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.

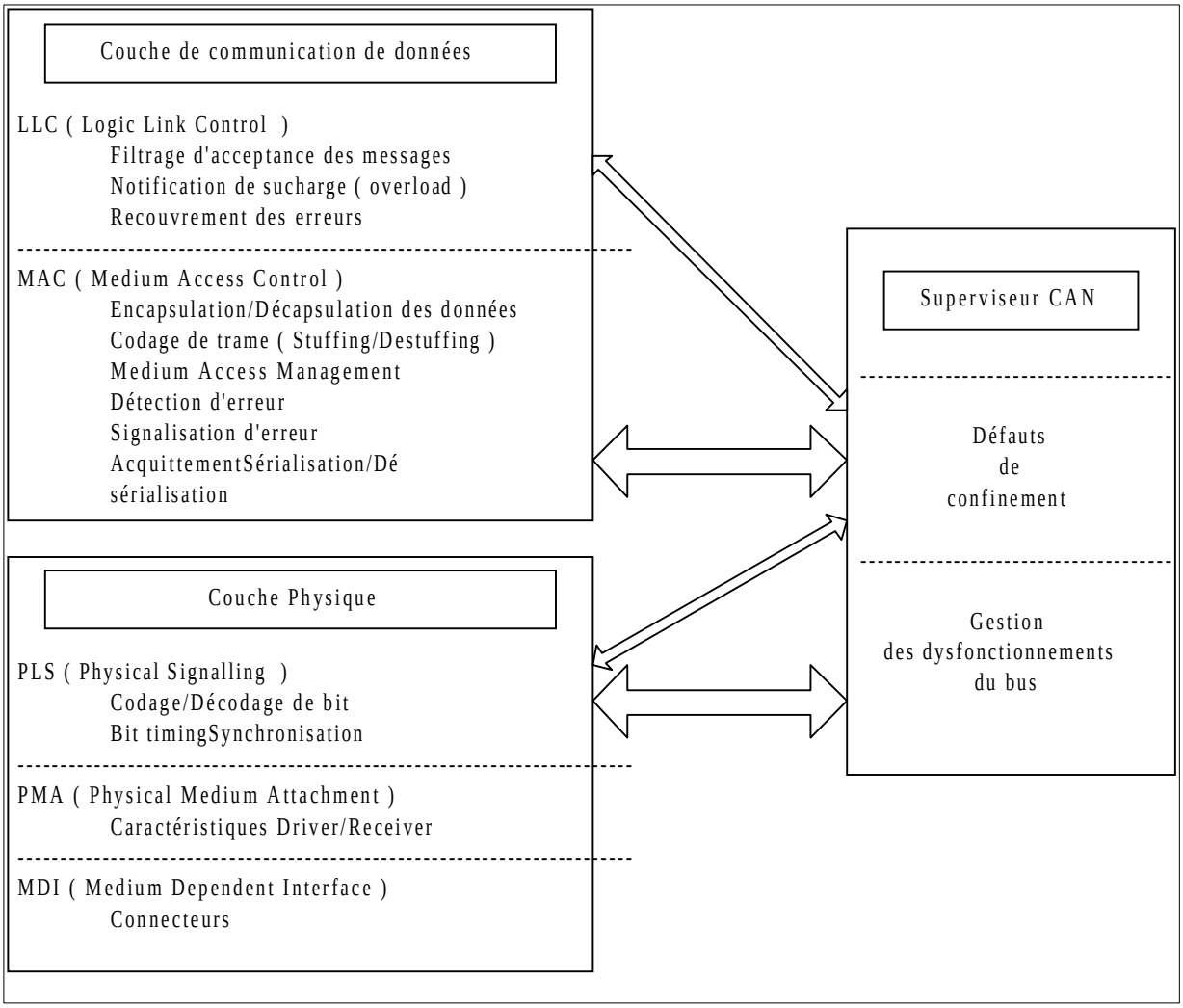

Figure 1 : Le protocole CAN et le modèle OSI

- 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 $4.7*10^{-11}$.
- *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 pat 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.

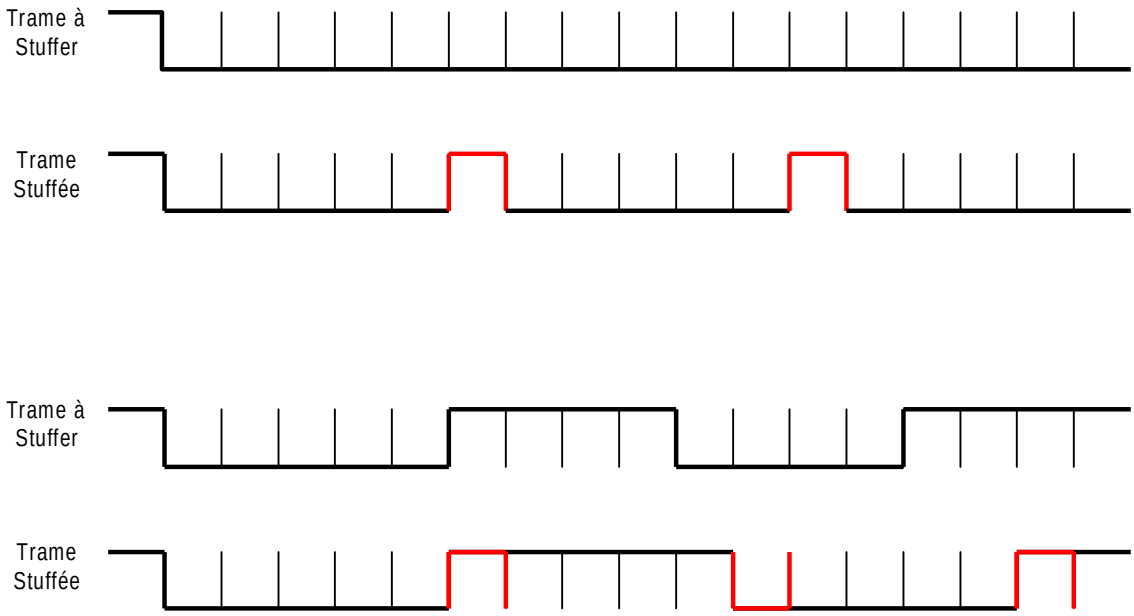

Figure 2 : Technique de *Bit-Stuffing*

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'acquittement (*Acknoledge*), 2 bits.
- le champ de fin de trame EOF (*End Of Frame*), 7 bits récessifs.

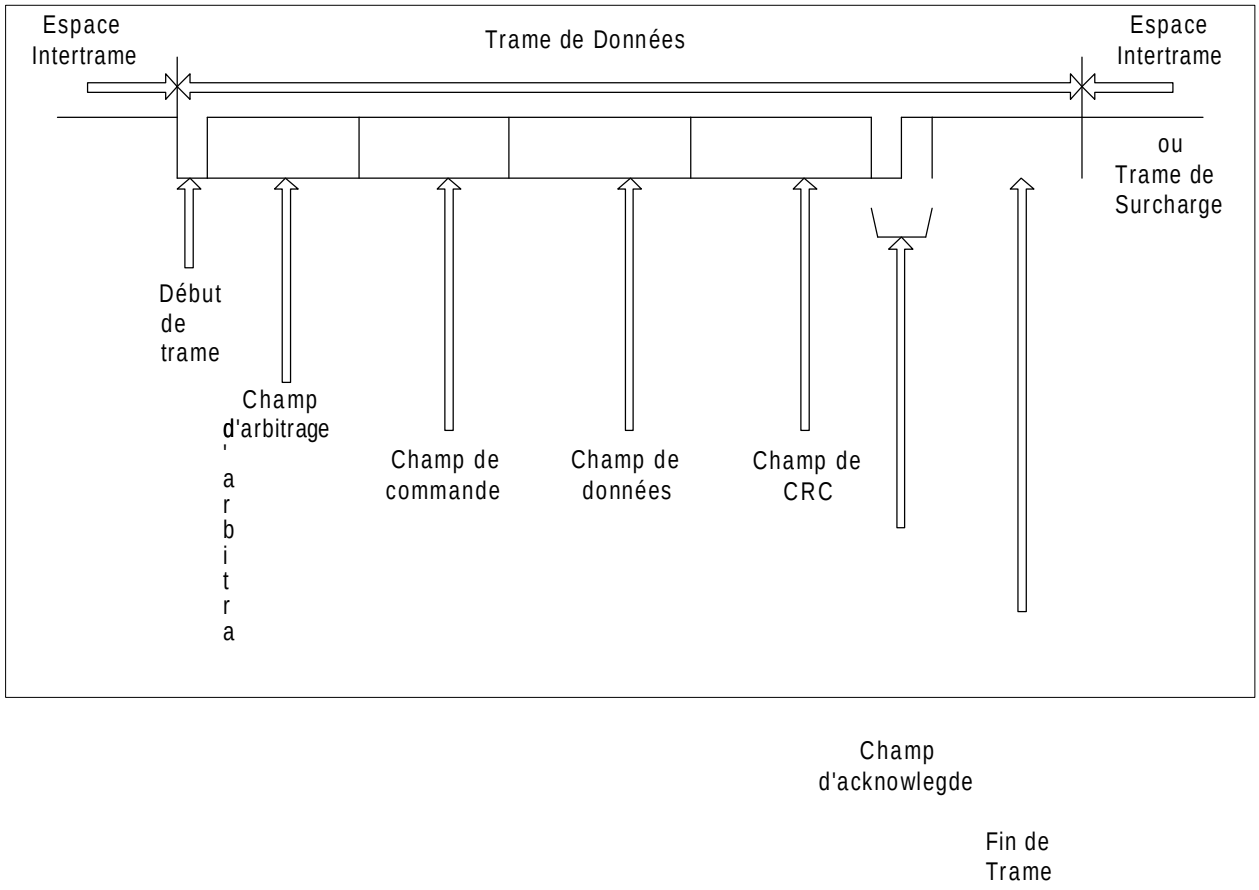

Figure 3 : Format de la trame CAN

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.

