

- **ENSEIRB** -



LE BUS CAN

Patrice KADIONIK
kadionik@enseirb.fr
<http://www.enseirb.fr/~kadionik>

TABLE DES MATIERES

1. Introduction	3
2. Le protocole CAN.....	3
3. Protocole CAN et couches OSI	3
4. Quelques règles de fonctionnement et définitions.....	5
5. Trame de données	8
5.1. Champ d'arbitrage.....	9
5.2. Champ de contrôle	10
5.3. Champ de données	11
5.4. Champ de CRC	11
5.5. Champ d'acquittement.....	12
5.6. Champ de fin de trame.....	13
6. Trame de requête	13
7. Traitement des erreurs	14
7.1. Les différents types d'erreurs	14
7.2. Les trames d'erreurs	16
7.3. Recouvrement des erreurs	17
8. Fin de trames CAN	20
8.1. Trame de surcharge	20
8.2. Période d'intertrame	20
8.3. Autres modes.....	21
9. Codage de ligne.....	21
10. Le Nominal Bit Time	22
10.1. Description des différents segments	23
10.2. Durée des différents segments et notion de <i>Time Quantum</i>	24
11. Synchronisation des horloges	25
11.1. Notion de <i>RJW</i>	26
11.2. Notion d'erreur de phase.....	26
11.3. Les règles de synchronisation.....	27
12. Caractéristiques physiques du bus CAN	29
12.1. Support de transmission	29
12.2. Débit sur le réseau et temps de latence.....	31
13. Bibliographie	32
14. Webographie.....	32

1. INTRODUCTION

Ce document a pour but de présenter les caractéristiques essentiels du bus/réseau de terrain CAN (*Control Area Network*). Bien plus qu'un bus au sens électrique, le bus CAN est un réseau à part entière respectant le modèle d'interconnexion des systèmes ouverts OSI de l'ISO. C'est un réseau de terrain aussi car il doit fonctionner dans un environnement limité et sévère comme une usine, un atelier, une voiture...

Le bus/réseau CAN, standard de fait, est promu à un essor rapide.

2. LE PROTOCOLE CAN

Le protocole CAN (*Control Area Network*) est un protocole de communication série qui supporte des systèmes 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 pour remplacer la boucle analogique 20mA.

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 nœuds défectueux.

En étudiant la norme BOSCH on se rend compte que le protocole CAN ne couvre seulement que deux des sept couches du modèle d'interconnexion des systèmes ouverts OSI de l'ISO.

3. PROTOCOLE CAN ET COUCHES OSI

On retrouve ainsi dans le protocole CAN, la couche liaison de données (couche 2) et la couche physique (couche 1) (figure 1). La couche de liaison de données est subdivisée en deux sous-couches (LLC *Logic Link Control*), et MAC (*Medium Access Control*), tandis que la couche physique est divisée en trois sous-couches (PLS *Physical Signalling*), PMA (*Physical Medium Access*), MDI (*Medium Dependent Interface*).

La sous-couche MAC représente le noyau du protocole CAN. Elle a pour fonction de présenter les messages reçus en provenance de la sous-couche LLC et d'accepter les messages devant être transmis vers la sous-couche LLC. Elle est responsable de :

- la mise en trame du message.
- l'arbitrage.
- l'acquittement.

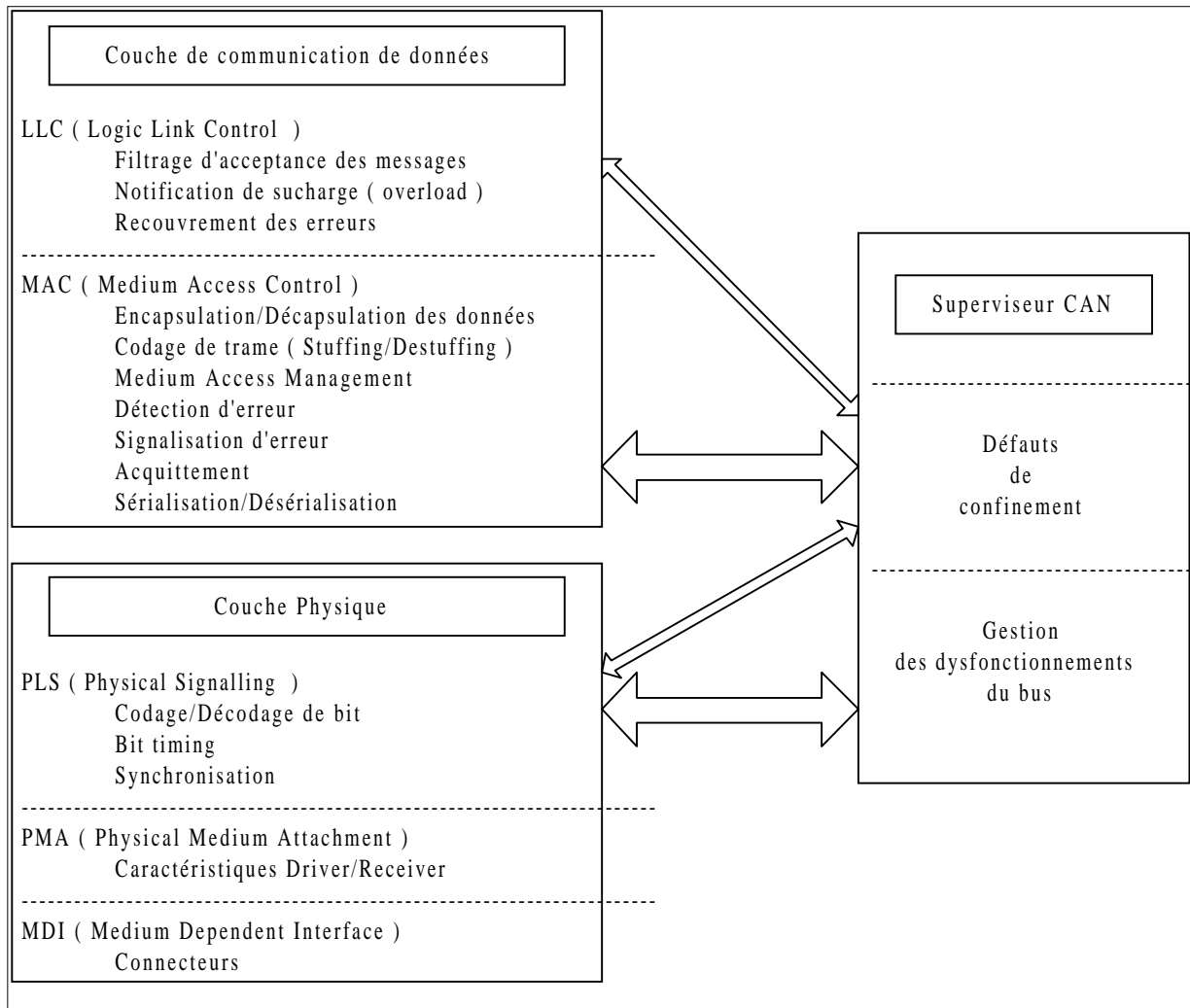


Figure 1 : Le protocole CAN et le modèle OSI

- la détection des erreurs.
- la signalisation des erreurs.

Elle est supervisée par une entité de supervision qui est un mécanisme apte à faire la distinction entre les dérangements de courtes durées et des pannes permanentes.

La sous-couche LLC s'occupe quant à elle :

- du filtrage des messages.
- de la notification de surcharge (*Overload*).
- de la procédure de recouvrement des erreurs.

La couche physique définit comment le signal est transmis et a pour conséquent pour rôle d'assurer le transfert physique des bits entre les différents nœuds en accord avec toutes les propriétés (électriques, électroniques...) du système. Il est évident qu'à l'intérieur d'un même et unique réseau la couche physique doit être la même pour chaque nœud. Cette couche s'occupe donc :

- de gérer la représentation du bit (codage, timing...).
- de gérer la synchronisation bit.
- de définir les niveaux électriques des signaux.
- de définir le support de transmission.

4. QUELQUES REGLES DE FONCTIONNEMENT ET DEFINITIONS

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 ET 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 $4.7 \cdot 10^{-11}$.
 - *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.

Par ailleurs il existe deux types de format (trame standard, trame étendue) pour les trames de données et de requête, et ils diffèrent seulement l'un de l'autre par l'identificateur (identificateur de 11 bits pour les trames standards, de 29 bits pour les trames étendues).

Le transfert des messages se manifeste et est commandé à l'aide de quatre types de trames spécifiques et d'un intervalle de temps les séparant. Outre les trames de données et de requête, on a donc également des trames d'erreurs (émises par n'importe quel nœud dès la détection d'une erreur), et des trames de surcharge (ces trames correspondent à une demande d'un laps de temps entre les trames de données et de requête précédentes et successives). Il existe un espace intertrame de 3 bits récessifs entre les trames de données et de requête.

En ce qui concerne le flot de bits des trames du bus CAN, la méthode de codage *NRZ (Non Return to Zero)* a été retenue. Ceci revient à dire que pendant la durée totale du bit généré son niveau reste constant qu'il soit dominant ou récessif.

De plus afin de sécuriser la transmission des messages on utilise la méthode dite de *Bit-Stuffing* (bit de transparence). Cette méthode consiste, dès que l'on a émis 5 bits de même polarité sur le bus, à insérer un bit de polarité contraire pour casser des chaînes trop importantes de bits identiques. On obtient ainsi dans le message un plus grand nombre de transitions ce qui permet de faciliter la synchronisation en réception par les nœuds. Cette technique est uniquement active sur les champs de SOF, d'arbitrage, de contrôle, de CRC (délimiteur exclu). Pour un fonctionnement correct de tout le réseau, cette technique doit être implémentée aussi bien à la réception qu'à l'émission.

