Chapitre 04 : Communication et Mobilité dans les systèmes embarqués

Introduction :

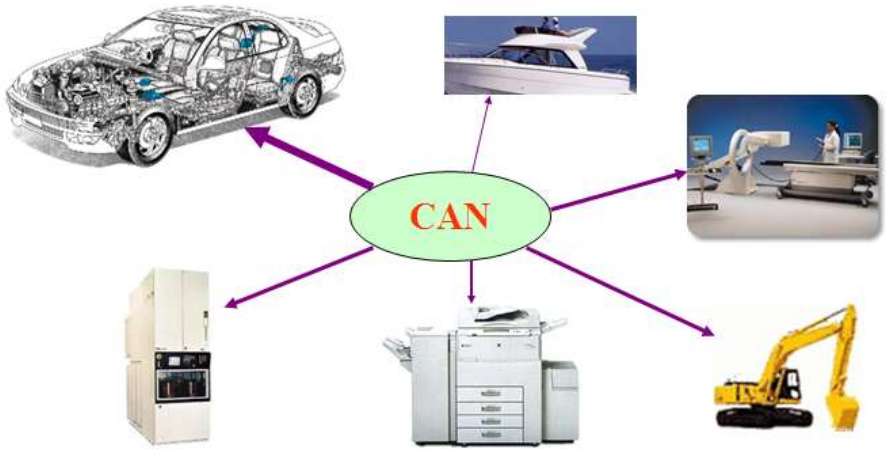
Depuis les années 1960 la longueur de câble utilisée dans l'automobile ne cesse de croître pour dépasser 2000 m en 1995. Le nombre des connexions atteint 1800 à cette même date. La fiabilité et la sécurité sont menacées.

Les normes en matière de pollution et de consommation d'énergie obligent les constructeurs à multiplier les capteurs et actionneurs intelligents dans leurs véhicules accélérant ce processus de multiplication des câbles et connexion depuis une vingtaine d'années.

Le besoin de sécurité accrue (ABS, ESP, AIR-BAG…) et la demande de confort (mémorisation des réglages de conduite, climatisation régulée par passager, système de navigation…) ne font que renforcer cette tendance.

La société BOSCH développe dès le début des années 1980 une solution de multiplexage des informations circulant à bord de la voiture. Le bus CAN apparaîtra et sera normalisé dans les années qui suivent (dès 1983).

Les composants CAN se démocratisent et investissent d'autres secteurs de l'électronique embarqué (médical, produits numériques, systèmes électrotechnique…).



2) Le bus CAN

Le bus CAN (*Control Area Network*) est un moyen de communication série qui supporte des systèmes embarqués temps réel avec un haut niveau de fiabilité. Ses domaines d'application s'étendent des réseaux moyens débits aux réseaux de multiplexages faibles coûts. Il est avant tout à classer dans la catégorie des **réseaux de terrain** utilisé dans l'industrie.

La structure du protocole du bus CAN possède implicitement les principales propriétés suivantes :

- hiérarchisation des messages.
- garantie des temps de latence.
- souplesse de configuration.
- réception de multiples sources avec synchronisation temporelle.
- fonctionnement multimaître.
- détections et signalisations d'erreurs.
- retransmission automatique des messages altérés dès que le bus est de nouveau au repos.
- distinction d'erreurs : d'ordre temporaire ou de non-fonctionnalité permanente au niveau d'un nœud, déconnexion automatique des noeuds défectueux.

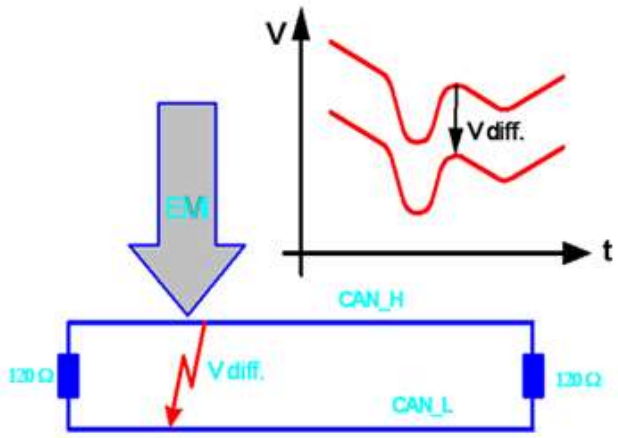
3) caractéristiques physiques du bus CAN

3.1) Support de transmission

La transmission des données est effectuée sur une paire filaire différentielle. La ligne est donc constituée de deux fils :

- CAN L (CAN LOW),
- CAN H (CAN HIGH).