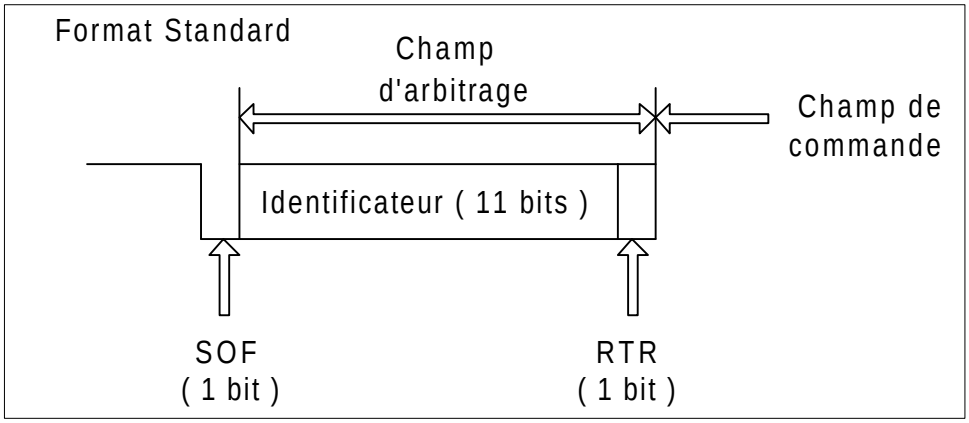

Figure 4 : Champ arbitrage

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 (*Data Length Code*)			
	DLC3	DLC2	DLC1	DLC0
0	D	D	D	D
1	D	D	D	R
2	D	D	R	D
3	D	D	R	R
4	D	R	D	D
5	D	R	D	R
6	D	R	R	D
7	D	R	R	R
8	R	D	D	D

D : bit Dominant, R : bit Récessif

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

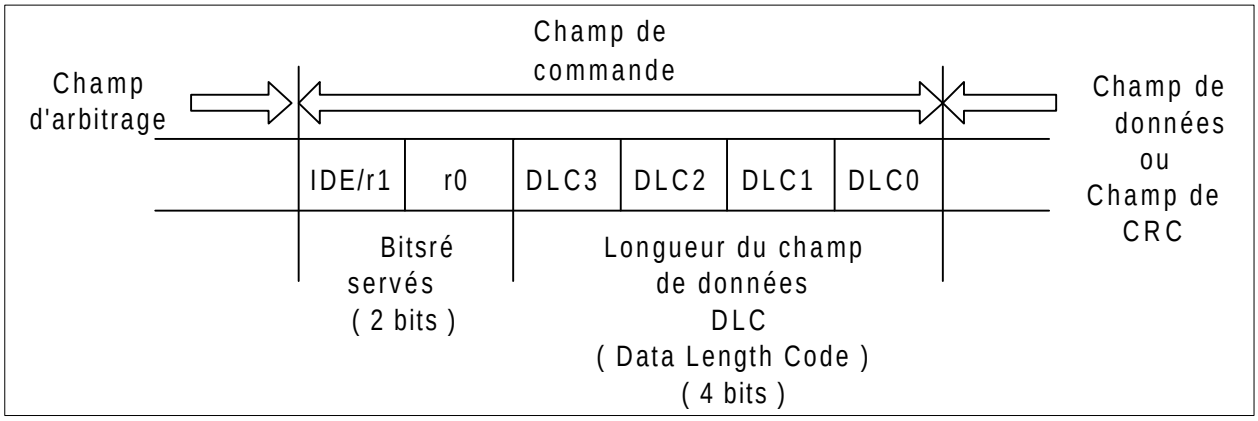

Figure 5 : Champ contrôle

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.

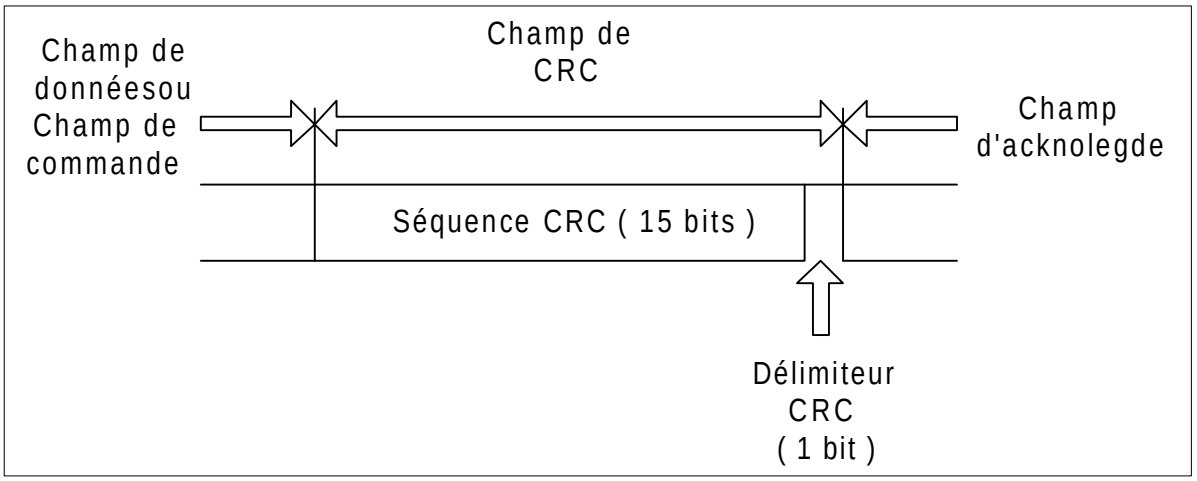

Figure 6 : Champ CRC

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 originel 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.

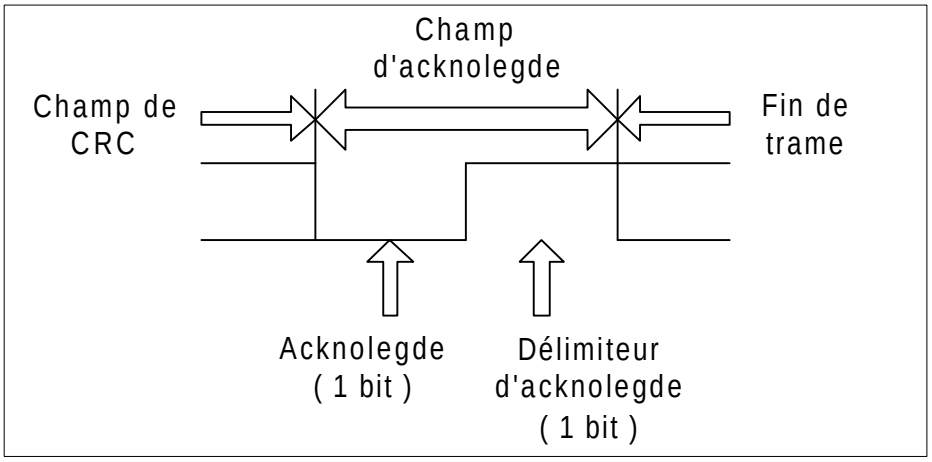

Figure 7 : Champ acquittement

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.

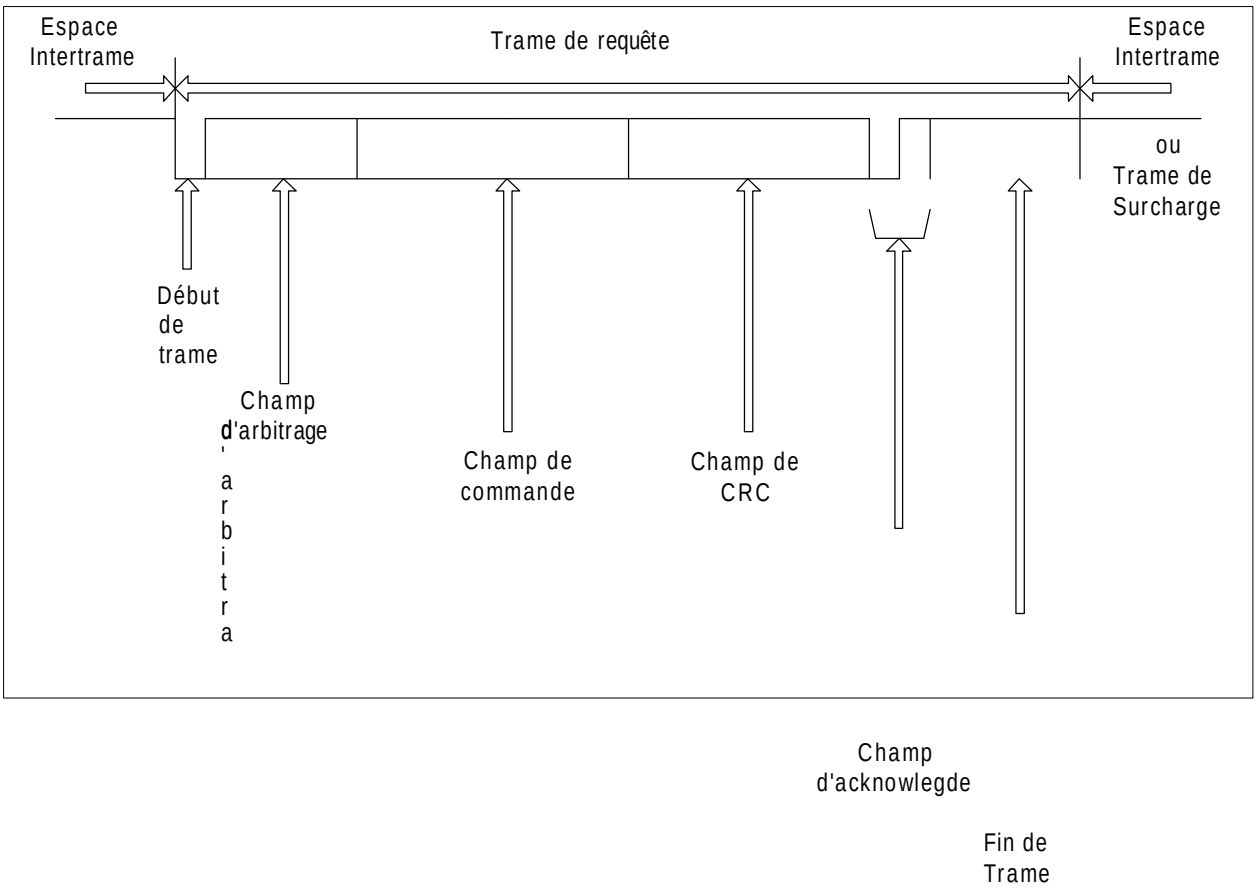

Figure 8 : Trame de requête

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 CRC. 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'acquittement.

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.

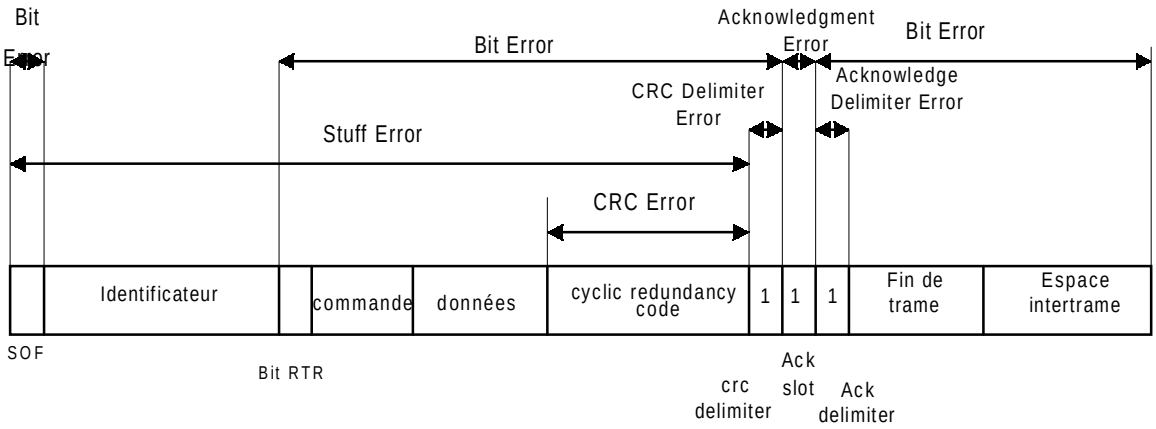

Figure 9 : Les sources d'erreur dans la trame CAN

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.