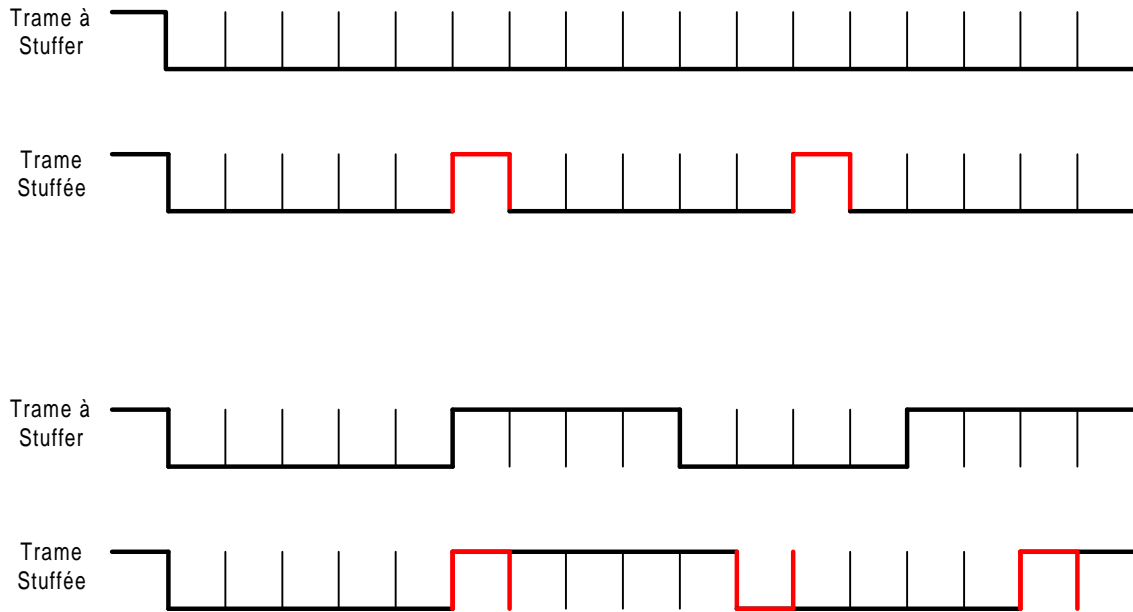


Figure 2 : Technique de *Bit-Stuffing*

5. TRAME DE DONNEES

Une trame de données se décompose en 7 champs différents (figure 3) :

- le début de trame SOF (*Start Of Frame*), 1 bit dominant.
- le champ d'arbitrage, 12 bits.
- le champ de contrôle, 6 bits.
- le champ de données, 0 à 64 bits.
- le champ de CRC (*Cyclic Redundancy Code*), 16 bits.
- le champ d'acquiescement (*Acknowledge*), 2 bits.
- le champ de fin de trame EOF (*End Of Frame*), 7 bits récessifs.

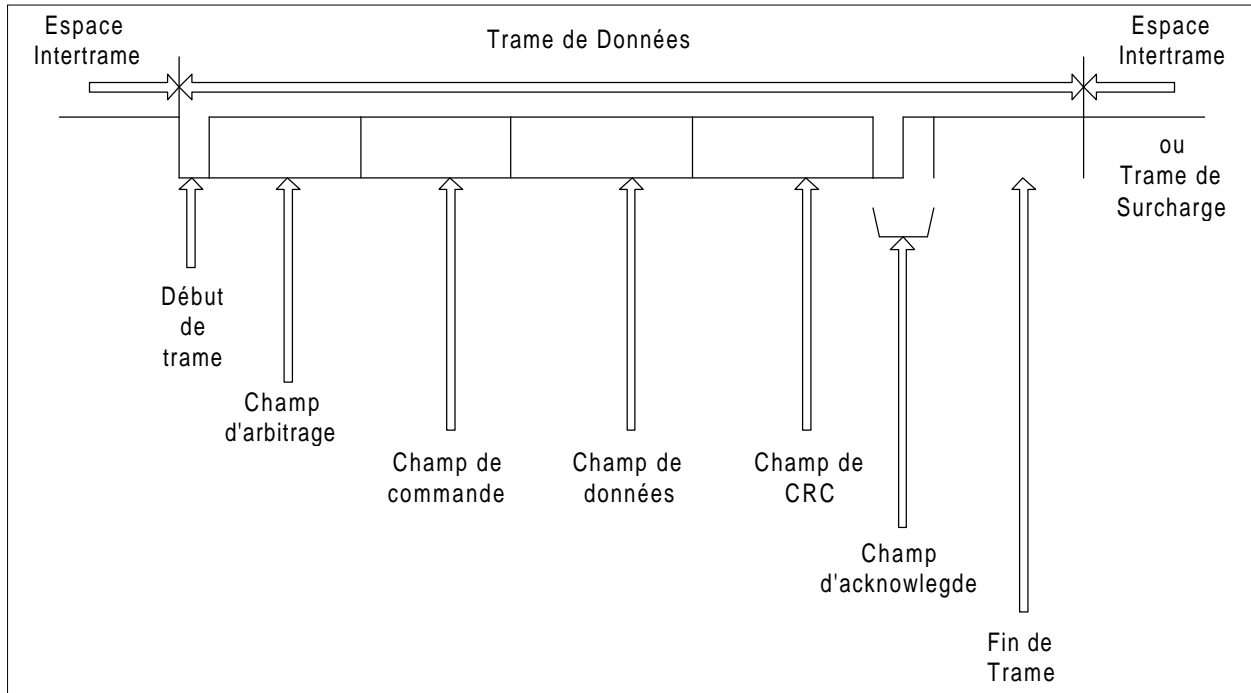


Figure 3 : Format de la trame CAN

Le début de trame n'est effectif que si le bus était précédemment au repos. Tous les nœuds du réseau doivent se resynchroniser sur le bit de SOF.

5.1. Champ d'arbitrage

Dans une trame standard, le champ d'arbitrage est composé des 11 bits de l'identificateur (figure 4) et d'un bit de RTR (*Remote Transmission Request*) qui est dominant pour une trame de données et récessif pour une trame de requête. On ne détaillera pas ici le champ d'arbitrage pour une trame. Ceux qui désire voir plus de détails sur ce point peuvent faire appel à la norme BOSCH. Pour l'identificateur 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 tous être récessifs. Pour des raisons de compatibilité avec des anciens circuits, les 4 derniers bits de l'identificateur (ID_3 à ID_0) ne sont pas utilisés, ce qui réduit le nombre de combinaisons possibles.

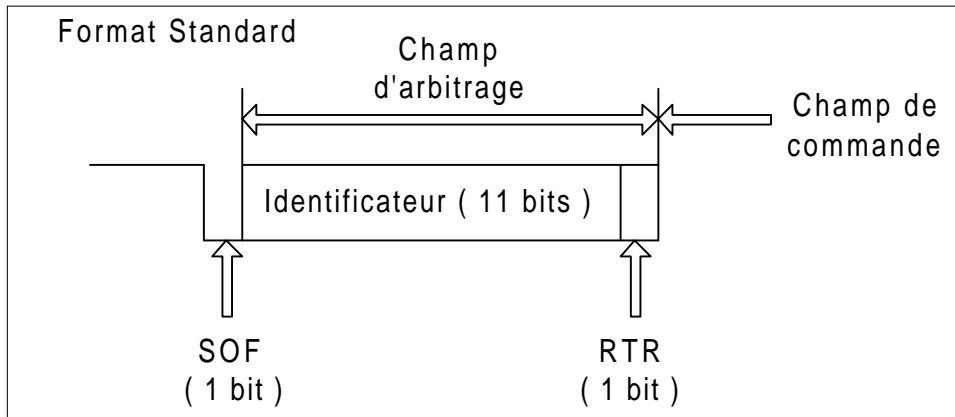


Figure 4 : Champ arbitrage

5.2. Champ de contrôle

Le champ de contrôle est composé de 6 bits (figure 5). Les deux premiers (r1 dans une trame standard, et r0) sont des bits de réserve et leur rôle est d'assurer des compatibilités futures ascendantes (par exemple avec les trames étendues). Les quatre derniers bits permettent de déterminer le nombre d'octets de données contenus dans le champ de données pour une trame de données ou bien le nombre d'octets de données dont a besoin un nœud du réseau lors d'une trame de requête. Le nombre d'octets de données ne peut pas excéder la valeur de 8.

Taille des données en octets	DLC (<i>Data Length Code</i>)			
	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

D : bit Dominant, R : bit Récessif

Tableau 1 : Codage des bits DLC suivant la taille des données en octets

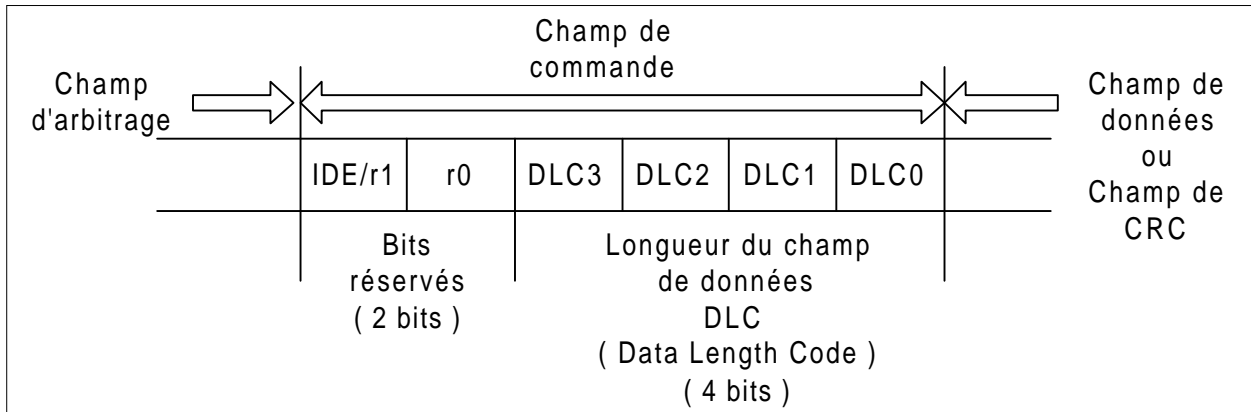


Figure 5 : Champ contrôle

5.3. Champ de données

Le champ de données a une longueur qui peut varier de 0 à 64 bits (0 à 8 octets). Cette longueur a été déterminée lors de l'analyse du champ de contrôle. Dans le cas d'une trame de requête, le champ de données est vide.