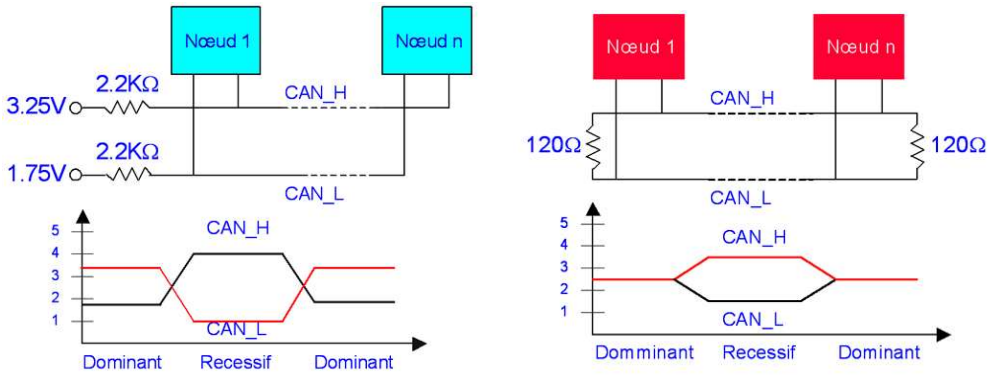
Le CAN est un bus de terrain, soumis à des parasites importants. La transmission en paire différentielle permet de s'affranchir de ces problèmes. Les montages différentiels ont en plus un fort taux de réjection en mode commun CMRR.



De par la nature différentielle de la transmission du signal sur le bus CAN, l'immunité électromagnétique est assurée car les deux lignes du bus sont toutes les deux affectées de la même manière par un signal perturbateur.

Pour les niveaux physiques sur le bus, il est important de distinguer les deux types de transmission possibles :

- transmission en bus CAN *low speed*,
- transmission en bus CAN *high speed*.

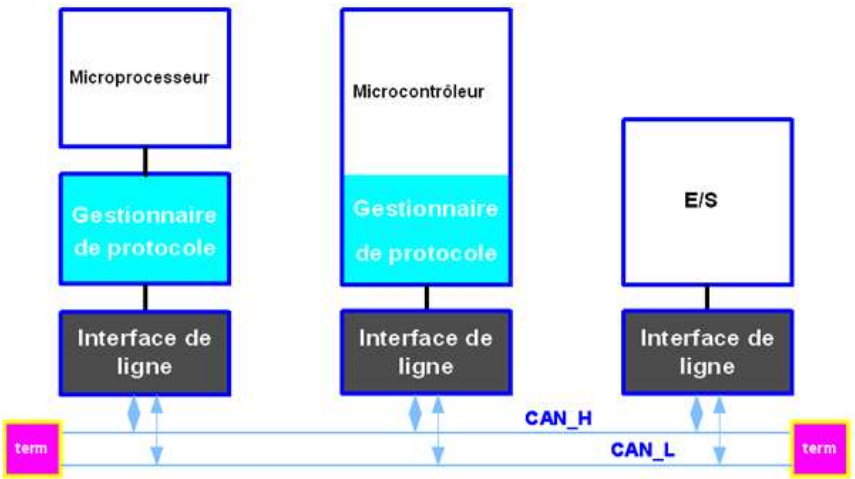


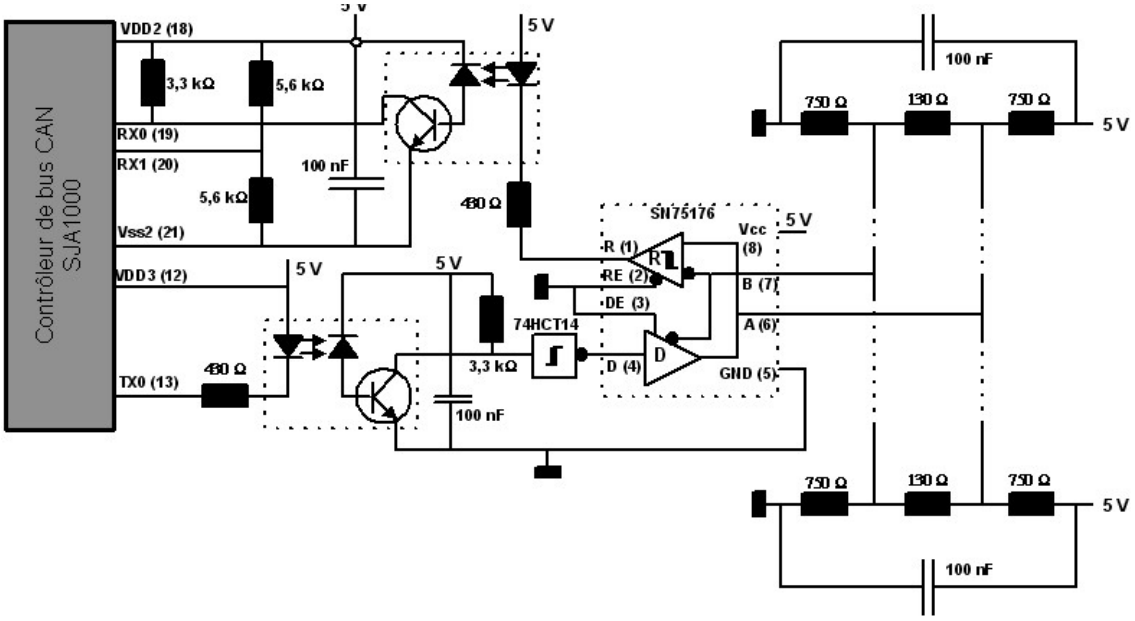
ISO11519-2 Low Speed CAN < 125KbpsISO11898 High Speed CAN 125Kbps - 1Mbps

Le tableau ci-dessous résume les principales différences entre les deux types de bus notamment sur les débits supportés.