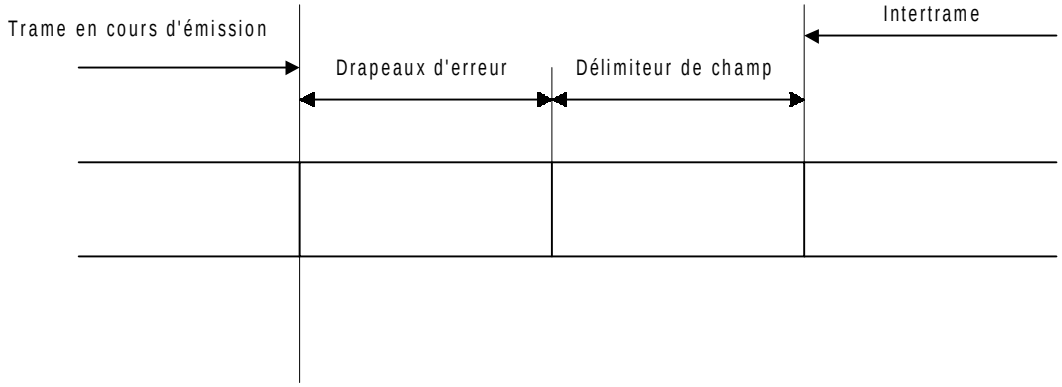

Figure 10 : Construction de la trame d'erreur

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 quelles contiennent. Les figures 11 et 12 représentent les deux types de trame avec leurs drapeaux respectifs.

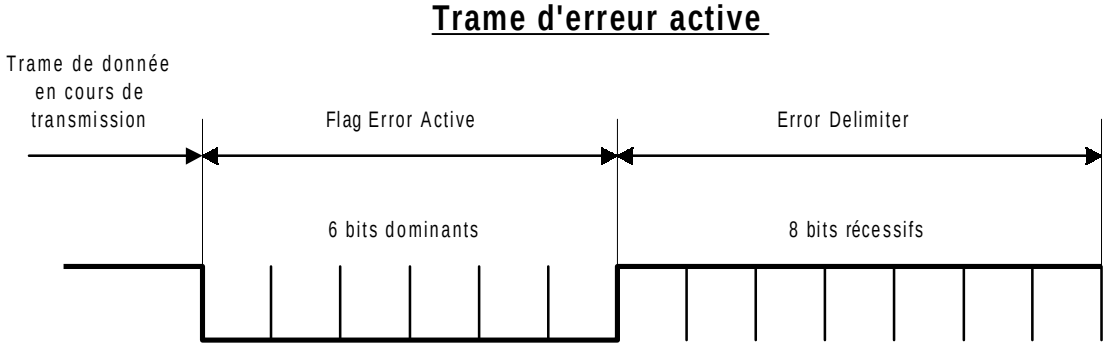

Figure 11 : Trame d'erreur active

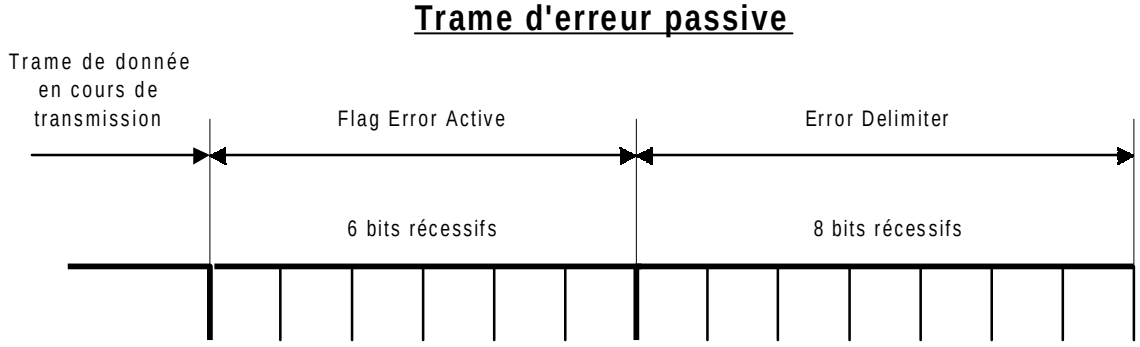

Figure 12 : Trame d'erreur passive

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 *d'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.

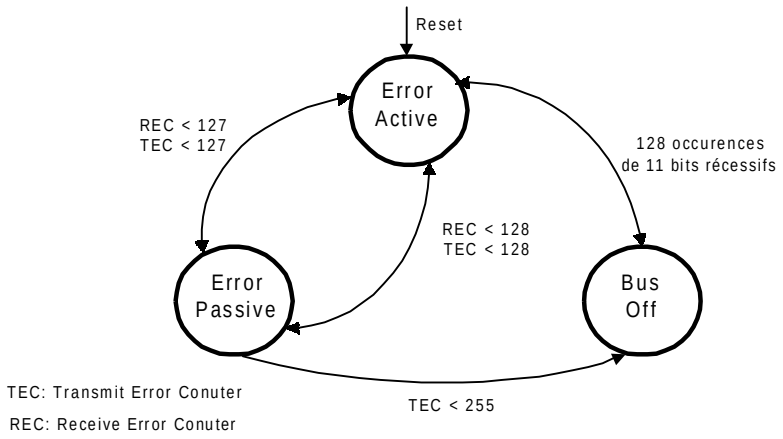

Figure 13 : Compteur d'erreur et état d'un noeud

Les règles de passages dans les modes :

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

Les règles d'incrémentation 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 Err*or 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'envoie 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'incrémentation). 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.

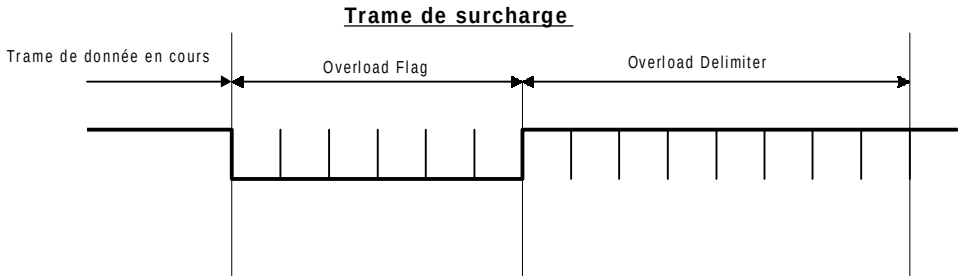

Figure 14 : Trame de surcharge

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.

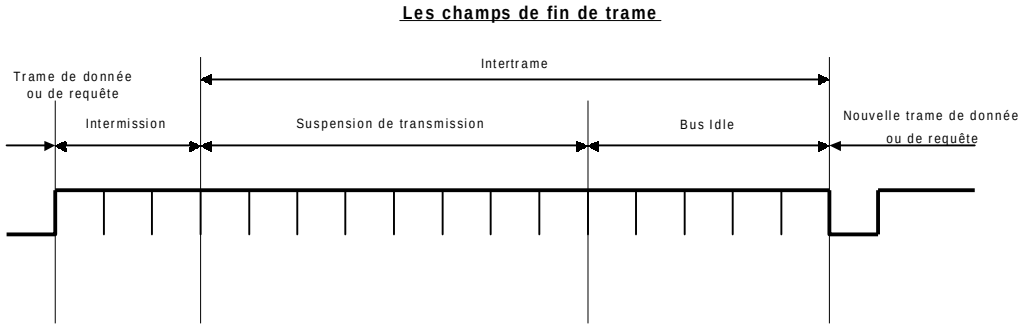

Figure 15 : Période d'intertrame

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.

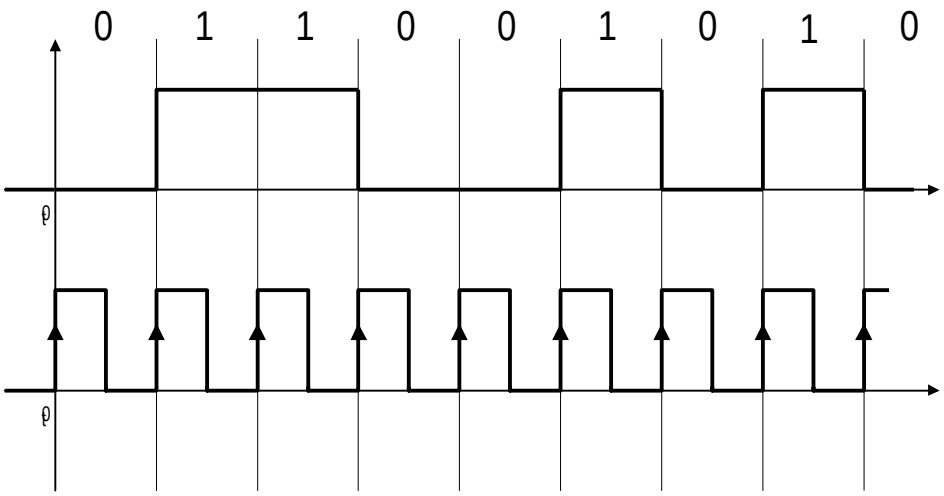

Figure 16 : Codage NRZ du bus CAN

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.

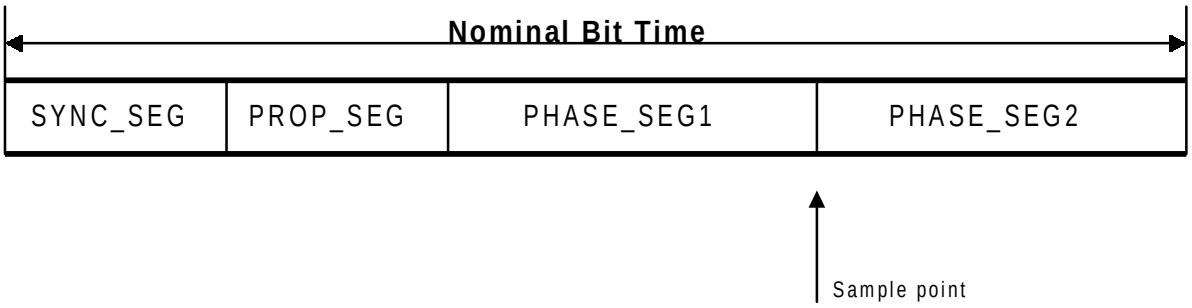

Figure 17 : Décomposition du nominal Bit Time

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:

$$\min\_Nominal\_Bit\_Time = \frac{1}{Nominal\_Bit\_Rate}.$$

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 303 à 313 ou de 313 à 303) 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 :

$$Durée\_PROP\_SEG\_ = 2\left(t\_propagation\_bus\_drivers\_retard + t\_propagation\_bus\_retard + t\_comparateurs\right) +$$

Les segments 3buffer phase13 et 3buffer phase23 : les segments 3buffer phase13 et 3buffer phase23 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 3buffer phase13 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* :

$$TIME\ QUANTUM = m * MINIMUM\ TIME\ QUANTUM$$

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.