5.4. Champ de CRC

Le champ de CRC est composé de 16 bits (figure 6). La séquence CRC calculée est contenue dans les 15 premiers bits tandis que le dernier bit est un délimiteur de fin de champ de CRC (bit toujours récessif).

Ce champ de CRC permet de s'assurer de la validité du message transmis, et tous les récepteurs doivent s'astreindre à ce procédé de vérification. Seuls les champs de SOF, d'arbitrage, de contrôle et de données sont utilisés pour le calcul de la séquence de CRC. Les codes utilisés par les contrôleurs de bus CAN sont des codes linéaires de. De fait la longueur maximale du début de trame ne doit pas excéder 2^{15} bits pour une séquence de CRC de 15 bits. Le nombre maximal d'erreurs détectées dans la trame est de 5.

La séquence de CRC est calculée par la procédure suivante :

- le flot de bits (hors *Bit-Stuffing*), constitué des bits depuis le début de la trame jusqu'à la fin du champ de données (pour une trame de données) ou bien la fin du champ de contrôle (pour une trame de requête) est interprétée comme un polynôme $f(x)$ avec des coefficients 0 et 1 affectés à la présence, effective ou non, de chaque bit. Le polynôme obtenu est alors multiplié par x^{15} complété pour l'ajout du mot de CRC.
- le polynôme ainsi formé est divisé (modulo 2) par le polynôme générateur $g(x)=x^{15}+x^{14}+x^{10}+x^8+x^7+x^4+x^3+1$. La chaîne de bits correspondante à ce polynôme est : 1100010110011001.
- Le reste de la division du polynôme $f(x)$ par le polynôme générateur $g(x)$ constitue la séquence CRC de 15 bits.

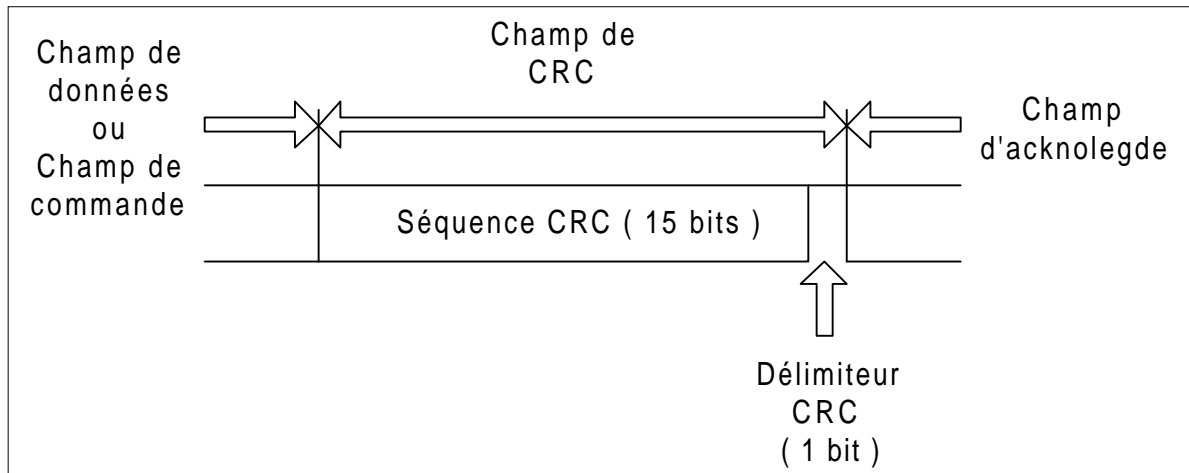


Figure 6 : Champ CRC

La réalisation du module de calcul de CRC est particulièrement aisée à l'aide de registres à décalages. La norme BOSCH propose le programme informatique correspondant à l'algorithme précédemment décrit :

```

CRC_REG=0 ;
REPEAT
  CRC_NXT_BIT=(NXT_BIT) XOR (CRC_REG(14)) ;
  CRC_REG(14:1)=CRC_REG(13:0) ;
  CRC_REG(0)=0 ;
  IF CRC_NXT_BIT THEN
    CRC_REG(14:0)=CRC_REG(14:0) XOR (4599hex) ;
  ENDIF
UNTIL(CRC SEQUENCE starts or there is an ERROR condition)
  
```

5.5. Champ d'acquittement

Le champ d'acquittement possède 2 bits (figure 7). La station émettrice de la trame laisse le bus libre pendant 2 coups d'horloge (ce qui correspond à l'émission de deux bits récessifs) et elle passe en mode réception pendant le premier coup d'horloge.

Le premier bit correspond à l'acquittement par l'ensemble des nœuds ayant reçu le message. Si aucune erreur n'a été détectée par un nœud (après calcul du CRC), ce dernier émet un bit dominant sinon il émet une trame d'erreur. La station émettrice du message original doit alors être capable de réagir en fonction de l'émission d'un bit dominant ou non par les autres stations sur le premier bit du champ d'acquittement.

Le second bit est un bit délimiteur d'acquittement qui doit toujours être récessif.

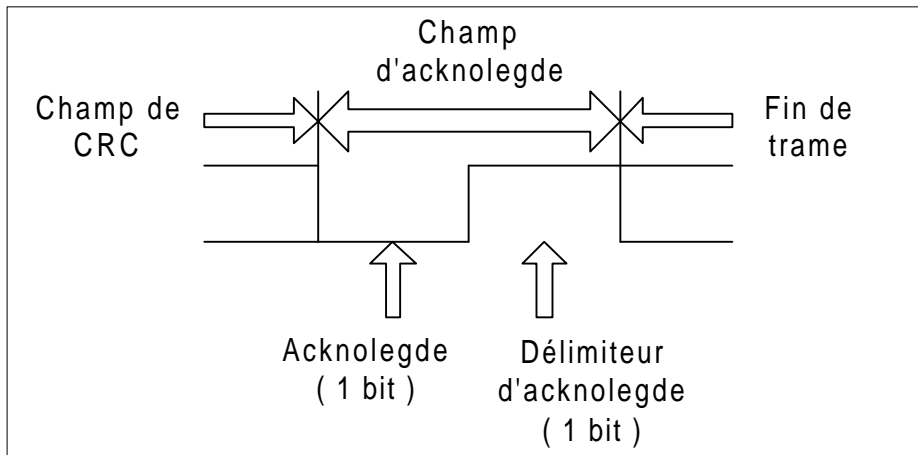


Figure 7 : Champ acquittement

5.6. Champ de fin de trame

Ce champ de fin de trame est constitué de 7 bits récessifs, ce qui déroge à la règle de *Bit-Stuffing*. Ce champ étant fixe, il est nécessaire de désactiver le codage (à l'émission) et le décodage (à la réception) suivant la règle du *Bit-Stuffing*.

6. TRAME DE REQUETE

Une trame de requête est constituée de la même manière qu'une trame de données sauf que le champ de données est vide (figure 8).

Dans le champ d'arbitrage, le bit de RTR est récessif. Par conséquent si deux nœuds émettent chacun une trame possédant le même identificateur (c'est à dire qu'un nœud émet une trame de données et l'autre une trame de requête), l'arbitrage sur le bit de RTR va donner la priorité à la trame de données.

Si un nœud a besoin d'un certain nombre de données, il va émettre une trame de requête dès que le bus sera libre en prenant soin d'indiquer dans le champ de contrôle le nombre d'octets de données dont il a besoin.

Les règles de construction des autres divers champs d'une trame de requête sont les mêmes que dans le cas d'une trame de données.

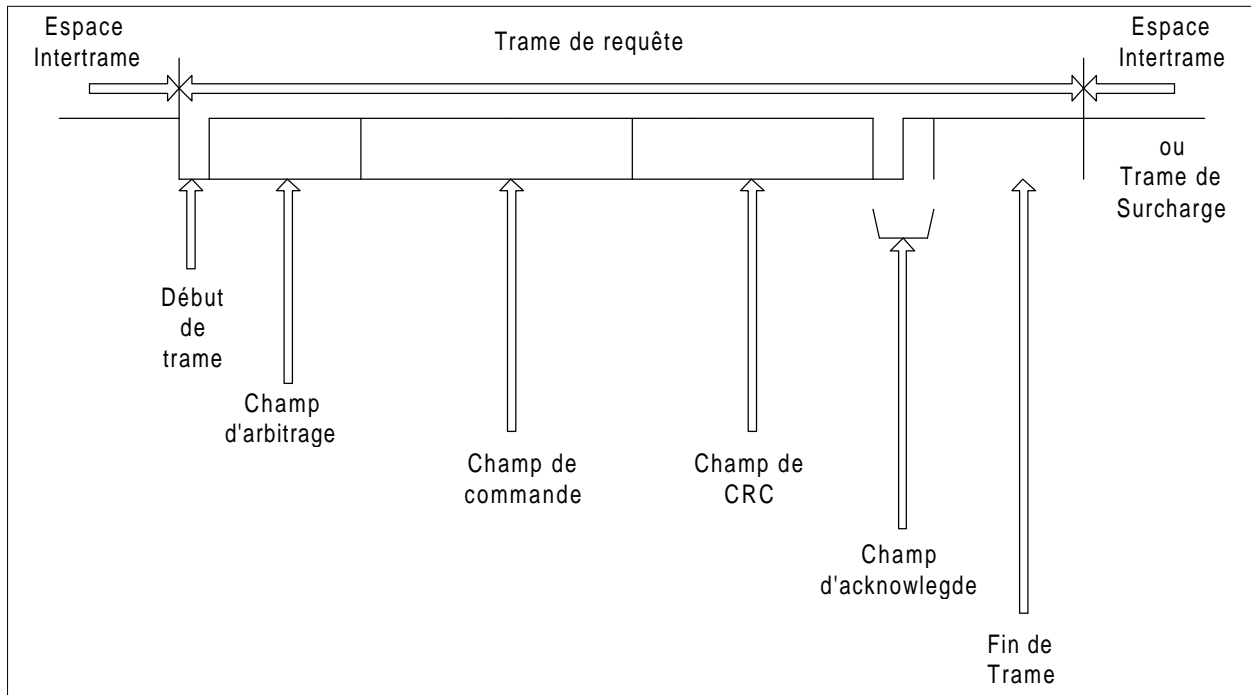


Figure 8 : Trame de requête

7. TRAITEMENT DES ERREURS

Lors de l'émission d'une trame sur le bus, des erreurs de transmission peuvent venir perturber le bon fonctionnement des différents utilisateurs du bus. L'erreur peut venir d'un nœud, et empêcher le réseau de fonctionner correctement. Pour cela, des méthodes de détection d'erreurs de transmissions sont prévues par le protocole CAN.