Paramètres	CAN *low speed*	CAN *high speed*
Débit	125 kb/s	125 kb/s à 1 Mb/s
Nombre de nœuds sur le bus	2 à 20	2 à 30
Courant de sortie (mode émission)	> 1 mA sur 2,2 kΩ	25 à 50 mA sur 60Ω
Niveau dominant	CAN H = 4V CAN L = 1V	$V_{CAN\,H} - V_{CAN\,L} = 2V$
Niveau récessif	CAN H = 1,75V CAN L = 3,25V	$V_{CAN\,H} - V_{CAN\,L} = 2{,}5V$
Caractéristique du câble	30 pF entre les câbles de ligne	2*120Ω
Tensions d'alimentation	5V	5V

Exemple de circuits CAN relié au bus





4) Protocole CAN

Le concept de communication du bus CAN est celui de la diffusion d'information (broadcast) : chaque station connectée au réseau écoute les trames transmises par les stations émettrices. Ensuite chaque nœud décide quoi faire du message, s'il doit y répondre ou non, s'il doit agir ou non, etc…

Le protocole CAN autorise différents stations à accéder simultanément au bus. C'est un procédé rapide et fiable d'arbitrage qui détermine la station qui émet en premier. L'accès au bus est donc aléatoire car une station peut émettre à n'importe quel moment. Mais cet accès se fait par priorité ; cette méthode est appelée CSMA CD/AMP (*Carrier Sense Multiple Acces with Collision Detection and Arbitration Message Priority*).

Comme dans la plupart des protocoles, il est nécessaire d'utiliser un vocabulaire adapté à la situation. Nous allons donc définir un certain nombre de termes et de règles de fonctionnement concernant le protocole CAN.

- *Nœud* :
Sous-ensemble relié à un réseau de communication et capable de communiquer sur le réseau selon un protocole de communication (ici le protocole CAN).
- *Valeurs du bus* :

Le bus peut avoir l'une des deux valeurs logiques complémentaires définies, non pas en 0 et 1 comme d'habitude, mais sous les formes dites de *dominante* et *récessive*. Dans le cas d'une transmission simultanée de bits *récessifs* et *dominants*, la valeur résultante du bus sera dominante (équivalence avec un OU câblé).

- *Message* :
Chaque information est véhiculée sur le bus à l'aide d'un message (trame de bits) de format défini mais de longueur variable (et limitée). Dès que le bus est libre (bus *idle*),

n'importe quel nœud relié au réseau peut émettre un nouveau message.

- *Routage des informations* :

Des nœuds peuvent être ajoutés au réseau sans qu'il n'y ait rien à modifier tant au niveau logiciel que matériel. Chaque message possède un identificateur (*identifier*) qui n'indique pas la destination du message mais la signification des données du message. Ainsi tous les nœuds reçoivent le message, et chacun est capable de savoir grâce au système de filtrage de message si ce dernier lui est destiné ou non. Chaque nœud peut également détecter des erreurs sur un message qui ne lui est pas destiné et en informer les autres nœuds.

- *Trame de données, trame de requête* :

Une trame de données (*data frame*) est une trame qui transporte, comme son nom l'indique, des données. Une trame de requête est émise par un nœud désirant recevoir une trame de données (l'identificateur est le même pour les deux trames dans ce cas).

- *Débit bit* :

Le débit bit peut varier entre différents systèmes, mais il doit être fixe et uniforme au sein d'un même système.

- *Priorités* :

Les identificateurs de chaque message permettent de définir quel message est prioritaire sur tel autre.

- *Demande d'une trame de données* :

Un nœud peut demander à un autre nœud d'envoyer une trame de données, et pour cela il envoie lui-même une trame de requête. La trame de données correspondant à la trame de requête initiale possède le même identificateur.

- *Fonctionnement multimaître* :

Lorsque le bus est libre, chaque nœud peut décider d'envoyer un message. Seul le message de plus haute priorité prend possession du bus.

- *Arbitrage* :

Le problème de l'arbitrage résulte du fonctionnement multimaître. Si deux nœuds ou plus tentent d'émettre un message sur un bus libre il faut régler les conflits d'accès. On effectue alors un arbitrage bit à bit (non destructif) tout au long du contenu de l'identificateur. Ce mécanisme garantit qu'il n'y aura ni perte de temps, ni perte d'informations. Dans le cas de deux identificateurs identiques, la trame de données gagne le bus. Lorsqu'un bit récessif est envoyé et qu'un bit dominant est observé sur le bus, l'unité considérée perd l'arbitrage, doit se taire et ne plus envoyer aucun bit. L'arbitrage est qualifié de CSMA/CA (*Carrier Sense Multiple Access - Collision Avoidance*).

- *Sécurité de transmission* :

Dans le but d'obtenir la plus grande sécurité lors de transferts sur le bus, des dispositifs de signalisation, de détection d'erreurs, et d'autotests ont été implémentés sur

chaque nœud d'un réseau CAN. On dispose ainsi d'un monitoring bus (vérification du bit émis sur le bus), d'un CRC (*Cyclic Redundancy Check*), d'une procédure de contrôle de l'architecture du message, d'une méthode de *Bit-Stuffing*. On détecte alors toutes les erreurs globales, toutes les erreurs locales au niveau des émetteurs, jusqu'à 5 erreurs aléatoires réparties dans un message. La probabilité totale résiduelle de messages entachés d'erreurs est inférieure à 4.7*10-11.