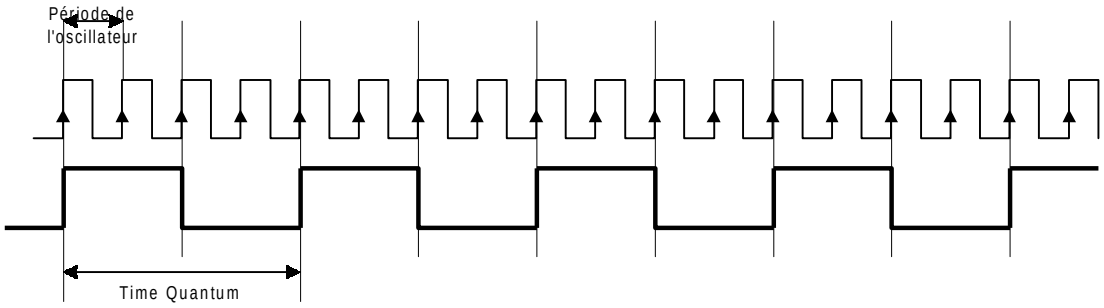

Figure 18 : Construction du Time Quantum

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*.

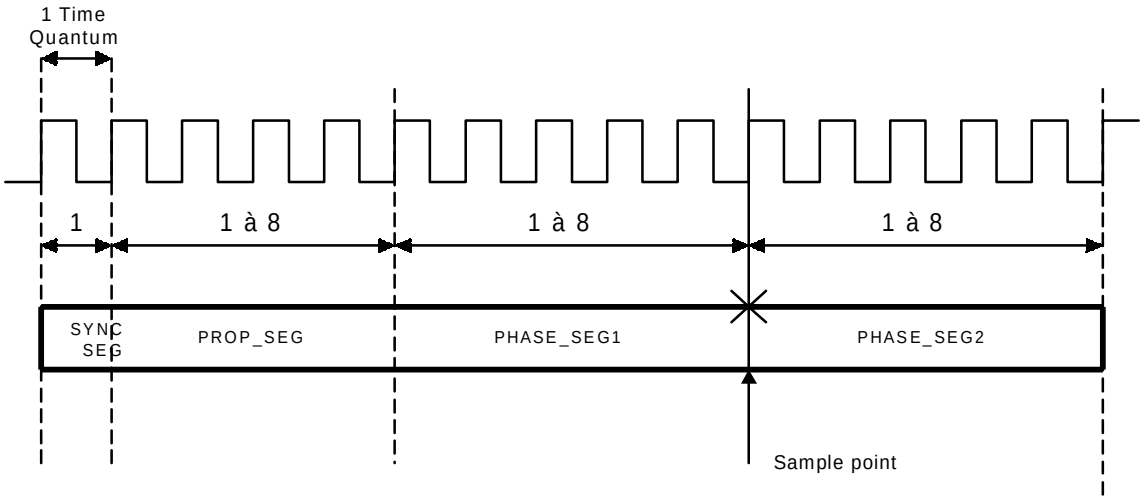

Figure 19 : Durée des différents segments

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.

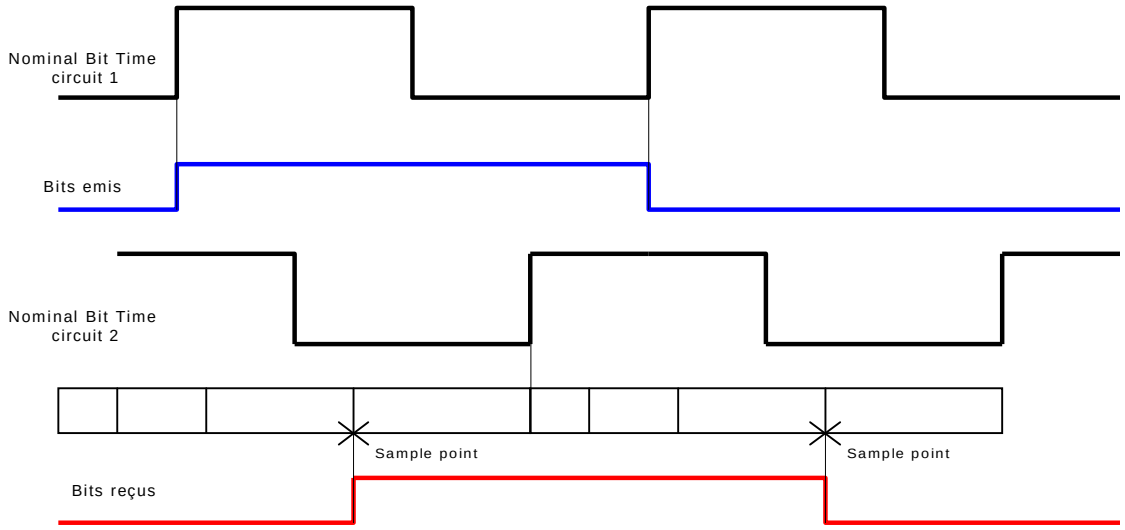

Figure 20 : Problème de la synchronisation

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).

$$1 \leq Rjw \leq \min(4, PHASE\_SEG1)$$

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*.

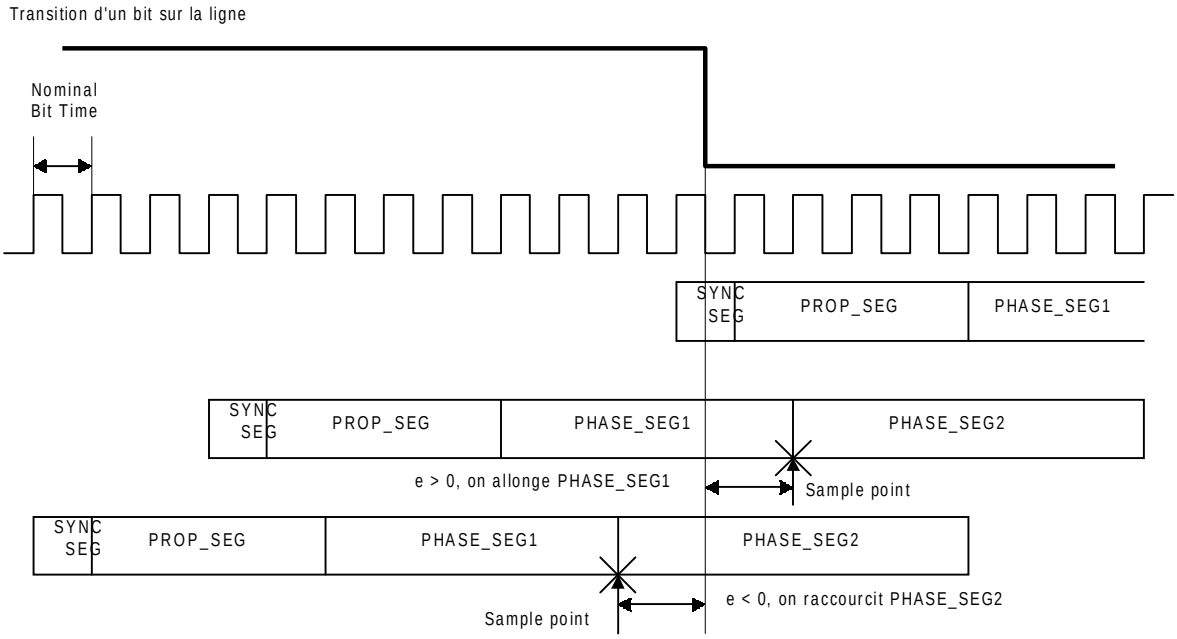

Figure 21 : Erreur de phase

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).

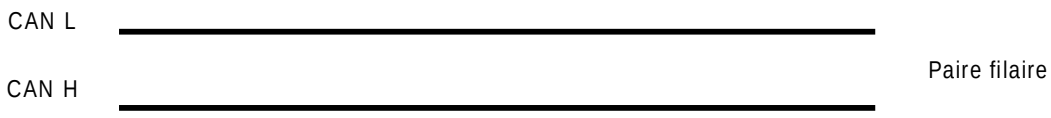

Figure 22 : Paire filaire du bus CAN

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	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 k&	25 à 50 mA sur 60&
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*120&
Tensions d'alimentation	5V	5V