7.1. Les différents types d'erreurs

- **Le Bit Error :**

Chaque fois qu'un émetteur envoie un bit sur le bus, il vérifie en même temps si le niveau émis sur le bus correspond à celui qu'il désire envoyer en faisant une surveillance du bus. Si le niveau ne correspond pas, il le signale par un *Bit Error*.

Cependant, le *Bit Error* n'est pas signalé dans les cas suivants :

- Aucune erreur de *Bit Error* n'est signalée lorsqu'un bit dominant est envoyé dans le champ d'arbitrage à la place d'un bit récessif. Le bit dominant signifie simplement une perte d'arbitrage.
- De même, pour un bit dominant lors de l'*acknowledge slot*, à la place d'un bit récessif.
- Un émetteur envoyant un *flag* d'erreur passive (bit récessif) et recevant un bit dominant, ne doit pas signaler un *Bit Error*.

- **L'erreur de Stuffing (Stuff Error) :**

Une erreur de *Stuffing* est détectée à chaque fois qu'il y a 6 bits ou plus consécutifs de même signe sur le bus.

Cependant, une erreur de *Stuffing* ne doit être signalée que dans les champs d'identificateurs, de commande et de données. La règle du *Bit-Stuffing* ne s'appliquant plus après la fin du CRC. En aucun cas, une erreur de *Bit-Stuffing* ne doit être signalée dans le champ de fin de trame ou dans le champ d'acquiescement.

- **L'erreur de Cyclic Redundancy Code (CRC Error) :**

Si la valeur du CRC calculée par le récepteur est différente de celle envoyée par l'émetteur, il y a erreur de CRC (*CRC Error*).

- **L'erreur d'Acknowledge Delimiter :**

Une erreur d'*Acknowledge Delimiter* est signalée lorsque le récepteur n'observe pas un bit récessif lors du champ de *Acknowledge Delimiter*. Il en est de même pour le *CRC Delimiter*.

- **L'erreur de Slot Acknowledge (Acknowledgment Error) :**

Une erreur de *Slot Acknowledge* est signalée par l'émetteur s'il ne lit pas un bit dominant lors du champ de slot acknowledge.

La figure 9 résume les différents types d'erreurs et leur validité suivant l'endroit où l'on se trouve dans la trame.

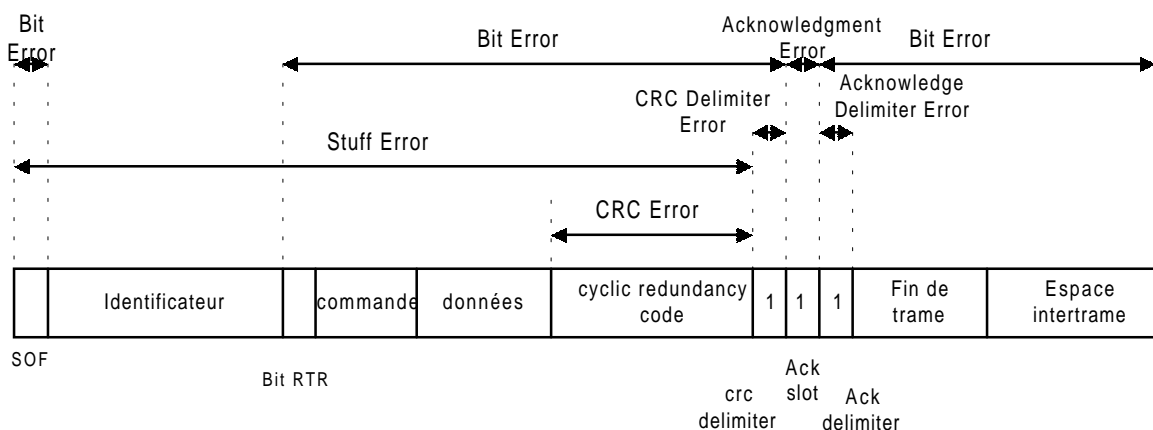


Figure 9 : Les sources d' erreur dans la trame CAN

7.2. Les trames d'erreurs

- **La trame d'erreur :**

La trame d'erreur est constituée de deux champs principaux :

- le drapeau d'erreur,
- le délimiteur de champ.

La figure 10 montre de quelle manière est construite la trame d'erreur.

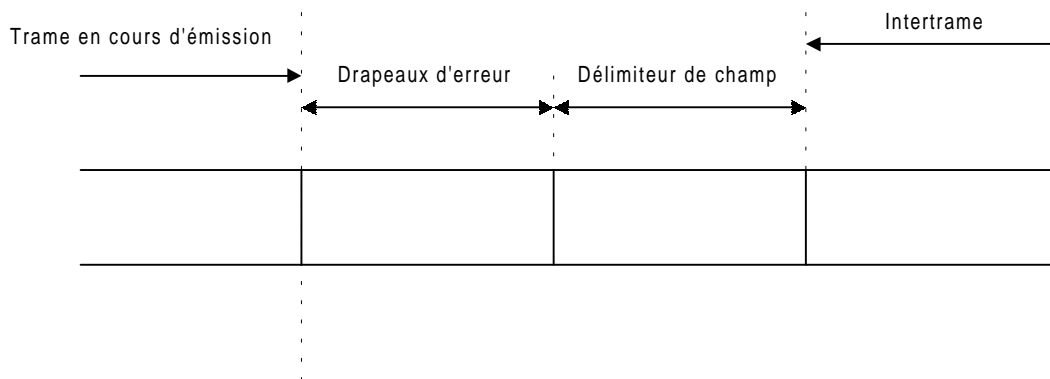


Figure 10 : Construction de la trame d' erreur

Le champ des drapeaux peut être constitué de deux sortes de drapeaux :

- les drapeaux d'erreur active (*Active Error Flag*),
- les drapeaux d'erreur passive (*Passive Error Flag*).

Les trames diffèrent suivant le type de drapeaux qu'elles contiennent. Les figures 11 et 12 représentent les deux types de trame avec leurs drapeaux respectifs.

Trame d'erreur active

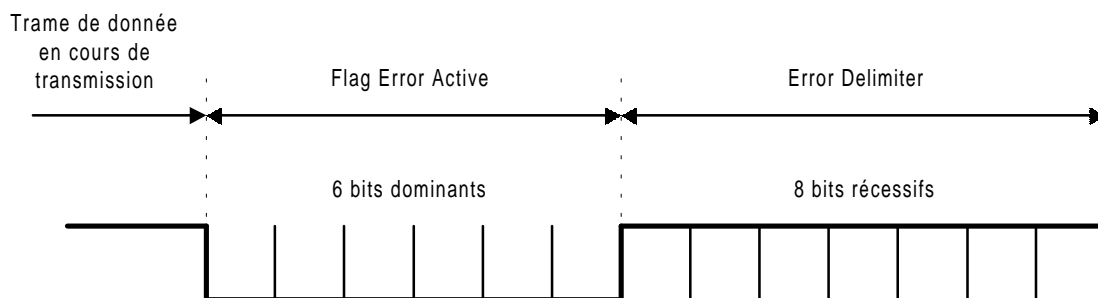


Figure 11 : Trame d' erreur active

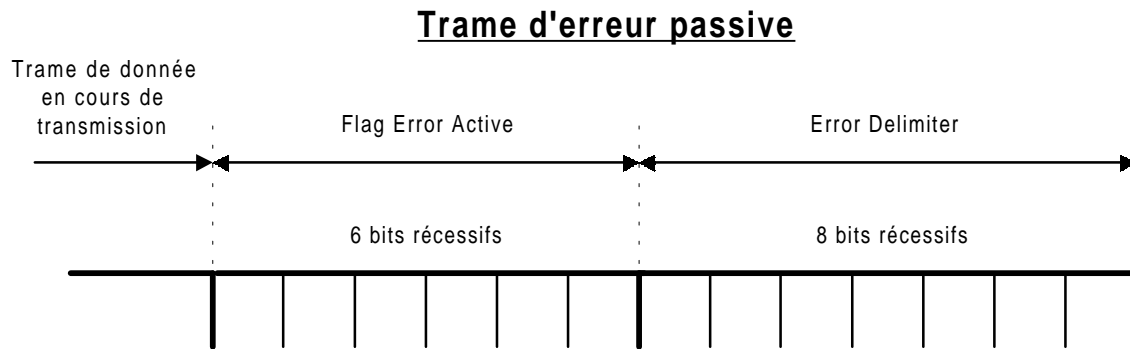


Figure 12 : Trame d'erreur passive

- **La trame d'erreur active :**

Elle est formée de six bits dominant consécutifs pour le champ de drapeau suivi de huit bits récessifs pour le délimiteur. Par construction, la trame d'erreur brise la règle du *Bit-Stuffing*. Les autres récepteurs vont donc se mettre à émettre des trames d'erreurs actives (s'ils sont en mode d'erreur active) à la fin du drapeau de la première station qui a émis la trame d'erreur. Toutes les stations vont donc émettre à tour de rôle la trame d'erreur. La dernière station aura en charge d'émettre le champ *Error Delimiter*, les autres champs ayant été remplacés par les bits dominants des drapeaux émis.

Remarque :

La norme limite le nombre de bits dominant consécutifs à 12 bits.