- *Signalement des erreurs et temps de recouvrement des erreurs* :

Tous les messages entachés d'erreur(s) sont signalés au niveau de chaque nœud par un *flag*. Les messages erronés ne sont pas pris en compte, et doivent être retransmis automatiquement.

- *Erreurs de confinement* :

Un nœud CAN doit être capable de faire les distinctions entre des perturbations de courtes durées et des dysfonctionnements permanents. Les nœuds considérés comme défectueux doivent passer en mode *switched off* en se déconnectant (électriquement) du réseau.

- *Points de connexion* :

La liaison de communication série CAN est un bus sur lequel un nombre important d'unités peuvent être raccordées. En pratique le nombre total d'unités sera déterminé par les temps de retard (dus aux phénomènes de propagation) et/ou les valeurs des charges électriques que ces unités présentent sur le bus.

- *Canal de liaison simple* :

Le bus consiste en un simple canal bidirectionnel qui transporte les bits. A partir des données transportées, il est possible de récupérer des informations de resynchronisation. La façon dont le canal est implémenté (fil standard, liaison optique, paire différentielle…) n'est pas déterminée dans la norme officielle BOSCH.

- *Acquittement* :

Tous les récepteurs vérifient la validité d'un message reçu, et dans le cas d'un message correct ils doivent acquitter en émettant un flag.

- *Mode 'Sleep' (sommeil), Mode 'Wake-up' (réveil)* :

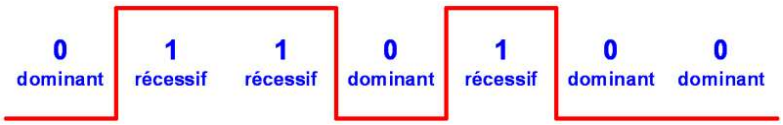
Afin de réduire la consommation d'énergie, chaque élément CAN peut se mettre en *Sleep mode*. Dans ce mode il n'y a aucune activité interne au nœud CAN considéré et ses drivers sont déconnectés du bus. La reprise de fonctionnement (mode *Wake-up*) s'effectue lorsqu'il y a une activité sur le bus ou par décision interne à l'élément CAN. On observe une attente due à une resynchronisation de l'oscillateur local qui teste la présence de 11 bits consécutifs sur le bus (l'activité interne au nœud CAN a cependant repris). Par suite les drivers se reconnectent au bus. Afin d'obtenir les meilleures performances en débit sur un réseau de type CAN, il est nécessaire d'utiliser des oscillateurs à quartz.

5) Caractéristiques du Bus CAN

5.1) Le codage NRZ : bits dominants et récessifs :

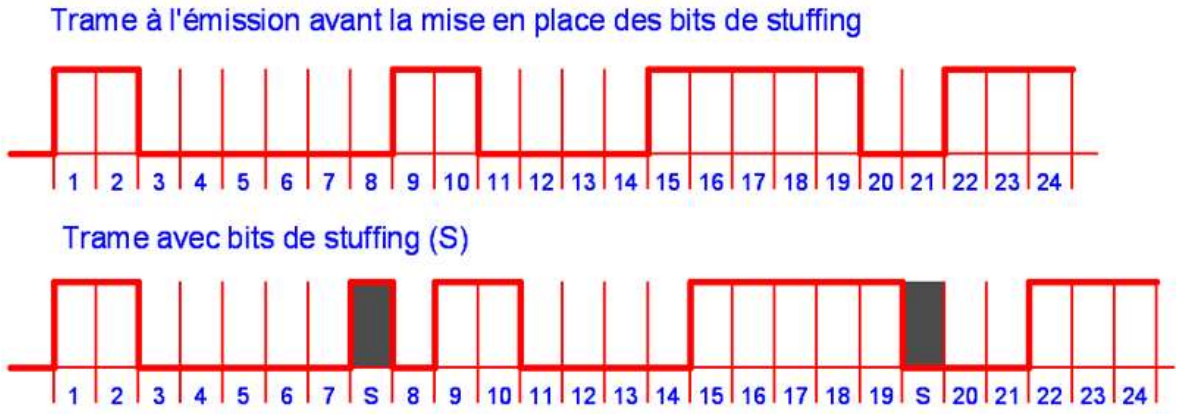
La succession de bits transitant sur le bus est codé avec la méthode du NRZ (Non Return To Zero).

Pendant la durée totale du bit, le niveau de tension de la ligne est maintenu, c'est à dire que pendant toute la durée durant laquelle un bit est généré, sa valeur reste constante qu'elle soit dominante ou récessive.



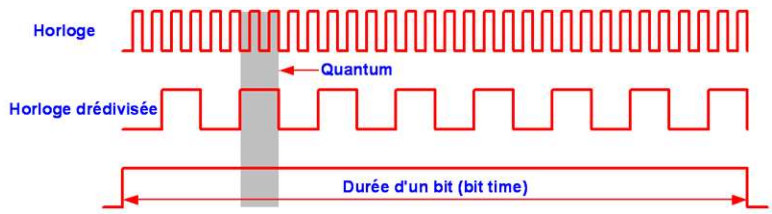
5.2) Le bit stuffing

Une des caractéristiques du codage NRZ est que le niveau du bit est maintenu pendant toute sa durée. Cela pose des problèmes de fiabilité si un grand nombre de bits identiques se succèdent. La technique du Bit Stuffing impose au transmetteur d'ajouter automatiquement un bit de valeur opposée lorsqu'il détecte 5 bits consécutifs dans les valeurs à transmettre.