Tableau 3 : Les 2 types de bus CAN

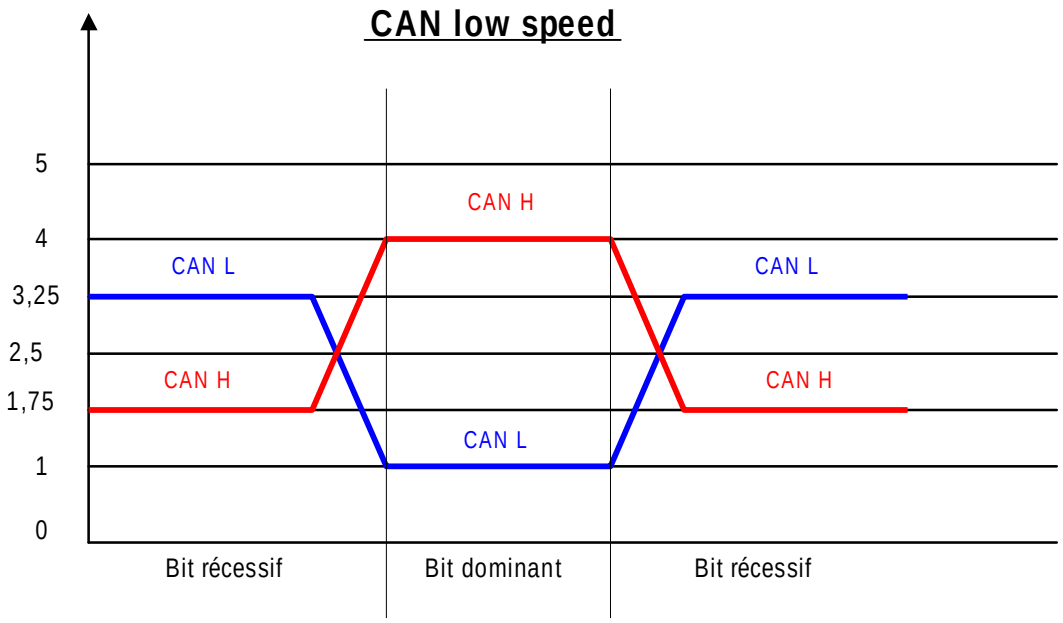

Figure 23 : Niveaux de tension du bus CAN *low speed*

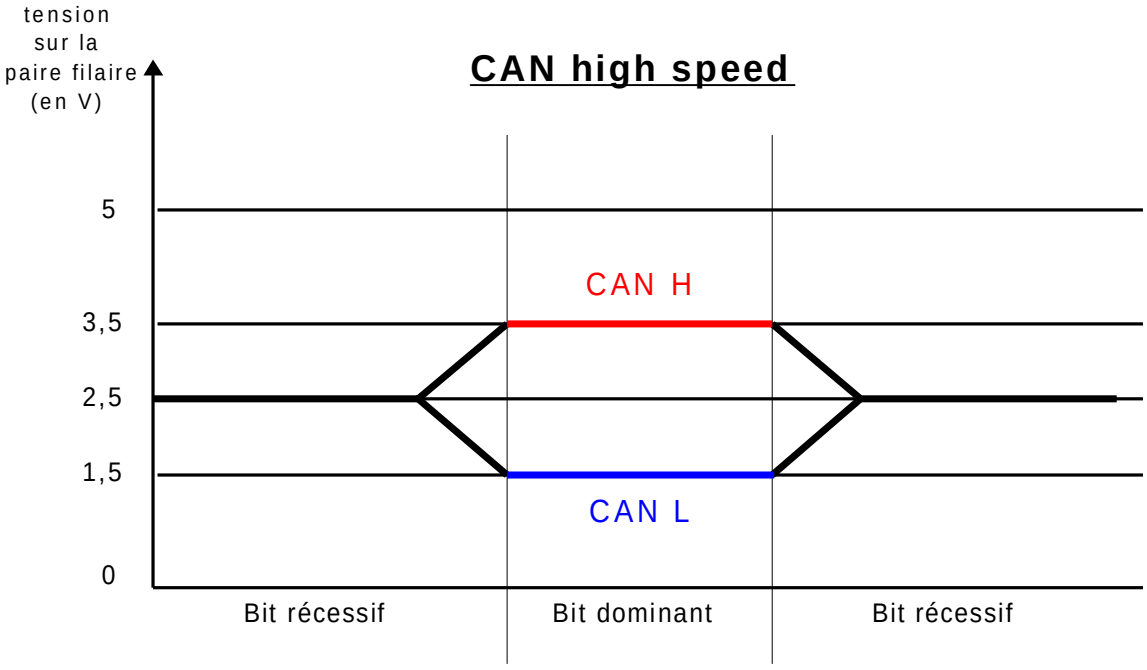

Figure 24 : Niveaux de tension du bus CAN *high speed*

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

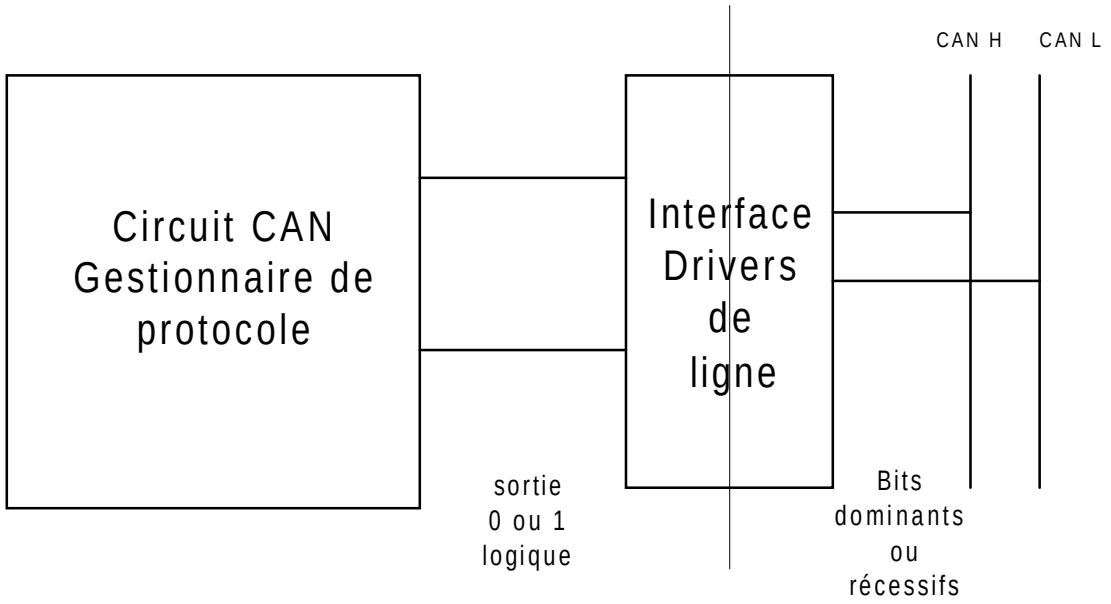

Figure 25 : Schéma de principe d'un circuit CAN

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 *Acknowlegde 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.

PROFILS CANOPEN

Sujet: Profils CANopen.
Verbe: Définir.
Complément: CANopen est un concept de réseau basé sur le bus sériel CAN (Controller Area Net- work) et la couche d'application CAL (CAN Application Layer). Avec CAL, il faut définir quelles
données seront transmises avec q uels ou tils ; c'est la tâche d e CANopen pour les applications dans les systèmes industriels en temps réel.

Le nombre d'équipements d'automatisation avec une interface CAN ne cesse d'augmenter et le besoin d'une interopérabilité entre ces équipements dans les installations multi-vendeurs se fait de plus en plus pressant. Aujourd'hui, le protocole ouvert CANopen basé sur la couche d'application CAN (CAL)
émerge sur le marché. L'interopérabili- té de CANopen a été démontrée à la foire industrielle de Hanovre 1996 dans une installation multi-vendeurs avec divers automates répartis, un robot, des systèmes d'entraînement et une quantité de modules d'entrées-sorties.

CAN en plein essor

CANopen est un concept de réseau basé sur le bus sériel CAN (Controller Area Network) et la couche d'application CAL (CAN Application Layer). Le bus bifilaire CAN qui, à l'origine, a été développé pour l'automobile, est aujourd'hui utilisé dans plus d'un million d'équipements industriels de commande, de capteurs et d'actionneurs. Ce bus est normalisé au niveau international selon la norme ISO 11898. Plusieurs grands fabricants de semiconducteurs fournissent des chips CAN et les quantités utilisées dans l'automobile garantissent le bas prix des chips ainsi qu'une disponibilité à long terme.

CANopen et CAL

La couche applicative CAL née au sein de l'organisation internationale CiA (CAN in Automation) est un langage généraliste pour les réseaux CAN dont la structure est similaire à celle du MMS. CAL propose un ensemble d'outils de communication sans décrire la façon de les utiliser (fig.1). Ainsi,

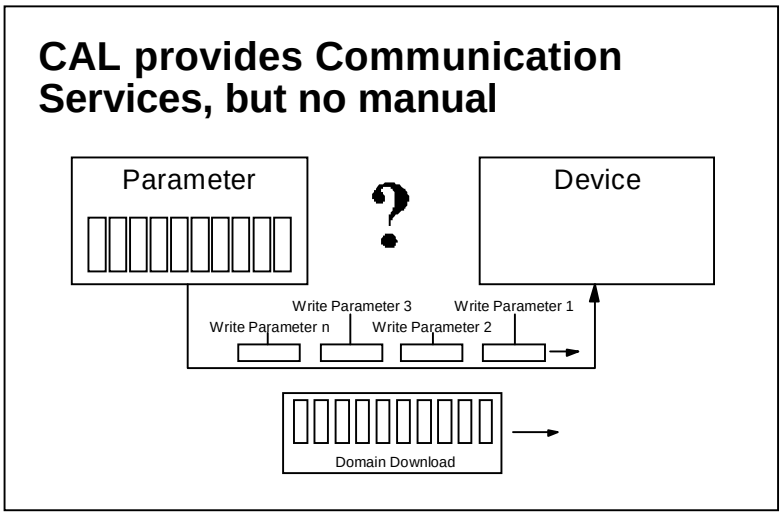

Fig. 1 CAL est un ensemble d'outils de communication mais ne définit pas la façon de les utiliser!

Fig. I: CAL est un ensemble d'outils de communication mais ne définit pas la façon de les utiliser.

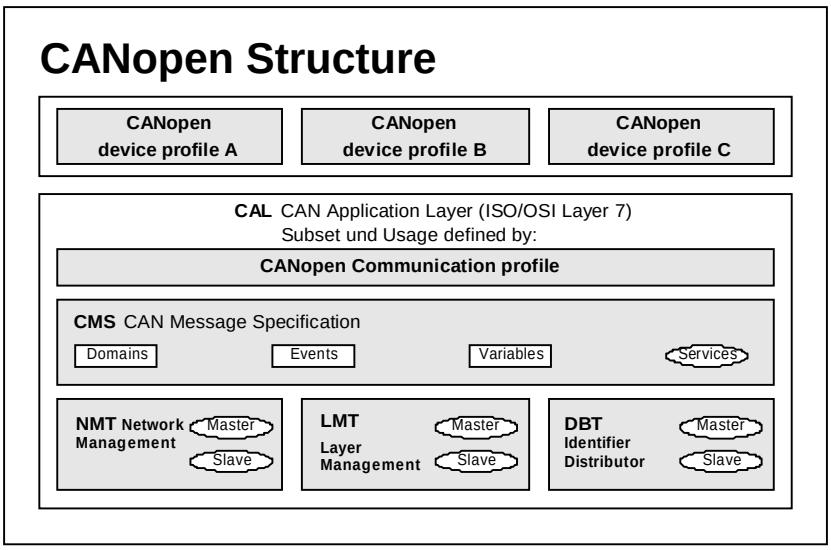

Fig. 2: CANopen est superposé à CAL et définit le contenu des données.

CAL est adopté à un très grand nombre d'applications. Cependant il faut, dans chaque cas, définir quelles données avec quels outils seront transmises. C'est la tâche de CANopen pour les applications dans les systèmes industriels en temps réel. CANopen définit encore ce que les données signifient dans les différentes catégories d'équipements (fig. 2). Pour définir une interface de communication adaptée aux équipements industriels, CANopen n'utilise qu'un certain nombre des outils de communication mis à disposition par la CAL. Ainsi CANopen ne nécessite que de faibles puissances de calcul et de capacité mémoire.