- **La trame d'erreur passive :**

La trame est formée de six bits récessifs pour le drapeau et de huit bits récessifs pour le délimiteur. Le champ du drapeau brise de nouveau la règle du *Bit-Stuffing* et les émetteurs envoient à tour de rôle le *Passive Error Flag* (s'ils sont en mode d'erreur passive). Mais une trame d'*Active Error Flag* reste prioritaire sur une trame de *Passive Error Flag* si elles sont envoyées en même temps. En effet, Les bits dominants de l'*Active Error Flag* remplacent les bits récessifs du *Passive Error Flag*. La fin de la trame quant à elle ne change pas puisqu'elle est formée dans les deux cas de huit bits récessifs.

7.3. Recouvrement des erreurs

Le recouvrement des erreurs est assuré par la retransmission automatique de la trame incriminée jusqu'à ce que l'émission de cette trame s'effectue sans erreur. La validité du message est acquise s'il n'y a aucune erreur depuis le SOF (*Start Of Frame*) jusqu'à la fin de trame.

Si l'émetteur n'arrive pas à émettre sa trame correctement, il essaye de nouveau de l'émettre jusqu'à ce que son compteur d'erreur passe en mode d'erreur passive.

La gestion des modes d'erreur :

Suivant le nombre d'erreur qu'un nœud comptabilise, l'état du mode de ce nœud peut différer. Un compteur mémorise le nombre d'erreur rencontré lors de la transmission des trames sur le bus. Deux compteurs séparés régissent respectivement le nombre d'erreurs en émission et en réception. Il se nomme :

- *Transmit Error Counter* pour l'émission,
- *Receive Error Counter* pour la réception.

Lorsque le nombre d'erreur devient trop important et que le gestionnaire est déjà en erreur passive, le nœud se met en *Bus Off* et se déconnecte du bus. Il ne reçoit ni émet à ce moment là aucune trame circulant sur le bus CAN.

Le passage dans les différents modes s'effectue suivant la valeur des compteurs comme le montre la figure 13.

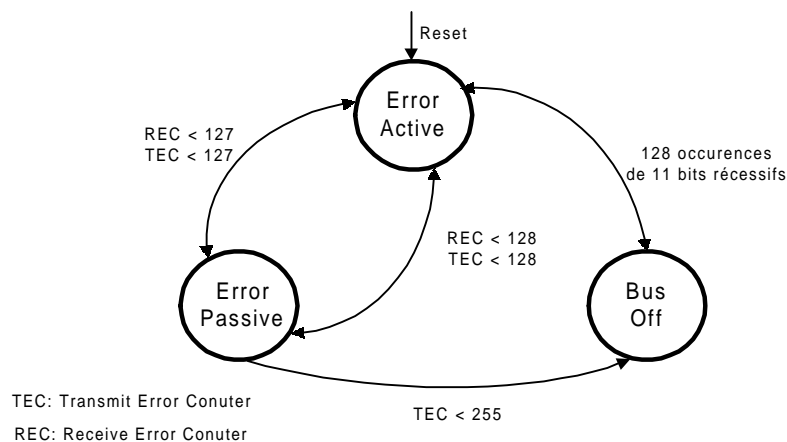


Figure 13 : Compteur d'erreur et état d'un nœud

Les règles de passages dans les modes :

L'incrémement et la décrémentation des compteurs ne se fait pas avec le même rapport. En effet, le compteur s'incrémement plus vite lorsqu'il y a une erreur qu'il ne se décrémente lorsque la trame reçue est correcte.

Les règles d'incrémement et de décrémentation des compteurs sont les suivantes :

- Le compteur de réception est incrémenté de 1 si :
 - un récepteur détecte une erreur, le compteur d'erreur de réception sera incrémenté de 1, sauf si l'erreur est un *Bit Error* durant un flag d'erreur active ou un flag de surcharge.
- Le compteur de réception est incrémenté de 8 si :
 - un récepteur reçoit un bit dominant juste après un flag d'erreur.
 - un récepteur voit un *Bit Error* tandis qu'il reçoit un flag d'erreur active ou un flag de surcharge.

- Le compteur de transmission est incrémenté de 8 si :
 - un émetteur envoie un flag d'erreur, son compteur d'émission s'incrémente de 8 sauf si l'émetteur est en erreur passive et voit un *Acknowledgment Error* (il ne détecte aucun bit dominant sur le champ d'acknowledge ou lors de l'envoi de la trame d'erreur passive) et également si l'émetteur envoie un flag d'erreur lors d'une erreur de *Bit-Stuffing* durant la période d'arbitrage (détection d'un bit dominant au lieu d'un bit récessif situé après le RTR).
 - un émetteur voit un *Bit Error* tandis qu'il émet un flag d'erreur active ou un flag de surcharge.

- Les compteurs d'émission et de réception s'incrémentent de 8 si :
 - chaque nœud recevant sept bits dominants consécutifs après réception d'un *Active Error Flag*, d'un *Passive Error Flag* ou d'un *Overload Flag*. Après détection de quatorzième bit dominant consécutif (pour l'*Active Error Flag* ou l'*Overload Flag*) ou du huitième bit dominant consécutif suivant le *Passive Error Flag*, et après toutes les suites de huit bits dominants consécutifs.

- Le compteur de réception est décrémenté de 1 si :
 - le récepteur reçoit une trame sans erreur (jusqu'au champ d'*Acknowledge Slot*) et si la valeur du compteur est comprise entre 1 et 127. Si le compteur est à 0, sa valeur ne change pas (pas d'incrémementation). S'il est supérieur à 127, sa valeur est ramenée entre 119 et 127.

- Le compteur d'émission est décrémenté de 1 si :
 - la transmission d'une trame se déroule sans erreur (jusqu'au champ d'*Acknowledge Slot*). Si la valeur du compteur est à 0, le compteur ne s'incrémente pas.

Les modes d'erreurs :

- **Mode d'erreur active :**

Le gestionnaire de protocole est en mode d'erreur active si le compteur de réception et le compteur d'émission ont une valeur inférieure à 127. Dans ce mode, le nœud émet des trames d'erreurs actives (*Active Error Flag*).

- **Mode d'erreur passive :**

Le gestionnaire de protocole est en mode d'erreur passive si le compteur de réception ou le compteur d'émission est supérieur ou égal à 128 et inférieur à 255. Dans ce mode, le nœud émet des trames d'erreurs passives (*Passive Error Flag*).

- **Mode Bus Off :**

Le gestionnaire de protocole est en mode *Bus Off* si la valeur d'un des deux compteurs est > 255. Le nœud est alors totalement déconnecté du bus (les drivers de lignes ne sont plus actifs). Il sort de cet état de *Bus Off* avoir reçu 127 trames de onze bits récessifs.

8. FIN DE TRAMES CAN

8.1. Trame de surcharge

La trame de surcharge indique aux autres nœuds qu'une station est surchargée. Elle est formée de deux champs :

- le drapeau de surcharge (*Overload Frame*) avec six bits dominants,
- le délimiteur de surcharge (*Overload Delimiter*) avec huit bits récessifs.

La figure 14 représente la trame.

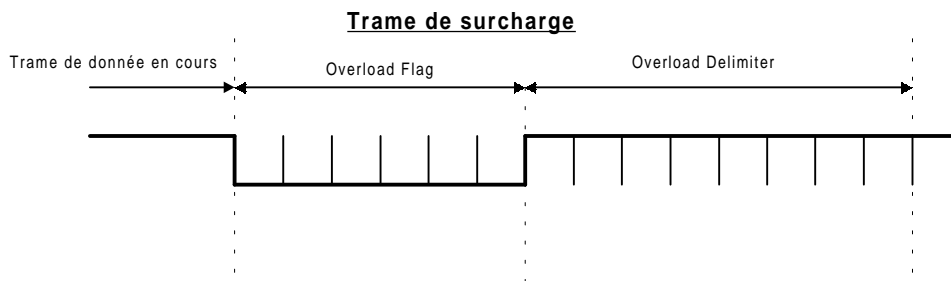


Figure 14 : Trame de surcharge

Une trame de surcharge est émise sur le bus si :

- un bit dominant est détecté durant la période d'intertrame.
- un récepteur n'est pas prêt pour la réception d'une nouvelle trame de donnée ou de requête (retard sur le traitement des informations circulant sur le bus).

Dès qu'une trame de surcharge est émise, les autres nœuds voient sur le bus une suite de six bits dominants qui ne respectent pas la règle du *Bit-Stuffing*. Ils émettent à leur tour une trame de surcharge. Seulement deux trames de surcharges consécutives sont autorisées sur le bus (pas plus de 12 bits dominants consécutifs émis sur le bus).

8.2. Période d'intertrame

Elle sépare les trames de données ou de requêtes entre elles. Il s'agit d'une suite de plusieurs bits récessifs.

- **Le champ d'intermission :**

Le champ d'intermission est une suite de 3 bits récessifs consécutifs. Durant la période d'intermission, l'émission de trame n'est pas autorisée. Les gestionnaires de protocole ne sont autorisés à signaler que les conditions de surcharge.

- **Le champ de *Bus Idle* :**

Le champ de *Bus Idle* est celui du bus quand il est au repos. Le niveau de repos est le niveau récessif et aucune trame ne circule sur le bus.

- **Le champ de suspension de transmission :**

Le champ de suspension de transmission est émis par un nœud lorsque celui-ci envoie une trame d'erreur passive.

La figure 15 représente les différents champs.

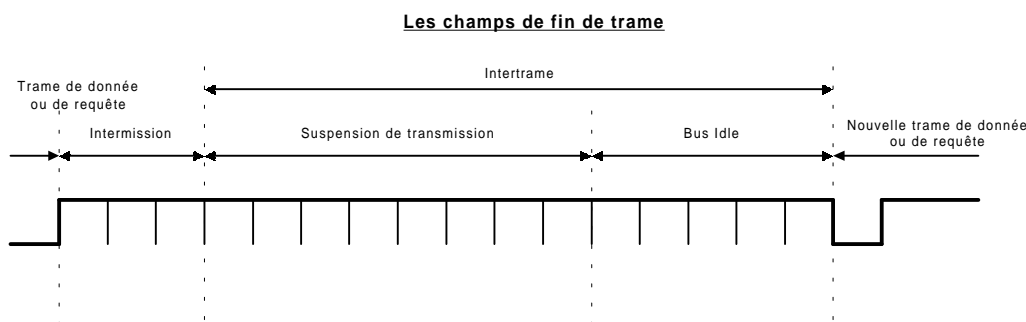


Figure 15 : Période d'intertrame

8.3. Autres modes

Pour la gestion de l'énergie sur le bus, les drivers de ligne peuvent être désactivés lorsqu'il n'y a plus de trames sur le bus.