5.3) Le bit timing

On définit la plus petite base de temps reconnue sur un bus CAN comme étant le *Time Quantum*. Cette base de temps est une fraction de l'horloge de l'oscillateur du bus. Un bit dure entre 8 et 25 quantum

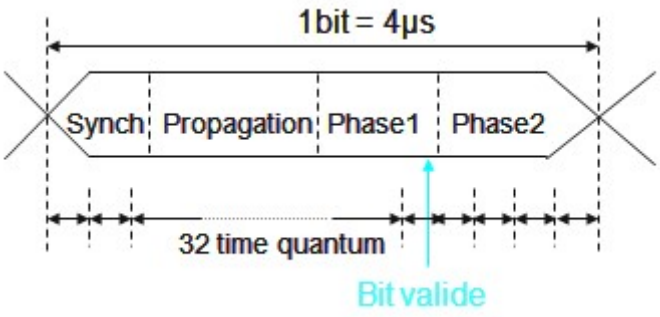


<u>Exemple de bit timing : lecture d'un bit</u>

ISO11898 : High Speed CAN 250 Kbps

- 1 bit correspond à 32 coup d'horloge

- La lecture du bit devra être faite au 20éme coup d'horloge



5.4) Longueur du bus et débit

La longueur du bus dépend des paramètres suivants :

1. Le délai de propagation sur les lignes physiques du bus.
2. La différence du quantum de temps défini précédemment, du aux différences de cadencement des oscillations des nœuds.
3. L'amplitude du signal qui varie en fonction de la résistance du câble et de l'impédance d'entrée des nœuds.

Pour une longueur de bus supérieure à 200 mètres il est nécessaire d'utiliser un optocoupleur, et pour une longueur de bus supérieure à 1 kilomètre il est nécessaire d'utiliser des systèmes d'interconnexion tels que des répéteurs ou des ponts. N'importe quel module connecté sur un bus CAN doit pouvoir supporter un débit d'au moins 20 kbit/s

Débit	Longueur	Longueur d'un bit
1 Mbit/s	30 m	1 µs
800 kbit/s	50 m	1,25 µs
500 kbit/s	100 m	2 µs
250 kbit/s	250 m	4 µs
125 kbit/s	500 m	8 µs
62,5 kbit/s	1000 m	16 µs
20 kbit/s	2500 m	50 µs
10 kbit/s	5000 m	100 µs

6) Les informations sur le bus

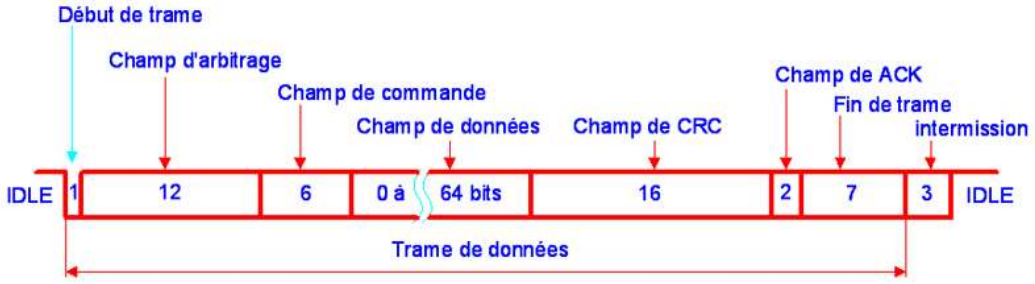
6.1) Trame de données (data frame)

Constitution de la trame de données de type standard CAN 2.0A, la plus utilisée. Cette trame se décompose en sept parties principales que l'on appelle des *champs* :

- début de trame (1 bit) *start off frame* (SOF)
- champ d'arbitrage (12 bits) *arbitration field*
- champ de commande (6 bits) *control field*
- champ de données (0 à 64 bits) *data field*

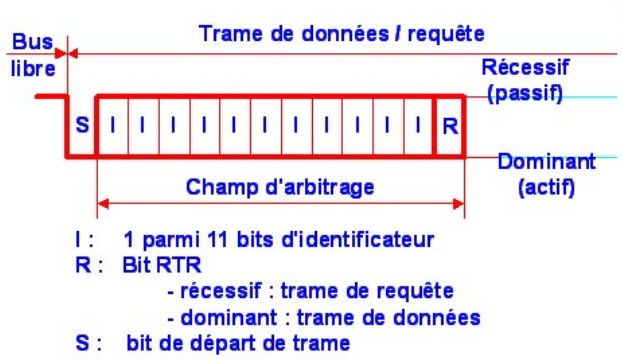
- champ de CRC (16 bits) *CRC sequence*
- champ d'acquittement (2 bits) *ACKnowledgement field*
- fin de trame (7 bits) *end of frame* (EOF)
- puis, une 8emme zone dite d'espace interframe (*intertrame*) qui fait partie intégrante de la trame.

