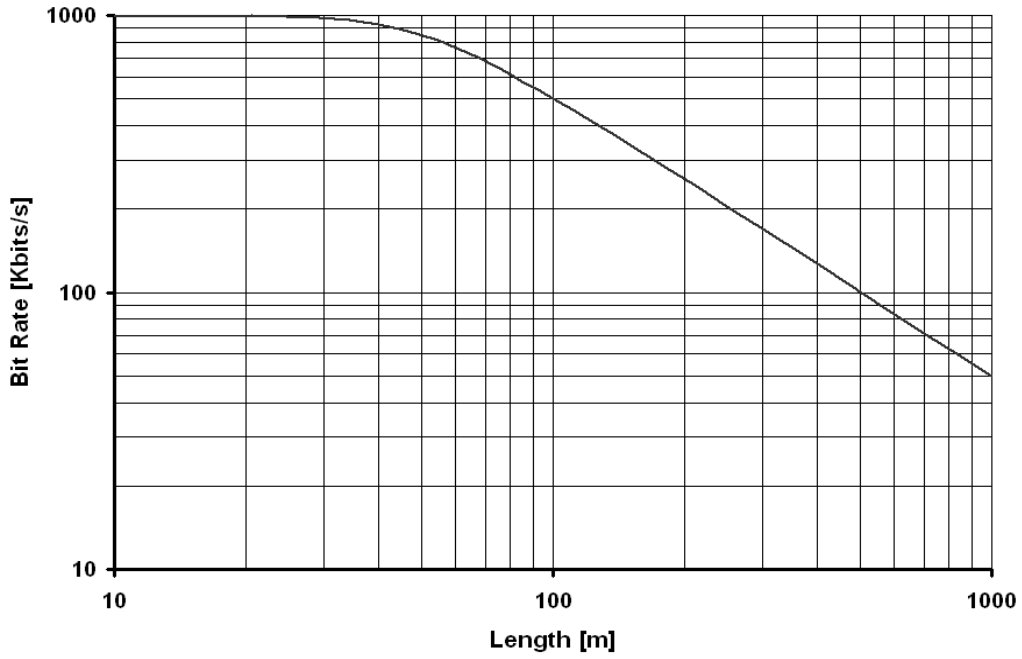

Figura 7: *Lunghezza indicativa della linea in funzione del Bit Rate.*

[21] Quanto appena affermato non è molto rigoroso né tanto meno vero, visto che non si considera l'impedenza di linea.

Le specifiche CAN FD

Le specifiche del protocollo CAN sono state aggiornate dalla Bosch nel 2013 introducendo lo standard CAN FD (Flexible Data Rate). Successivamente lo standard è stato aggiornato anche dalle normative ISO 11898-1-2015 introducendo piccole modifiche sullo stesso. Il nuovo standard modifica il formato del Data Frame estendendo il numero massimo di byte trasmessi da 8 a 64 fornendo inoltre l'opzione di trasmettere il Data Frame ad una frequenza superiore rispetto al resto del Frame. In questo modo il rapporto tra il numero di byte dati e il numero di byte totali è aumentato, permettendo di raggiungere di conseguenza un bit rate effettivo più elevato. In particolare il protocollo CAN FD permette di estendere il bit rate in applicazioni automobilistiche dal tipico 500Kb/s a 1-2Mb/s. Il protocollo CAN FD potrebbe supportare anche un bit rate superiore a 10Mb/s ma gran parte dei transceiver e tool di sviluppo supportano bit rate tra 2-5Mb/s. Da quanto appena detto si può intuire che essendo solo il Data Frame ad aumentare la frequenza, il nuovo protocollo CAN FD potrebbe coesistere in una rete CAN in cui sono presenti anche transceiver CAN 2.0, ciononostante si distinguono tre categorie di transceiver:

- **CAN FD Enabled**
  Rappresentano quella famiglia di transceiver CAN che permettono di trasmettere e ricevere messaggi CAN FD.
- **CAN FD Tollerant**
  Rappresentano quella famiglia di transceiver CAN che non supportano i messaggi FD ma non disturbano o generano errori nel caso in cui nel bus siano trasmessi messaggi CAN FD.
- **CAN Legacy**
  Rappresentano quella famiglia di transceiver CAN che supportano solo messaggi CAN standard (Frame Classici).