Le profil de communication

La manière dont s'effectue la communication avec les équipements est définie dans le profil de communication CANopen (CiA DS-301). Celui-ci est en quelque sorte le manuel d'utilisation qui permet d'établir une communication ouverte et interopérable avec CAL. Tous les équipements qui communiquent selon le profil de communication CANopen, peuvent parfaitement s'intégrer dans le même réseau physique. Dans la plupart des réseaux de systèmes d'automatisation on trouve deux genres bien distincts de données c'est-à-dire les données en temps réels et les données de paramétrages (fig. 3).

Les premières nommées PDO (Process Data Object) sont transmises rapidement de préférence sans "overhead" et avec une structure prédéfinie.

C'est un échange de données "pur CAN". Cependant les propriétés Broadcast de CAN restent intactes et un message peut être reçu et traité par chaque participant (fig. 4). Les PDO peuvent être transmises au choix de manière cyclique, événementiel ou synchrone (nécessaire pour la commande d'axes). En mode événementiel, la charge sur le bus est minimale et la capacité de communication très élevée pour un taux de transmission comparativement bas. Le mode synchrone (fig. 5), lui permet de synchroniser plusieurs axes dans la technique d'entraînements de moteurs, de lire des entrées en parallèle ou d'activer des sorties simultanément. Les deux modes peuvent aussi fonctionner en étant mélangés.

Deuxièmement, il y a les données de paramétrage nommées SDO (Service Data Object) dont les caractéristiques sont très différentes. Elles peuvent être nombreuses (beaucoup d'octets) et dans ce cas sont divisées en plusieurs segments. Elles sont typiquement transmises en asynchrone et la vitesse n'est pas critique. Le canal de données SDO est très performant et les paramètres d'un équipement donné peuvent être généralement écrits dans son répertoire objet (CAL multiplexed domaine) ou lu de celui-ci en un seul handshake (fig. 6).

En outre, la structure d'un répertoire objet CANopen correspond exactement à celle des autres systèmes de bus offrant ainsi une couche applicative quasi compatible.

Les profils d'équipements

Les profils d'équipements CANopen décrivent le contenu de la communication pour chaque type d'équipements. Celui-ci est indépendant des fabricants et

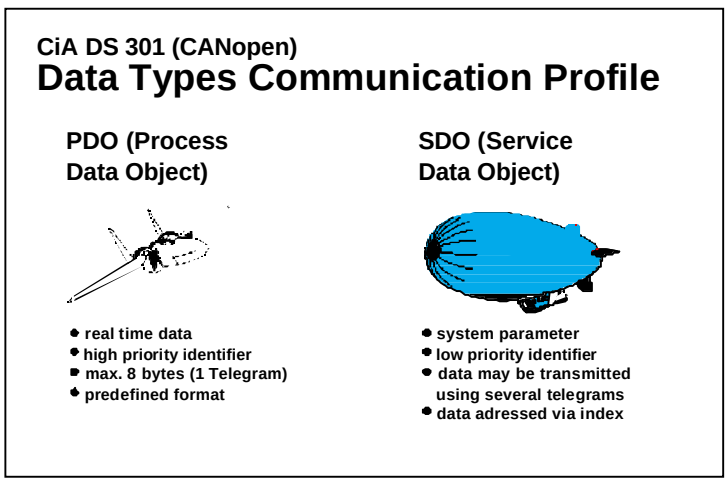

Fig. 3: Les services de CANopen sont optimi- sés pour les différents genres de données.

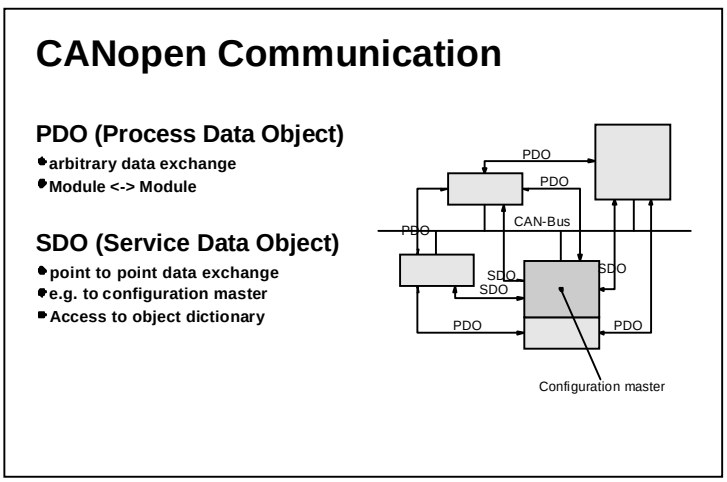

Fig. 4: Les PDO ont un caractère "Broadcast" tandis que les SDO sont "canalisés".

les fonctions de base d'un type d'équipements donné (par exemple: unités d'entrées-sorties, entraînements servo, onduleurs de fréquence) s'effectuent toujours de la même façon (fig. 7).

Ainsi, les équipements qui respectent le même profil sont échangeables entre eux. Même si beaucoup de paramètres sont déjà prédéfinis, les profils d'équipements CANopen offrent encore de l'espace pour des extensions "fabricants". Cette approche assure l'avenir de CANopen.

Le CiA s'occupe de toutes les spécifications CANopen. On y trouve entre autres les profils d'équipements pour unités d'entrées-sorties (CiA DS-401) et pour systèmes d'entraînement (CiA DS-402). D'autres profils d'équipements ainsi que des essais et la certification de produits CANopen sont en préparation. D'autre part plusieurs implantations CANopen (code OEM) sont disponibles sur le marché. A ce sujet, le CiA publie régulièrement un catalogue des produits CANopen.

La gestion du réseau

CANopen propose un démarrage (Boot-Up) simplifié du réseau qui peut être étendu modulairement selon les exigences de l'installation (fig. 8). Il n'est souvent pas nécessaire de configurer le réseau car des valeurs de défaut sont définies pour les paramètres du réseau et les fonctions des équipements. Si toutefois l'utilisateur veut optimiser son réseau ou y ajouter d'autres fonctions, il peut le faire à l'aide d'outils de configuration disponibles chez plusieurs fabricants.

Naturellement CANopen ne se prive pas des avantages de CAN comme par exemple de pouvoir communiquer sans maître dans le réseau. Dans un réseau CANopen, tous les participants ont les mêmes droits et l'échange des données s'organise directement entre eux. Ainsi CANopen est particulièrement bien adapté à la décentralisation de l'intelligence où plusieurs automates ayant les mêmes droits sont répartis dans un système.

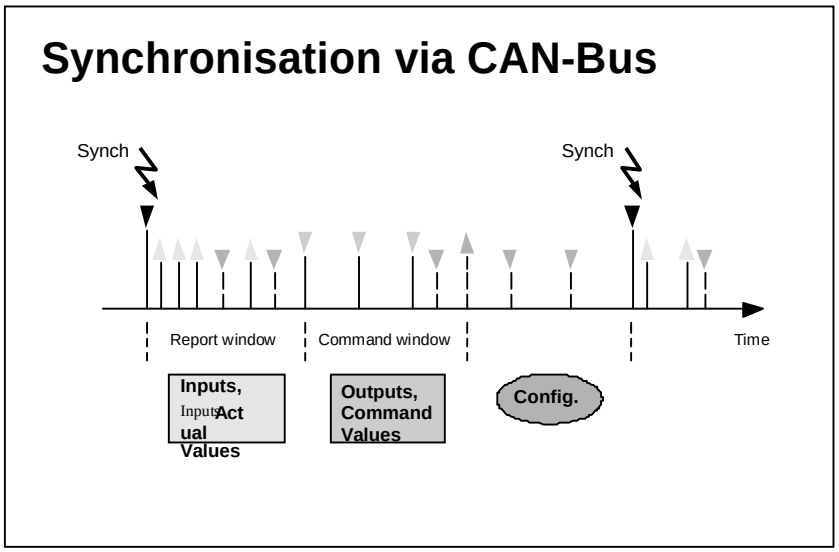

Fig. 5: Le mode synchrone répond aux exigences des entraînements d'axes.

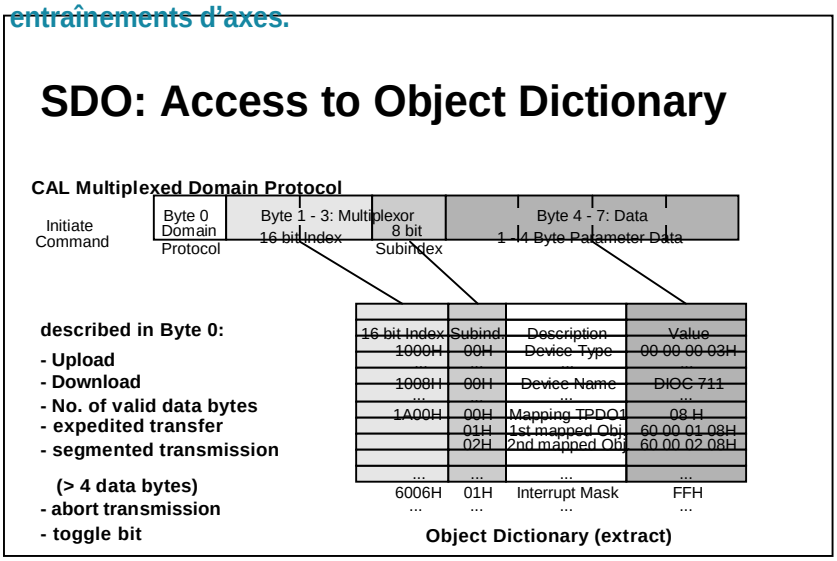

Fig. 6: Les SDO permettent d'accéder au répertoire objet.

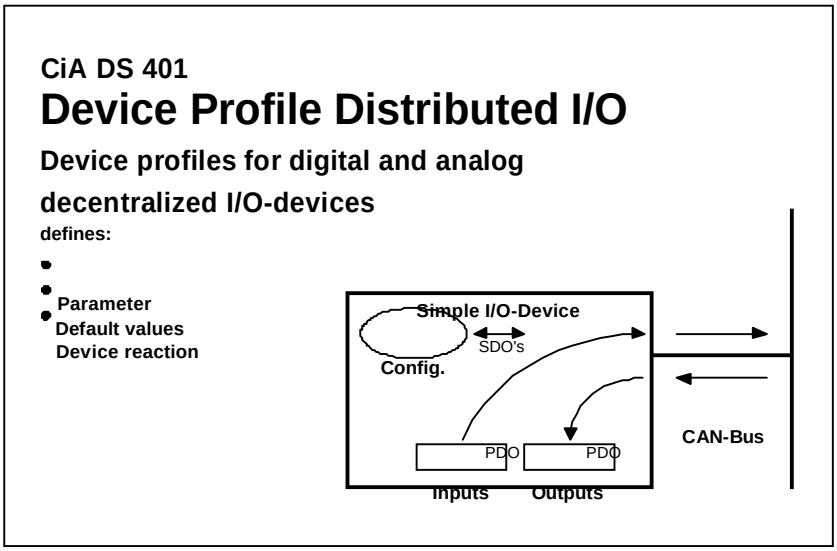

Fig. 7: Les profils d'équipements définissent le contenu de la communication et décrivent les équipements.