Pour activer ces drivers sur le bus, la station devra observer 11 bits récessifs à la suite. La procédure ainsi décrite est la procédure de réveil appelée *Wake-up*. Un identificateur a été réservé à cette fonction pour éviter de perdre un trop grand nombre de trames lors de la reconnexion sur le bus.

Lors des démarrages d'une station sur le bus, le *Start-up* se charge de connecter les drivers de lignes et d'observer la séquence voulue pour commencer à émettre ou à recevoir des trames du bus.

9. CODAGE DE LIGNE

Dans le protocole CAN le code de ligne (en bande de base) choisi pour la transmission des données sur le bus est le code NRZ (*Non Return to Zero*). La figure 16 donne un exemple de codage.

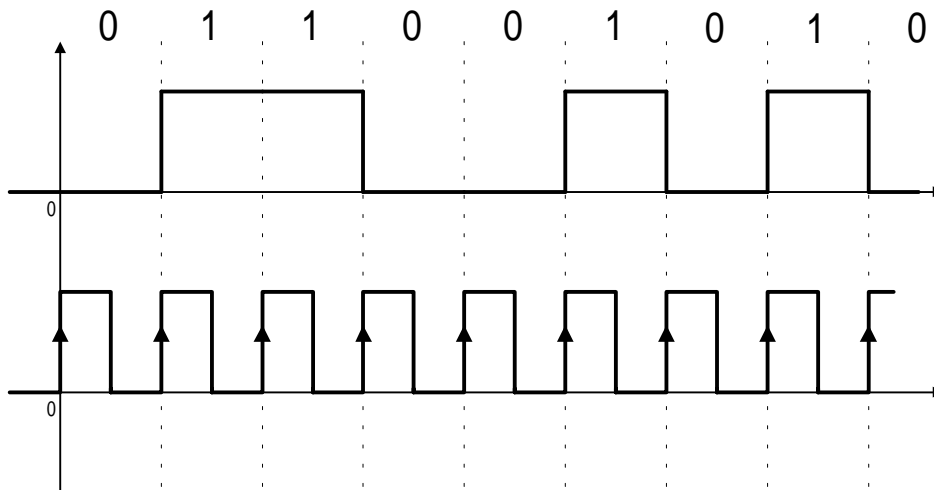


Figure 16 : Codage NRZ du bus CAN

Remarque :

La règle du *Bit-Stuffing* ne modifie en aucun cas le procédé de codage de ligne.

Comme le montre la figure 16, les transitions des bits s'effectuent sur chaque front montant de l'horloge. Dans le protocole CAN, une période d'horloge correspond à ce que l'on appelle le *Nominal Bit Time*.

10. LE NOMINAL BIT TIME

Le *Nominal Bit Time* représente en fait la durée du bit sur le bus. Cette durée est, comme nous l'avons vu, étroitement liée à la période de l'horloge. Chaque station reliée sur le bus doit être cadencée avec le même *Nominal Bit Time* pour pouvoir émettre et recevoir correctement les données circulant sur le bus.

Ainsi, la durée du bit time de chaque circuit est construite à partir d'un nombre déterminé de périodes d'horloge issue de l'horloge interne de chaque circuit CAN.

La norme BOSCH décrit avec précision la composition de ce *Nominal Bit Time* qui est divisé en plusieurs segments :

- le segment de synchronisation (SYNC_SEG),
- le segment de propagation (PROP_SEG),
 - le segment de phase buffer n°1 (PHASE_SEG1),
 - le segment de phase buffer n°2 (PHASE_SEG2).

La figure 17 donne un aperçu de ces divers composants et de leur agencement.

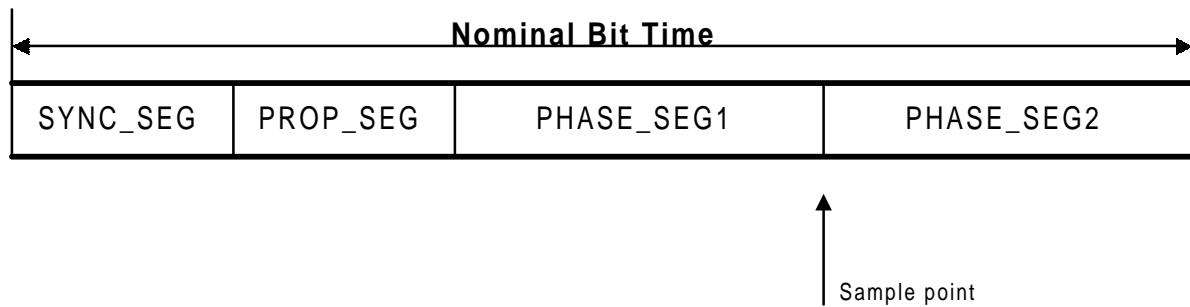


Figure 17 : Décomposition du nominal Bit Time

En fait, le *Nominal Bit Time*, exprimé en s, correspond évidemment à l'inverse du débit sur le bus. Nous avons donc la formule suivante:

$$No\ min\ al_Bit_Time = \frac{1}{No\ min\ al_Bit_Rate}$$

10.1. Description des différents segments

- **le segment de synchronisation** : le segment de synchronisation est utilisé pour synchroniser les différents nœuds du bus. Comme nous le verrons par la suite, une transition (de "0" à "1" ou de "1" à "0") doit s'effectuer dans ce segment pour permettre une resynchronisation des horloges des différents nœuds en mode de réception de trames.
- **Le segment de propagation** : le segment de propagation est utilisé pour compenser les phénomènes de temps de propagation sur le bus. Par définition :

$$Durée_PROP_SEG = 2 * (t_propag_bus + t_retard_comparateurs + t_retard_drivers_lignes)$$

- **Les segments "buffer phase1" et "buffer phase2"** : les segments "buffer phase1" et "buffer phase2" sont surtout utilisés pour compenser les erreurs de phase détectées lors des transitions. Nous verrons aussi que ces segments peuvent être plus courts ou plus longs à cause des phénomènes de resynchronisation.
- **Le point d'échantillonnage ou sample point** : le point d'échantillonnage ou sample point est le point où la valeur du bit est lue sur le bus. Il est situé à la fin du segment de "buffer phase1" et constitue la seule valeur mémorisée pour le niveau du bit. On s'affranchit des phénomènes de propagation et d'oscillation des données sur le bus dans les segments précédents.

10.2. Durée des différents segments et notion de *Time Quantum*

Le *Time Quantum* : Le *Time Quantum* est une unité de temps qui est construite à partir de la période de l'oscillateur interne de chaque nœud. Les fréquences de fonctionnement du bus CAN s'étendant de 125 KHz à 1 MHz et celle des oscillateurs étant de plusieurs MHz, le *Time Quantum* représente plusieurs périodes d'une horloge d'oscillateur. La période d'horloge de l'oscillateur est appelée *minimum Time Quantum*. La valeur du préscalaire *m* détermine le rapport entre le *Time Quantum* et le *Minimum Time Quantum* :

$$\underline{TIME_QUANTUM = m * MINIMUM_TIME_QUANTUM .}$$

La valeur de *m* peut varier de 1 à 32. La figure 18 représente la construction d'un *Time Quantum* à partir d'une période d'horloge interne au circuit.

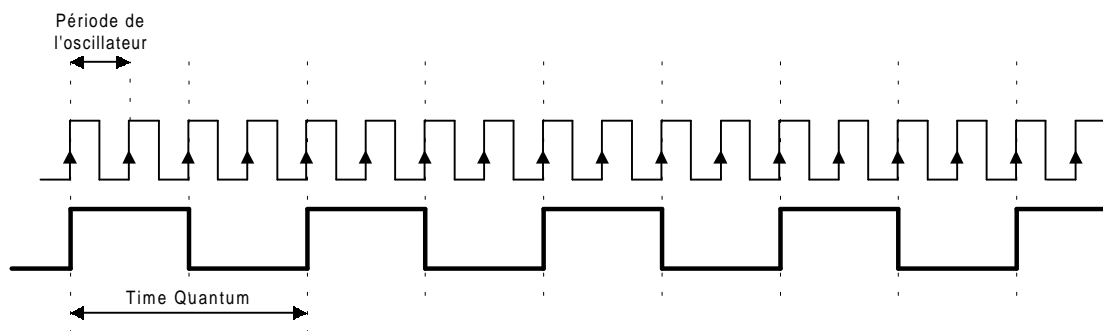


Figure 18 : Construction du Time Quantum

Dans l'exemple ci-dessus, le facteur *m* est égal à 4.

- **La durée des différents segments :**

Segment	Durée en Time Quanta
Synchronisation - SYNC_SEG	1
Propagation - PROP_SEG	1 à 8
Buffer phase1 - PHASE_SEG1	1 à 8
Buffer phase1 - PHASE_SEG2	1 à 8

Le nombre de *Time Quanta* dans un *Nominal Bit Time* peut ainsi varier de 8 à 25. La figure 19 donne le nombre de *Time Quanta* possible par segment de *Nominal Bit Time*.