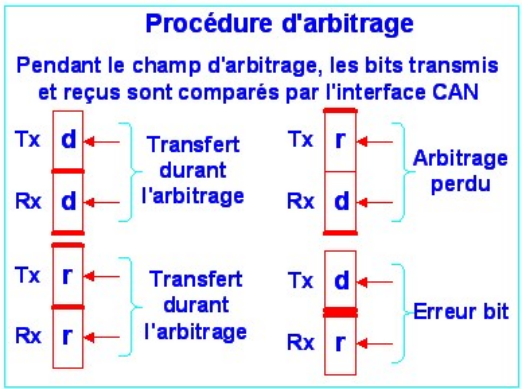
6.2) Les champs de la trame de données



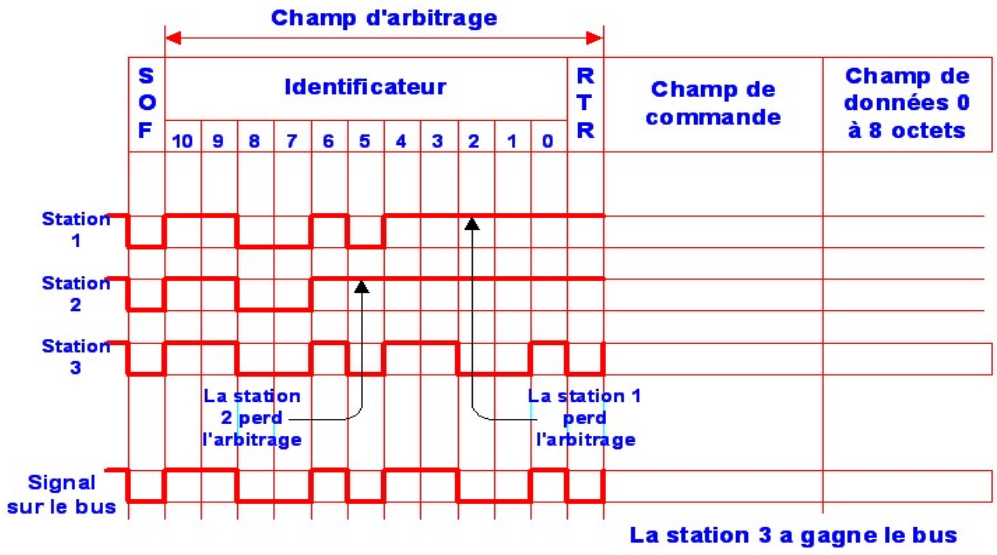
6.3) La méthode d'arbitrage

Le champ pendant lequel s'effectue l'arbitrage est constitué des bits de l'identifier ainsi que du bit immédiatement suivant dit RTR (*Remote Transmission Request*).





Exemple d'arbitrage



6.4) Rôle des bits dans le champ d'arbitrage:

Le bit SOF (début de trame de données)

C' est dominant il signale à toutes les stations le début d'un échange. Cet échange ne peut démarrer que si le bus était précédemment au repos. Toutes les stations doivent se synchroniser sur le front avant la transition du bit de départ.

Identificateur :

La longueur de l'identificateur est de 11 bits, les bits sont transmis dans l'ordre de ID_10 à ID_0 (le moins significatif est ID_0). Par ailleurs les 7 bits les plus significatifs (de ID_10 à ID_4) ne doivent pas être tous récessifs.

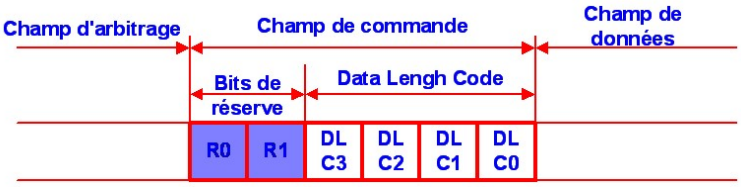
ID = 1111111XXXX (X valeur indéterminée), c'est-à-dire un nombre maximal d'identificateurs de : (211 - 24) = 2048 - 16 = 2032 combinaisons.

Le bit RTR :

Lors d'une *dataframe*, le bit de remote *transmission request* (RTR) doit être dominant.

6.5) Champ de commande

Il est constitué de 6 bits.



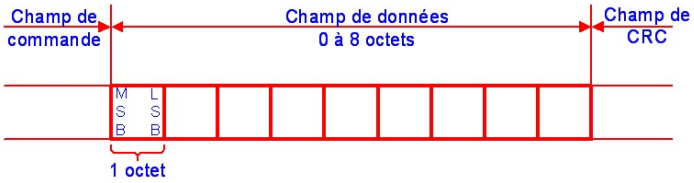
Il y a deux Bits de réserves : Les deux premiers bits (émis dominants en trame 2.0A) sont en réserve d'usages ultérieurs et permettent d'assurer des compatibilités futures ascendantes (notamment celles de la trame dite étendue CAN 2.0B). Les contrôleurs CAN doivent être aptes à traiter toutes combinaisons de tous les bits du champ de commande.
4 bits DLC : Les 4 derniers bits du champ de commande (champ DLC - *Data Length Code*) indiquent le nombre d'octets qui seront contenus dans le champ de données.