Un autre aspect à relever est la gestion "lean" du réseau CANopen. En effet, il permet aussi bien la mise en réseau d'équipements complexes dans les niveaux supérieurs que la connexion d'unités simples telles que capteurs et

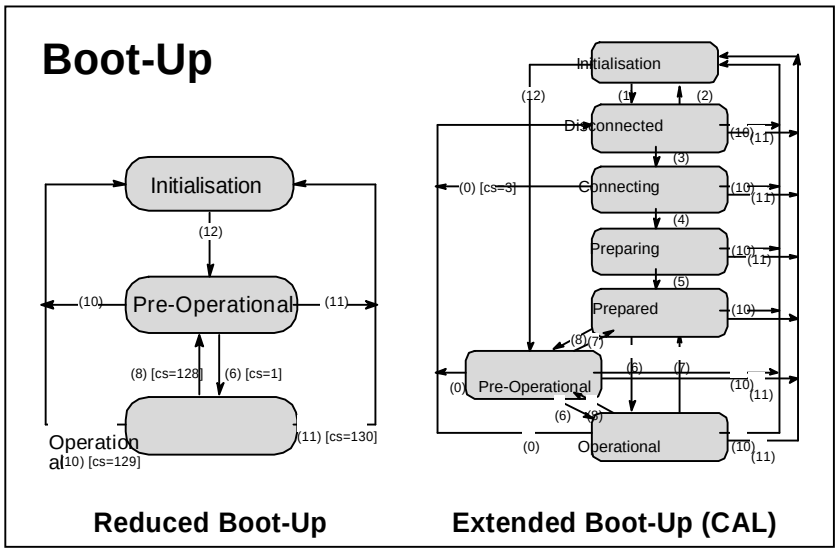

Fig. 8: CANopen propose un “Boot-Up” simplifié en sus du comportement CAL.

actionneurs proches des processus. Les passerelles qui relient le niveau terrain au niveau de communication supérieur ne sont pas nécessaires et même un PC peut “écouter” les données des processus dans un réseau CANopen sans influencer le trafic de données en temps réel. CANopen offre donc une solution de réseau “light” qui permet de diminuer le nombre de systèmes de bus et ainsi de réduire les coûts de formation, de mise en oeuvre et de stockage.

Flexibilité et sécurité

Dans un réseau maître-esclave conventionnel, la capacité en temps réel de tout le système est déterminée par l’unité maître. En général les unités esclaves ne peuvent communiquer entre elles directement, ce qui nécessite au minimum deux transferts de données. Cela rend la communication plus complexe mais aussi le risque d’erreurs de transmission plus important. CANopen évite ces inconvénients et le comportement temporel peut même être déterminé individuellement en fonction des tâches de chaque participant.

Il s’ensuit que tout le système de communication ne doit pas inutilement être plus performant si cela n’est exigé que par quelques participants. En outre, une tâche d’automatisation est répartie entre plusieurs participants CANopen et ainsi, les performances des équipements mis en réseau sont mises à profit de façon optimale. Cette tâche peut être petit à petit augmentée en ajoutant des participants.

Produits et applications

Le domaine d’utilisation de CANopen est très large. Des systèmes d’automatisation décentralisés, des systèmes d’acquisition et de commande répartis ainsi que la mise en réseau de capteurs et d’actionneurs sont facilement réalisables. Des produits tels que les unités d’entrées-sorties Selecontrol MAS-DP sont aujourd’hui disponibles avec le protocole CANopen. L’implantation de CANopen a été complètement réalisée autant dans les modules T.O.R. (fig. 9) que ceux analogiques (fig. 10) et ils peuvent ainsi être mis en réseau dans un environnement CAL classique. Un set d’initiation est aussi disponible chez Selectron. Il se compose d’une carte PC intelligente qui s’occupe de la liaison CANopen, de modules d’entrées-sorties, du câble, d’une bibliothèque de logiciel ainsi que d’un moniteur CANopen simple qui facilite l’initiation.

Les premières applications CANopen ont été réalisées dans l’industrie des machines (par exemple dans l’emballage, le textile ou l’impression) et dans la manutention (par exemple dans des robots et systèmes d’assemblage).

CANopen est aussi utilisé dans des installations de convoyage et de stockage, dans des systèmes embarqués (machines de chantier et palettiseurs) ainsi que dans la gestion technique des bâtiments (climatisation et élévateurs).

CANopen a même été intégré à des systèmes de traitement de l’image pour l’assurance de qualité.

Dipl.-Ing. ETS J.-L. Steiner
Selectron Lyss SA

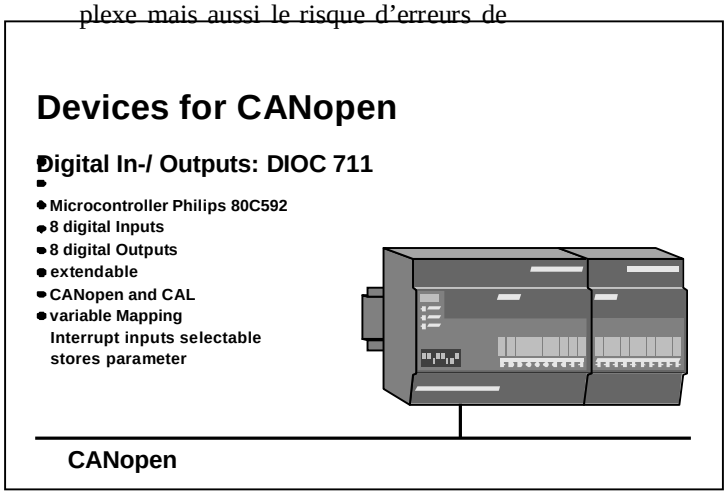

Fig. 9: Unités d’entrées-sorties T.O.R. avec CANopen.

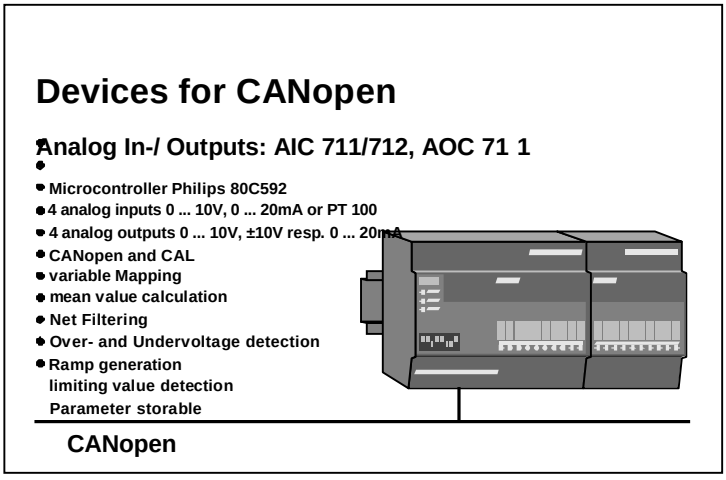

Fig. 10: Des possibilités étendues avec les uni- tés analogiques.

Performances des couches applications pour CAN

Avant de commencer à s'intéresser à son propre problème spécifique, la question la plus fréquemment posée par les utilisateurs potentiels de couches applicatives du CAN est de savoir quel est le meilleur protocole applicatif. Cette question ressemble étrangement à celle qui consiste à vouloir savoir quel est le meilleur moyen de transport, voiture, train, bateau, avion sans annoncer le lieu géographique où l'on désire se rendre, le temps que l'on souhaite mettre, etc. La réponse dépend du problème que vous désirer résoudre et, dans la plupart des cas de l'opinion et des goûts de l'utilisateur.

Dès à présent, nous tenons à signaler que le contenu de cet article n'a pas pour but ni de proposer préférentiellement telles ou telles couches applicatives, ni de les comparer entre elles puisque leurs origines, buts et vocations sont différents et qu'il ne sert à rien de comparer de choses non comparables. Nous ne présenterons donc pas de tableaux se voulant comparatifs mais, afin de pouvoir vous aider dans vos choix futurs, cet article présente, sous forme de tableaux, un résumé de leurs propriétés principales - à ne surtout pas donc prendre pour des tableaux comparatifs !! -.

Toutes les couches applicatives décrites ci dessous se basent sur les propriétés intrinsèques du protocole de communication CAN. Cet article n'a pour but que de faire ressortir les particularités et propriétés des couches applicatives hors de ce protocole.

Les différents paragraphes et tableaux qui suivent ont pour but de résumer brièvement les principales propriétés de ces couches au niveau des systèmes et principes :
- d'assignation des valeurs des identificateurs des messages,
- des méthodes d'échange et traitement des données,
- des possibilités d'établissement de communications point à point,
- des méthodes d'établissement des connexions de données,
- de l'administration du réseau,
- des principes de modélisation et profiles des éléments.

Assignation des valeurs des identificateurs des messages

La méthode utilisée pour déterminer l'assignation des valeurs des identificateurs des messages devant circuler

> Sujet: Les couches applications de Can.
> Verbe: Appliquer.
> Complément: Avant de commencer à s'intéresser à son propre problème spécifique, la question la plus fréquemment posée par les utilisateurs potentiels de couches applicatives du CAN est de savoir quel est le meilleur protocole applicatif.

sur le réseau a une incidence directe sur :
- la priorité relative du message,
- le temps de latence du message,
- les possibilités de filtrage du message,
- les structures possibles des communications,
- et enfin, l'efficacité de l'usage de l'identificateur.

Compte tenu des critères évoqués ci-dessus, le choix de la méthode d'assi-

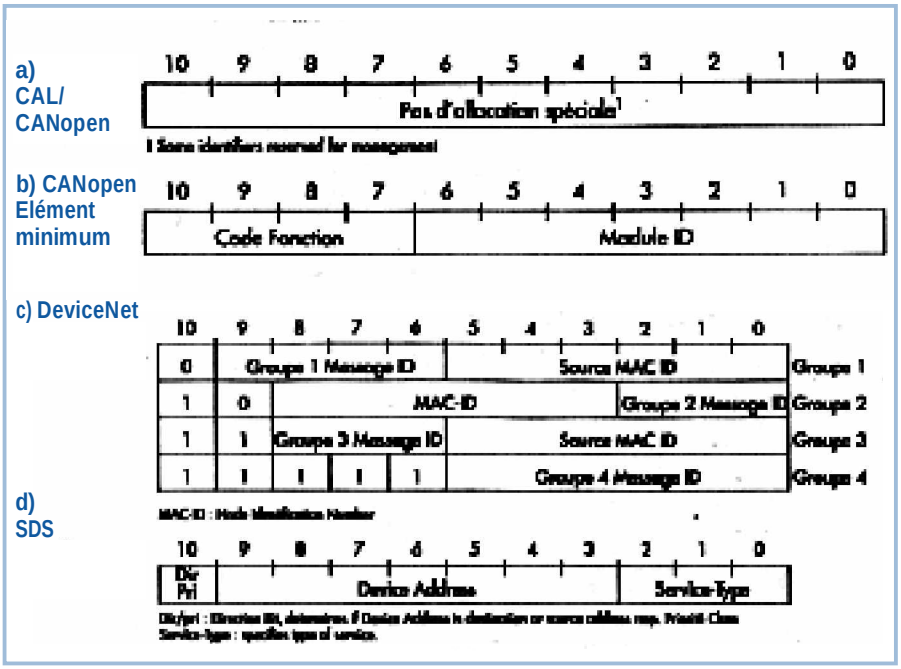

Tableau 1

gnation des valeurs des identificateurs dépend de la philosophie applicative que l'utilisateur souhaite mettre en œuvre dans son système.