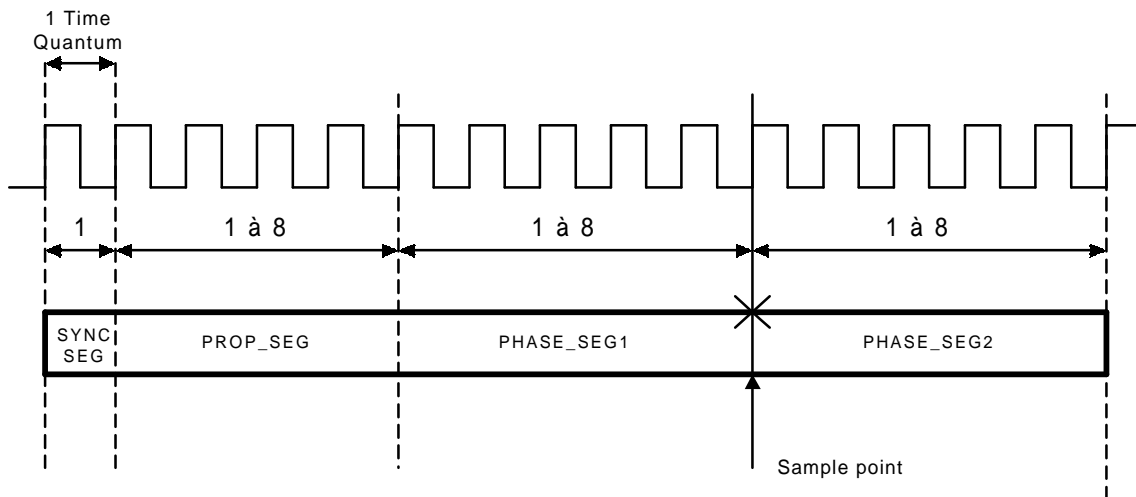


Figure 19 : Durée des différents segments

Le choix du nombre de *Time Quanta* pour chaque segment dépend de la fréquence de l'oscillateur. Un nombre important de *Time Quanta* par segment augmente la précision de la synchronisation des différents nœuds sur le bus.

11. SYNCHRONISATION DES HORLOGES

Chaque nœud doit produire un nominal *Bit Time* pour pouvoir recevoir et émettre les données circulant sur le bus en synchronisme avec les autres circuits. En effet, si les Nominal Bit Time de chaque nœud ne sont pas du tout synchronisés, la valeur lue sur le bus au moment de l'échantillonnage peut ne pas être la valeur correcte au bon moment, comme le représente la figure 20. Ces retards peuvent être gênants, dans la phase d'acquittement de la trame où il y a peu de temps pour finir de calculer le CRC et envoyer un bit à l'état dominant lors de l'*Acknowledge Slot* pour confirmer que la trame a bien été reçue.

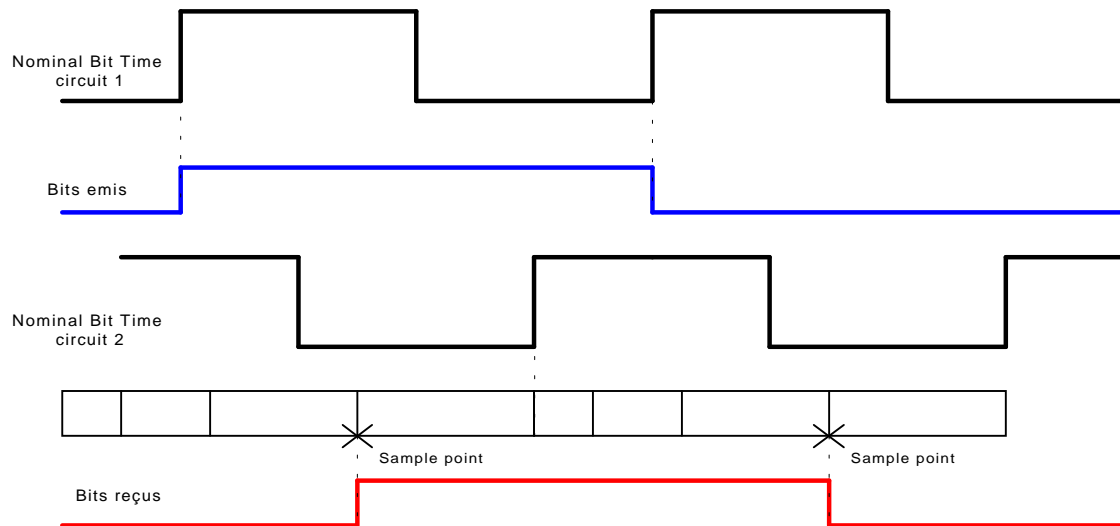


Figure 20 : Problème de la synchronisation

La norme BOSCH prévoit donc des règles de synchronisation du *Nominal Bit Time* de chaque circuit pour palier aux inconvénients exposés ci-dessus.

11.1. Notion de RJW

Pour corriger les dérives du *Nominal Bit Time*, il faut placer des butées que la dérive de la période ne pourra pas dépasser. La butée en question s'appelle le RJW : *Resynchronisation Jump Width*. Le RJW est une variable entière programmée à une valeur comprise entre 1 et le minimum de (4, segment de phase1).

$$1 < R_{jw} < \min(4, PHASE_SEG1).$$

La valeur est mise dans le registre du circuit lors de l'initialisation et ne change pas en cours de fonctionnement.

11.2. Notion d'erreur de phase

L'erreur de phase (PHASE_ERROR) est détectée lorsqu'une transition d'un bit dominant à récessif ou d'un bit récessif à dominant ne s'effectue pas à l'intérieur du segment de synchronisation. Une variable notée e sert à quantifier cette erreur de phase et fournit le signe. Le calcul de e est fait de la manière suivante :

- e = 0, si la transition s'effectue dans le segment de synchronisation (SYNC_SEG).
- e < 0, si la transition s'effectue avant le point d'échantillonnage (Sample Point).
- e > 0, si la transition s'effectue après le point d'échantillonnage (Sample Point).

La règle simple évoquée ci-dessus sert de base pour resynchroniser les *différents Nominal Bit Time* de chaque circuit connecté au bus. La règle s'appuie sur les transitions des bits récessifs à dominant ou dominant à récessif qui arrivent au moins tous les 5 bits de même signe consécutifs, à cause de la règle du *Bit-Stuffing*.

L'erreur de phase e est donc calculée par rapport au *Sample Point* qui détermine si le PHASE_SEG1 doit être allongé ou si le PHASE_SEG2 doit être raccourci pour que la prochaine transition s'effectue dans le SYNC_SEG. La figure 21 donne un exemple et les conséquences des emplacements des transitions sur la longueur des segments du *Nominal Bit Time*.

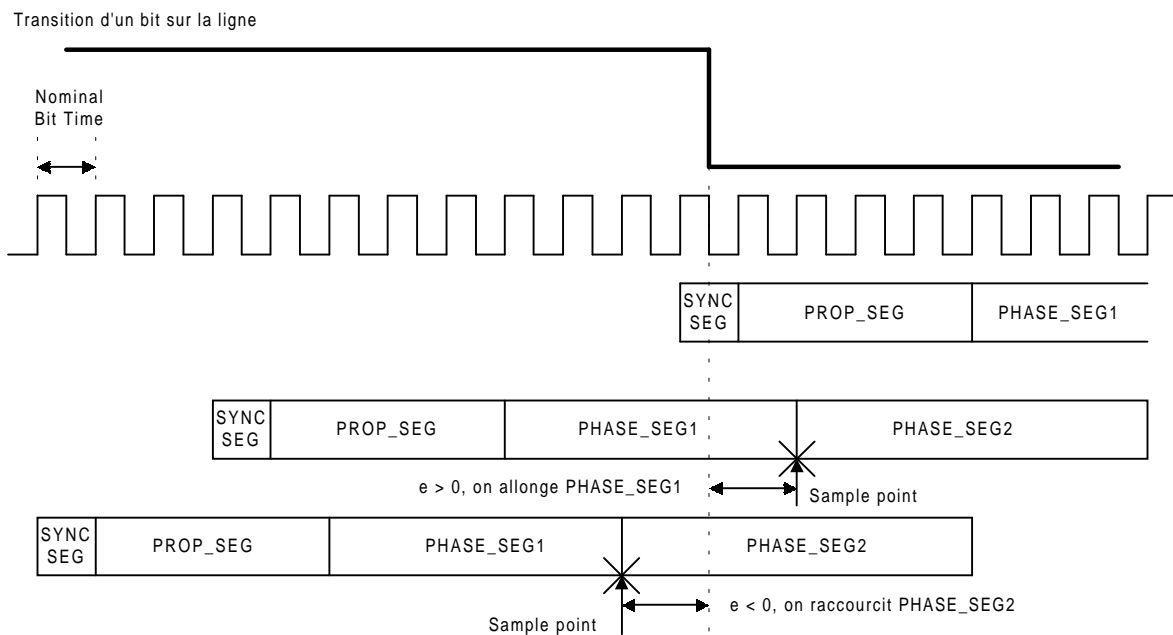


Figure 21 : Erreur de phase

11.3. Les règles de synchronisation

La hard-synchronisation :

L'effet d'une *hard-synchronisation* est de faire repartir le *Nominal Bit Time* depuis le segment de synchronisation (SYNC_SEG) à chaque fois qu'un ordre de *hard-synchronisation* est donné par le gestionnaire du protocole. Le *Nominal Bit Time* en cours est abandonné et un nouveau Nominal Bit Time repart dès le *Time Quantum* suivant, depuis le segment de synchronisation (SYNC_SEG).

La resynchronisation :

Le calcul et l'ordre de resynchronisation sont donnés à partir de la valeur de l'erreur de phase e , et dépendent aussi de la valeur du RJW :

- Si l'erreur de phase est nulle ($e = 0$, la transition est dans le SYNC_SEG), l'effet de la resynchronisation est le même que celui de la *hard-synchronisation*.

- Si l'erreur de phase est positive et inférieure en valeur absolue à RJW ($0 < e < RJW$), le PHASE_SEG1 sera rallongé de e.
- Si l'erreur de phase est négative, mais inférieure à RJW en valeur absolue ($e < 0$ et $|e| < RJW$) le PHASE_SEG2 est raccourci de e.
- Si l'erreur de phase est positive et supérieure ou égale RJW ($e > 0$ et $e > RJW$), le PHASE_SEG1 est rallongé de RJW.
- Enfin, si l'erreur de phase est négative et supérieure à RJW (en valeur absolue $-e < 0$ et $|e| > RJW$) le PHASE_SEG2 est raccourci de RJW.

Le tableau suivant résume les règles évoquées ci-dessus.