Nombre d'octet	DLC 3	DLC 2	DLC 1	DLC 0
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

6.6) Champ de données

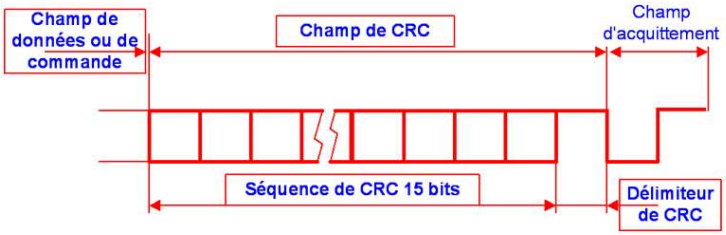
Le champ de données est l'endroit où se trouvent les données utiles transmises. Il peut être composé de 0 octet minimum à 8 octets maximum transmis avec le MSB (*Most Significant Bit*) en tête.

Remarque : De 0 à 8 inclus, cela fait neuf valeurs donc 4 bits du DLC pour définir le nombre de données contenues



6.7) Le champ de CRC :

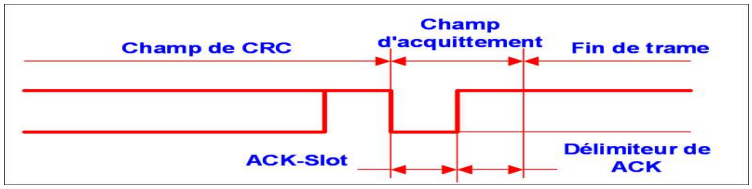
Il est composé de la séquence de CRC sur 15 bits suivi du CRC Delimiter (1 bit récessif). La séquence de CRC (Cyclic Redundancy Code) permet de vérifier l'intégrité des données transmises. Les bits utilisés dans le calcul du CRC sont ceux du SOF, du champ d'Arbitration, du champ de Control et du champ Data Field.
Le CRC est un polynôme calculé de la même manière par l'émetteur et par le récepteur de la trame : le message est vu par l'algorithme comme un polynôme qui est divisé par $X^{15}+X^{14}+X^{10}+X^{8}+X^{7}+X^{4}+X^{3}+1$ et le reste de cette division est la séquence CRC transmise avec le message.



6.8) Le champ ACK

Il est composé de 2 bits, l'ACK Slot et le ACK Delimiter (1 bit récessif).

- un nœud en train de transmettre envoie un bit récessif pour le ACK Slot.
- un nœud ayant reçu correctement un message en informe le transmetteur en envoyant un bit dominant pendant le ACK Slot : il acquitte le message.



6.9) Fin de trame de donnée

La trame de donnée se termine par un drapeau formé par une séquence de 7 bits récessifs, ce qui, dépasse de deux bits la largeur de la norme de *bit stuffing*.

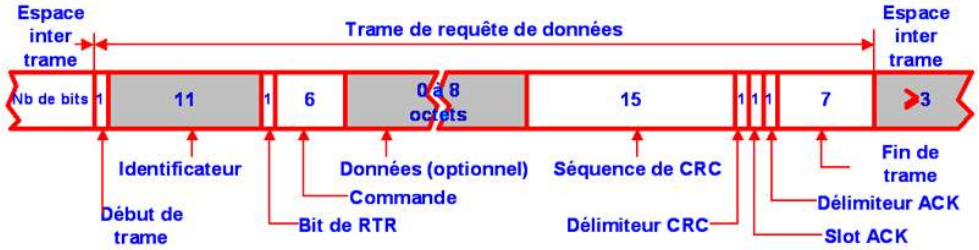
Ce champ a une structure fixe et les logiques de codage (à l'émission) et de décodage (aux réceptions) de *bit stuffing* sont désactivées pendant la séquence du champ de fin de trame.

6.10) Trame de requête (remote frame)

Chacun émet sans savoir si l'information envoyée à servi à l'un des participants. Il se peut aussi qu'un nœud ait besoin d'information d'un certain type dont il ne dispose pas pour assurer la mission qui lui est dévolue. Dans ce cas, une station nécessitant des données peut initialiser la demande d'une transmission des données considérées par un autre nœud en envoyant une *remote frame*. Cette trame ne se compose que de six parties au lieu des sept précédentes :

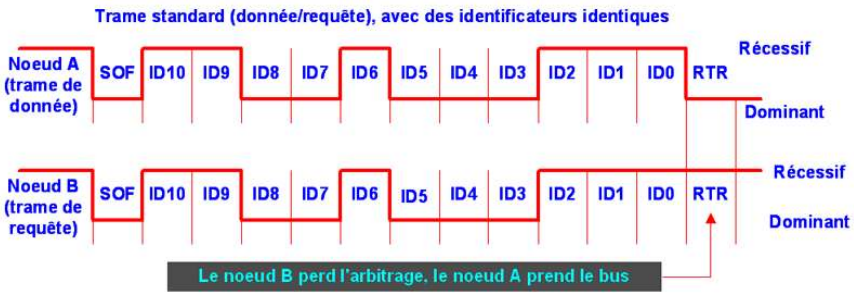
1. - le début de trame,
2. - le champ d'arbitrage,
3. - le champ de commande,
4. - le champ de CRC
5. - le champ d'acquittement,
6. - la fin de trame,
7. - puis une 7e zone dite d'espace *interframe*.

Format de la trame de requête



Contrairement au cas précédent, dans le cas d'une *remote frame*, le bit RTR est récessif. C'est donc ce bit qui différencie une *data frame* d'une *remote frame*.