Oltre a supportare un numero maggiore di byte dati ed aumentare il bit rate, le specifiche CAN FD introducono anche nuove caratteristiche nella gestione del bit stuffing e il calcolo del CRC, riducendo ulteriormente la probabilità che eventuali errori possano passare inosservati. Il bit di stuffing è introdotto anche nella sezione CRC, in aggiunta a ciò lo standard ISO ha introdotto anche lo Stuff Count, ovvero tre bit prima del CRC che tengono il conteggio dei bit di stuffing introdotti.
Il CRC viene calcolato facendo uso di polinomi più complessi ed in particolare si ha il supporto del polinomio CRC17 (28 bit) fino a 16 byte dati e l'uso del polinomio CRC21 (33 bit) nel caso di un numero maggiore di byte.

Le nuove specifiche hanno portato alla definizione di un nuovo Frame, estendendo le specifiche 2.0. La modifica è implementata in maniera analoga al modo con cui è stata introdotta l'estensione 2.0B. Il dettaglio della struttura del Frame è riportato in Figura 8, in particolare si definisce formato del Frame Classico (Classical Frame Format) quello relativo alle specifiche CAN 2.0, mentre il Frame delle specifiche FD è definito Frame FD (FD Frame Format).

SOF	11 bits Identifier	RRS	IDE	FDF	rec.	BRS	ESI	4 bit DLC	Data Field	20 bits CRC	ACK	EOF	IMF

Figura 8: *Struttura del Frame nel protocollo CAN FD .*

Dove:

- RRS: Remote Request Substitution, posizione del precedente RTR (1 bit).
- IDE: Identifier Extension (1 bit).
- FDF: FD Format, recessive per identificare il formato FD (1 bit).
- BRS: Bit Rate Switch, recessive in caso di bit rate switch (1 bit).
- ESI: Error State Indicator (1 bit).
- CRC: Cyclic Redundancy Check (28 o 33 bit)
- ACK: Acknowledge (2 bit).
- EOF: End Of Frame (7 bit).
- IMF: Intermission Field (3 bit).

Si fa notare che il protocollo CAN FD non supporta il Remote Frame.

Nel caso di un Extended Frame, i bit mancanti per raggiungere 29 bit seguono il bit IDE come per il formato CAN 2.0B.

Indice Alfabetico

A

ACK....23
ACK Delimiter....10
ACK Field....10
ACK Slot....10, 16
Acknowledge....23
Acknowledgment Error....16
Acknowledgment Field....10
Active Error Frame....11
address oriented....5
air bag....4
Application Layer....6
Arbitration Field....8
autodiagnosi....16
autodiagnosticare....5
automotive....4

B

bit di stuffing....22
bit dominante....5
Bit Error....15
Bit Rate Switch....23
Bosch....4
BRS....23
bus idle....8
Bus Idle....12
Bus OFF....16

C

CAN....4
CAN -....19
CAN +....19
CAN FD....19
CAN HIGH....19
CAN LOW....19
CAN V+....19
CANRX....20
CANTX....20
Classical Frame Format....22
common mode choke....21
Confinamento degli Errori....5
connettore DB9....19
Control Field....9, 11
Controller Area Network....4
CRC....10, 22 e seg.
CRC Delimiter....10
CRC Error....15
CRC Field....10
Cyclic Redundancy Check....23

D

Data Field....10
Data Frame....7 e seg.
Data Length....11
Data Length Code....9
destuffing....11
dominante....5