Erreur de phase	Effet sur PHASE_SEG1	Effet sur PHASE_SEG2
$0 < e < RJW$	Allongé de e	
$E < 0$ et $ e < RJW$		Raccourci de e
$e > 0$ et $e > RJW$	Allongé de RJW	
$E < 0$ et $ e > RJW$		Raccourci de RJW

Tableau 2 : Règles de resynchronisation

Les règles de synchronisation :

Un seul type de synchronisation est autorisé pour un même *Nominal Bit Time*.

• **Hard-synchronisation :**

- Une *hard-synchronisation* est faite à chaque fois qu'une transition s'effectue dans le segment de synchronisation SYNC_SEG.
- Une *hard-synchronisation* est effectuée lorsque le bus est au repos (*bus idle*) et qu'une transition d'un bit récessif à un bit dominant est détectée, autrement dit lors d'un SOF (*Start Of Frame*).

• **Resynchronisation :**

- Une resynchronisation est effectuée si une transition est détectée au point d'échantillonnage précédent et que la valeur lue sur le bus immédiatement après la transition est différente de celle lue sur le bus avant la transition.
- Les transitions des bits récessifs à dominants peuvent être utilisées pour la resynchronisation si elles respectent la règle précédente, sauf si un nœud émet un bit dominant qui ne suit pas la règle de resynchronisation avec une transition de récessif à dominant et une erreur de phase positive. La règle ne s'applique que si les transitions des bits de récessif à dominant sont utilisées pour la resynchronisation.

12. CARACTERISTIQUES PHYSIQUES DU BUS CAN

12.1. 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).

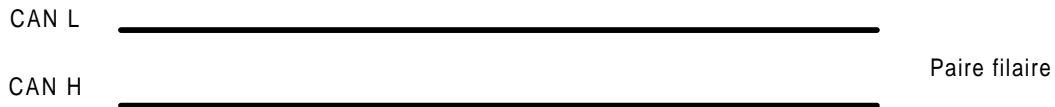


Figure 22 : Paire filaire du bus CAN

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.

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*.

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

Paramètres	<i>CAN low speed</i>	<i>CAN high speed</i>
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

Tableau 3 : Les 2 types de bus CAN

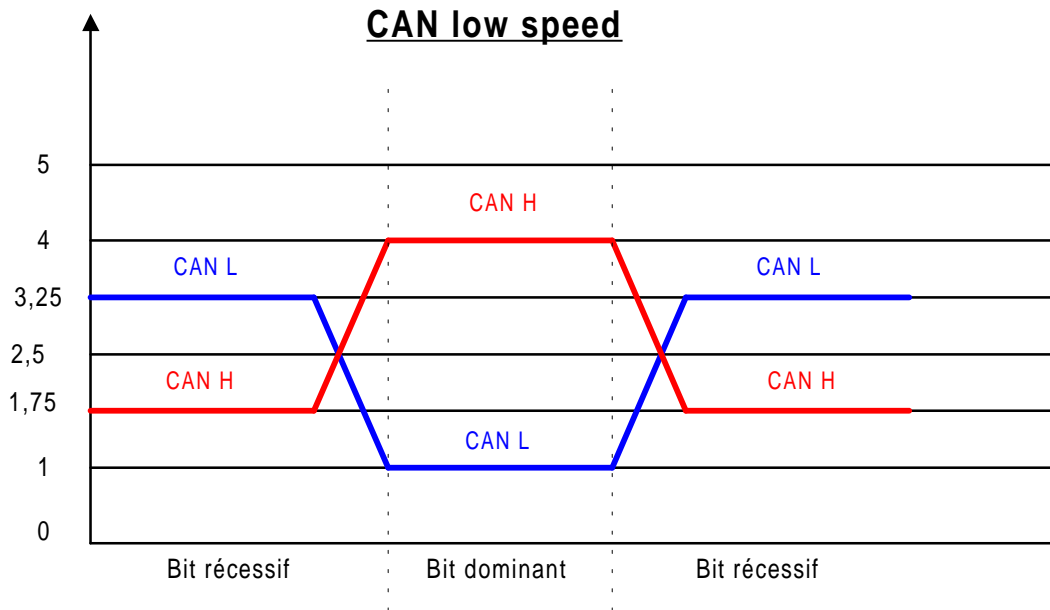


Figure 23 : Niveaux de tension du bus CAN *low speed*

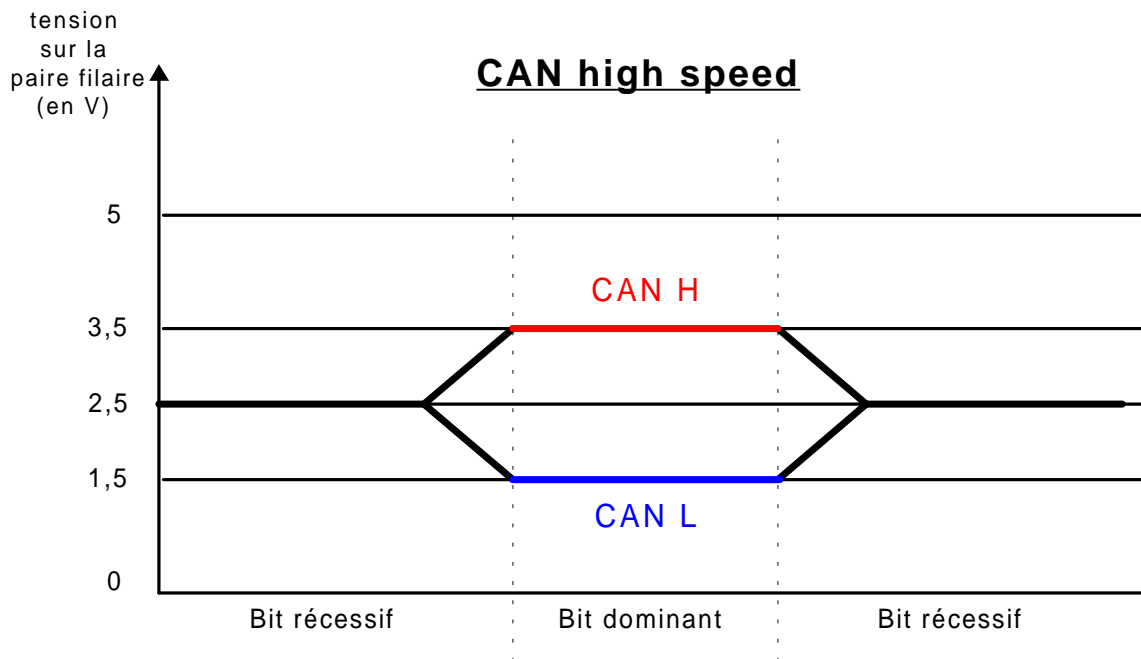


Figure 24 : Niveaux de tension du bus CAN *high speed*

Le schéma d'un circuit CAN relié au bus est présenté figure 25.

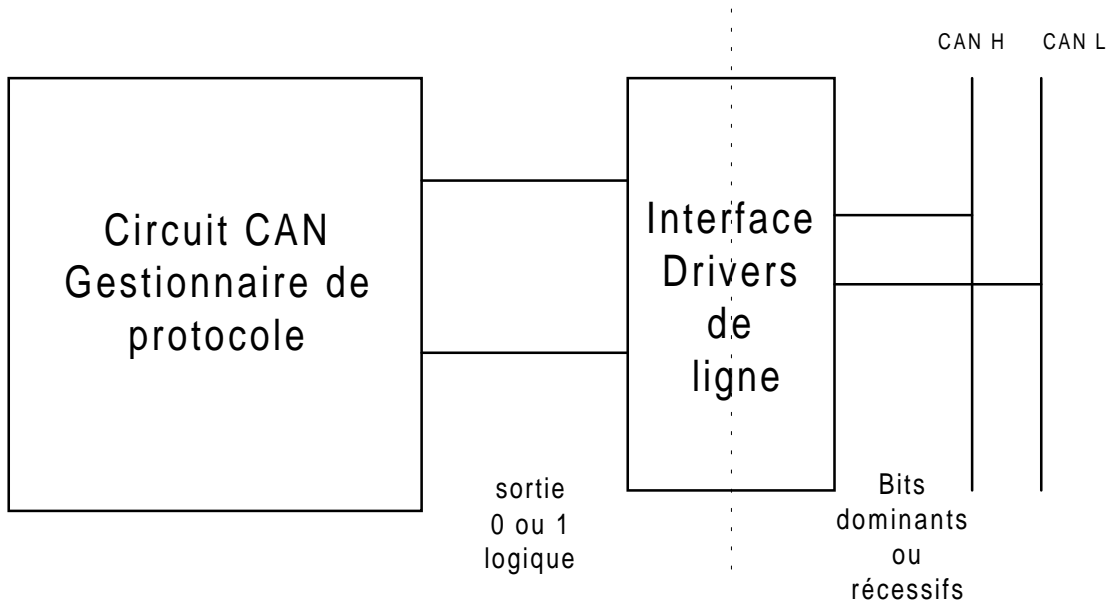


Figure 25 : Schéma de principe d'un circuit CAN

12.2. Débit sur le réseau et temps de latence

Le débit du réseau est calculé par rapport au *Nominal Bit Time*. Il s'étend de 125 kb/s à 1 Mb/s suivant le type de bus utilisé. Les valeurs ci-dessus correspondent au débit dit brut du réseau, c'est à dire en comptant tous les bits qui sont transmis sur le bus.

Le débit dit net ne tient compte que des bits transportant des informations utiles. Le débit net est en fait le débit utile du bus en ne comptant pas tous les bits tels que le SOF, les *Acknowledge Delimiter*, les bits de *Bit-Stuffing*...

Le *temps de latence* est un intervalle de temps qui représente la durée écoulée entre le moment où une demande de requête est formulée et l'instant où la réponse est présente sur le bus. Le temps de latence dépend du nombre de nœuds maîtres désirant effectuer un transfert de données.

13. BIBLIOGRAPHIE

- [1] Dominique PARET. Le bus CAN Controller Area Network. 1997
- [2] BOSCH. Norme CAN.
- [3] Pascal BORGHESI, Cyril RAPIN. Projet de fin d'étude ENSERB. 1998

14. WEBOGRAPHIE

<http://www.omegas.co.uk/CAN/index.html>

<http://www.can-cia.de/>