Exemple :
Comparaison de 2 trames avec le même identificateur, l'une de données l'autre de requête : la trame de donnée est prioritaire sur la trame de requête.



6.11) Trame de surcharge (overload frame)

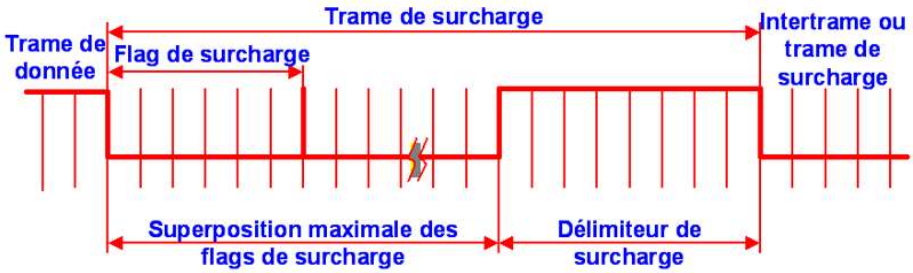
Cette trame indique qu'une station est surchargée pendant un certain laps de temps. Il y a deux sortes de conditions de surcharge qui mènent toutes deux à la transmission d'un *overload flag* :

- les conditions internes d'un récepteur qui nécessitent un certain temps (un retard) pour accepter la prochaine *data frame* ou *remote frame*.
- la détection d'un bit dominant durant la phase intermission. Dans ce cas le démarrage de *l'overload frame* a lieu juste après la détection du bit dominant.

Afin de ne pas bloquer le bus indéfiniment seules deux *overload frame* consécutives peuvent être générées pour retarder les *data* ou *remote frame* suivantes. Cette trame ne comprend que deux champs :

- Le champ des *flags* de surcharge,
- Le délimiteur de champ.

Comme l'indique la figure, elle peut se produire à la fin d'un *end of frame* ou d'un *error delimiter* ou encore d'un autre *overload delimiter* en lieu et place du début de l'*interframe*.



6.12) Période d'intertrame (interframe)

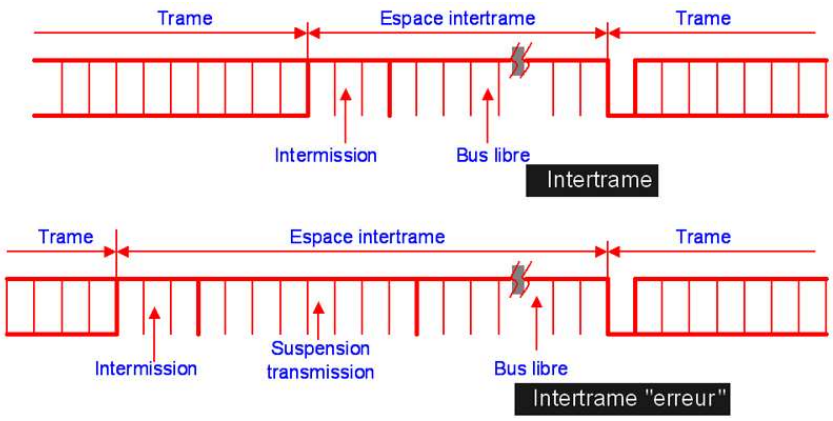
Les *data frame* et *remote frame* sont séparées des trames précédentes (de quelques types qu'elles soient : *data, remote, error, overload frame*) par un champ de bits appelé *interframe space*.

Au contraire, les *overload frame* et *error frame* ne sont pas précédées par un *interframe space* et les multiples *overload frame* ne sont pas séparées par un *interframe space* (revoir toutes les figures déjà présentées et observer en détail ces phases de fonctionnement du bus).

L'*interframe space* se compose de deux ou trois champs selon les cas. Ce sont :

- le champ de bits *intermission*
- le champ de bits de *bus idle* (bus libre),
- d'un champ de bits de *suspend transmission*, pour les stations en *error passive* qui ont envoyé un message d'erreur.

Voici un exemple de 2 zones d'inter trame l'une sans trame d'erreur l'autre à la suite d'une trame d'erreur (inter trame « erreur »)



6.13) La trame d'erreur

Pour différentes raisons, comme l'existence de fortes perturbations ou de pertes importantes lors de la transmission, le protocole CAN dispose d'un système de gestion des erreurs locales.

Le principe du bit stuffing vu précédemment permet de localiser une erreur et un nœud qui détecte ce type d'erreur transmettra aux autres nœuds un message dit « Error Flag » contenant six bits de même polarité.

Après avoir transmis le message Error Flag, le nœud essaiera à nouveau de transmettre le message, et si aucun message de priorité supérieure ne prend la main sur le réseau ce nouveau message est transmis 23 bits au plus après.

Les bits formant l'Error Flag sont dominants et écrasent donc les données contenues dans la Data Frame. Ils provoquent la retransmission de cette dernière. Dans le cas d'erreurs successives, il y aura superposition d'Error Flags.

La trame d'erreur

Les 8 bits de l'Error Délimiter donnent l'autorisation aux nœuds du réseau de reprendre leurs communications.

Des recherches ont montré que le taux d'erreurs non détectées par le protocole CAN est très faible : 1 erreur non détectée pour 1000 années de fonctionnement