E

Electrostatic Discharge....21
End of Frame....11
End Of Frame....23
EOF....23
Error Active....16
Error Delimiter....12
Error Frame....7, 11, 15
Error Passive....16
Error State Indicator....23
ESD....21
ESI....23
Extended Data Frame....14
Extended Frame....13
Extended Remote Frame....14

F

FD Format....23
FD Frame Format....22
FDF....23
Form Error....15
Frame....7
Frame Classico....22
Frame FD....22

H

Hard Synchronization....17

I

IDE....14, 23
Identifier....8
Identifier Extension....23
Identifier Extension bit....14
IMF....23
immunità ai disturbi....5
Information Processing Time....18
interfaccia RS485....21
Interframe....8
Interframe Space....12
Intermission....12
Intermission Field....23
International Standards Organization....19
IPT....18
ISO1042....20
ISO11898....5
ISO11898-1-2003....19
ISO11898-1:2015....19

ISO11898-2-2003....19
ISO11898-3-2006....19
J
J1939....19
L
L9615....20
M
master....5
MCP2551....20
Mercedes....4
message oriented....5
N
nodi....5
nodo....5
Nominal Bit Time....17
O
object layer....6
Overload Delimiter....12
Overload Flag....12
Overload Frame....8
P
Parte A....4
Parte B....4
Passive Error Frame....11
periferiche....5
Phase Buffer Segment 1....18
Phase Buffer Segment 2....18
PHASE_SEG1....18
PHASE_SEG2....18
physical layer....6
prescaler....18
Priorità dei messaggi....5
PROP_SEG....18
Propagation Time Segment....18
Protocollo multimaster....5
R
Receive Error Count....16
recessivo....5
Remote Frame....7
Remote Request Substitution....23
Remote Transmission Request....8
resistenza di terminazione....21
Resynchronization Jump Bit....18
RRS....23
RS485....21
RTR....8, 11
S
SAE....19
sincronizzazione....17
sistema antislittamento....4
SN65HVD233....20
SN65HVD265....20
Society of Automotive Engineers....19
SRR....13
Start of Frame....8
Stuff Count....22
Stuff Error....15
stuffing....11
Substitute Remote Request....13
SYNC_SEG....17
Synchronization Segment....17 e seg.
T
TCAN1044V....20
TCAN337....20
Tempi di risposta....5
transceiver....19
transfer layer....6
Transmit Error Count....16
twisted pair....19
V
Voltage Transient Suppressor....21
VTS....21

Bibliografia

[1] www.LaurTec.com : sito di elettronica dove poter scaricare gli altri articoli menzionati, aggiornamenti e progetti.
[2] http://www.can.bosch.com : sito dove poter scaricare tutta la documentazione originale della Bosch
[3] iCC 2013 : Bit Time Requirements for CAN FD (Florian Hartwich).
[4] iCC 2015 : Advantages of CAN FD Error detection mechanisms compared to Classical CAN (Dr. Arthur Mutter , Florian Hartwich).
[5] www.michrochip.com : sito dove poter scaricare la seguente documentazione:
[6] AN713 "Controller Area Network (CAN) Basics
[7] AN738 "PIC18C CAN Routine in C"
[8] AN916 "Comparing CAN and ECAN modules"
[9] DS39500A "PICmicro® 18C MCU Family Reference Manual"
[10] Datasheet PIC18F4580

History

Data	Versione	Autore	Revisione	Descrizione Cambiamento
26/03/20	2.1	Mauro Laurenti	Mauro Laurenti	Aggiornamento Formattazione. Correzioni errori di battitura. Aggiornamento lista componenti CAN PHY.
01/15/16	2.0	Mauro Laurenti	Mauro Laurenti	Nuova impaginazione. Estensione CAN FD. Estensione CAN FlexRay Aggiornamenti vari nei paragrafi (Standard, Transceiver, protezioni, ecc..).
01/01/06	1.0	Mauro Laurenti	Mauro Laurenti	Versione Originale.