Le tableau 1 indique les structures générales d'assignation des valeurs des identificateurs.

Méthode d'échange et de traitement des données

Un autre point intéressant à examiner est celui qui a trait aux modes et méthodes d'échange et de traitement des données. En effet, ceux-ci constituent l'essence même d'une application distribuée. Ils doivent être les plus efficaces possibles dans un environnement donné. De ce fait, la transmission d'une donnée spécifique à une application doit être réalisée en accord avec le mode «Producteur - Consommateur»

dans lequel la signification de la donnée transférée est sous-entendue par la valeur donnée à l'identificateur du message associé. Dans ce cas, Producteur et Consommateur sont supposés connaître la signification de la donnée transmise et savoir ce qu'ils ont à en faire.

Ce paragraphe s'intéresse donc à l'architecture et l'organisation des données présentes dans le champ de données (8 octets) de la trame CAN.

Architecture et organisation

En quelques mots :

| CAL

Globalement, CAL est un standard dans lequel les possibilités de communication sont indépendantes de l'application. CAL fournit des objets de communication en termes de «variable, événement» et «domaine». Avec des «variables» et «événements» de base, la transmission de données est effectuée sans enrobage (overhead) supplémentaire jusqu'à un maximum de 8 octets transmis par trame de communication. Les «variables» multiplexées quant à elles n'autorisent le transport que de 7 octets au maximum par trame.

| CANopen et DeviceNet

Au premier coup d'œil ces deux couches sont assez similaires en ce qui concerne les modes d'échanges et de traitement des données, respectivement «PDO» pour CANopen, les «I/O messages» pour DeviceNet.

Le tableau 2 résume les principales caractéristiques des CANopen, DeviceNet et SDS sur ce sujet.

| SDS

La structure de SDS est particulièrement bien adaptée aux solutions des systèmes distribués utilisant principalement des dispositifs «binaires» simples (organes d'entrées / sorties, valves, ...).

Déclenchement des messages

Le déclenchement des messages est supporté, avec quelques petites nuances, par les trois couches applicatives.

| CANopen quant à lui peut déclencher :
- sur un événement,
- sur une Application Request,
- après la réception d'un message de synchronisation.

| DeviceNet supporte les modes de déclenchement :
- cyclique,
- sur un changement d'état,
- sur la reconnaissance d'un objet particulier de l'Application.

| SDS, enfin, se déclenche :
- soit cycliquement,
- soit sur des changements d'état ou de valeurs.

	CANopen	DeviceNet	SDS
Nom de l'objet de communication	Process Data Object PDO	I/O message	Multicast Channel APD U
Nombre maximal d'objets de communication par élément	512 PDO transmis 512 PDO reçus	27 I/O messages transmis 1701 I/O messages reçus par élément	32 canaux Multicast chacun jusqu'à 32 objets embarqués
Longueur maxi. du champ de données	8 octets	8 octets si fragmentation longueur arbitraire	6 octets si fragmentation 64 x 4 octets
Protocole	en accord avec le CMS du CAL sans enrobage suppl. sans acquittement protocole de Notify, read «stored event»	**non fragmenté** transmission sans enrobage suppl trois classes de transport supportant : sans acquittement acquittement par le serveur objet de connexion acquittement par l'appli. **fragmenté** transmission sans acquittement protocole de fragmentation avec un octet d'enrobage par trame	**non fragmenté** transmission 2 octets suppl. d'enrobage sans acquittement **fragmenté** transmission 4 octets suppl. d'enrobage par fragment acquittement protocole de fragmentation acquit. après réception d'un bloc complet
Types de mode de production de messages déclenchés	sur requête de l'appli. locale sur requête d'appli déportée cyclique non cyclique synchrone	spécifique à l'appli. changement d'état cyclique	changement d'état/valeur cyclique
Cartographie des objets d'application	le nombre max. des objets d'applications/PDO dépend de la taille des objets (64 appli. de 1 bit max.) définition de l'objet au moyen de «mapping paramètre record»	nombre arbitraire mappable à l'aide du protocole de fragmentation définition de l'objet au moyen de «Assembly object»	le Network data descriptor définit la taille, type de données, des données I/O de l'objet embarqué

Tableau 2

J’automatise N°8 -
Janvier - Février 2000

Cartographie (mapping)
des «objets Application»
Généralement un élément produit ou consomme plus d'un objet lié à l'application et assemble plus d'un objet applicatif dans un PDO ou I/O message approprié. Ce paragraphe s'intéresse à la façon dont sont répartis ces objets dans l'application.

CAL et SDS supportent des principes simples et CANopen et DeviceNet supportent quant à eux des systèmes beaucoup plus élaborés

CAL
Dans les applications conçues autour de CAL, la cartographie des données d'application est effectuée par le programmeur lors de la définition des objets de communication (variables CMS, événements CMS, ...).

CANopen
CANopen spécifie la cartographie des objets d'application dans le PDO au moyen d'une structure de données appelée «PDO Mapping Record». Cette structure spécifie cela sous forme d'une liste d'identification des objets (dictionnaire d'objets, avec index et sous-index) et de longueur de données Du fait que la cartographie des PDO est accessible au moyen des SDO, celle-ci est configurable au moyen de outils de configuration.

DeviceNet
Dans DeviceNet, le groupage des données d'application est spécifié au moyen des instances de l'objet - «Assembly» (chemin de connexion producteur ou consommateur) - qui définissent le format des données objet d'application. Un élément peut contenir plus d'un I/O Assembly et la sélection de la bonne Assembly peut être une option de la configuration de l'élément.

SDS
SDS décrit la cartographie des données d'entrées / sorties de l'objet embarqué dans le champ de données de l'APDU au moyen du «Network Data Descriptor». Cette information peut être lue par les autres éléments du réseau.

Le tableau 2 résume les principales caractéristiques de ces couches en ce

Le pourquoi de l'absence des couches 3 à 6 du modèle de l'ISO / OSI dans le CAN

Une question se pose très souvent : pourquoi les couches de 3 à 6 du modèle OSI / ISO sont-elles absentes du protocole CAN ? Cette absence est due aux raisons sui- vantes.

couche 3 - Network layer -
La structure même du protocole et le principe de diffusion adopté pour le CAN - dif- fusion de type «broadcast» - font que chaque message émis atteint TOUS les modules présents sur le bus et que, par conséquent, il n'y a pas nécessité d'effectuer des interconnexions entre réseaux ou d'avoir des fonctions de routage.

couche 4 - Transport layer -
Dans le modèle de référence ISO / OSI, le but de cette couche est d'autoriser les couches supérieures à «fiabiliser le transfert des messages de longueur arbitraire sur des réseaux non fiables» en offrant des fonctions telles que fragmentation, segmenta- tion, séquencement, essai automatique et détection de trame dupliquée. Cependant pour des applications de commandes distribuées fonctionnant en temps réel, chaque message transféré tente de se passer de lui-même. Ces types d'applications requiè- rent de grands débits de transfert, des messages courts et nécessitent de connaître immédiatement si une tentative de message a réussi ou échoué de façon à être capable d'agir en temps et en heure.

Du fait que la couche 3 - Network layer - n'est pas nécessaire, et que la couche 2 - Data Link layer - du CAN est réputée pour être suffisamment fiable (gestion et traite- ment très performant des erreurs de communication, ...), les applications CAN ne nécessitent pas d'implémenter la couche 4 - Transport layer - pour garantir un servi- ce fiable de transfert de messages. De plus la couche 7 - Application layer - est tota- lement apte à fournir des services qui autorisent les applications nécessitant d'envoyer des messages de longueurs arbitraires.

Ceci conduit à conclure que la fonctionnalité de la couche 4 - Transport layer - n'a pas lieu d'être pour le protocole CAN.

couche 5 - Session layer -
Dans des applications distribuées industrielles de commandes en temps réel, les concepts de sessions, points de synchronisations et mécanismes d'enrôlements ne sont généralement pas supportés. Cependant, dans le futur, le l'organisme CAN in Automation - CiA - se réserve le droit d'implémenter une option de couche Session pour supporter des réductions de puissance à l'aide de possibilités de Stand-by.

couche 6 - Presentation layer -
La couche de présentation s'intéresse au transfert via le réseau des données d'appli- cation et à leurs significations. Dans le Modèle de Référence CAN toute application doit utiliser une structure constituée de données de type «basic» pour décrire leurs données. Cette donnée est codée au travers d'une syntaxe de transfert et il est suppo- sé que toutes les applications connaissent à priori le sens des données. Ceci conduit à conclure qu'il n'y a pas de fonctionnalité de la couche 6 - Presentation layer - pour le protocole CAN.

Il ne reste pour le CAN que les couches 1, 2 et 7 - Application layer - qui aura pour mission d'assurer l'interfaçage entre l'environnement de communication de données et l'application visée et qui utilise cet environnement pour coopérer avec d'autres applications. L'ensemble des applications coopérant ensemble via le réseau forme ce que l'on a généralement l'habitude d'appeler une «application distribuée».

Les applications basées sur des réseaux CAN

Le CAN permet de satisfaire de très nombreuses possibilités de communication et, de fait, la réalisation d'un très large éventail d'applications, principalement celles des marchés de l'Auto- matisme Industriel, des réseaux de terrains industriels et du secteur de l'Automobile. L'une des principales raisons de ce succès est la très bonne valeur du rapport coût / performances de ce système. CAN offre en effet un débit suffisamment rapide et un niveau de fiabilité élevé de transport des données pour un prix comparativement faible face à tous ses concurrents. A ce sujet, il est bon de rappeler que les prix des composants CAN sont en baisse constante du fait de l'explosion de son uti- lisation.

Applications Industrielles

Les applications Industrielles du CAN ont démarré dès l'appa- rition des premiers composants sur le marché et touchent prin- cipalement le marché de la communication dans l'automatis- me tels que les commandes logiques programmables (PLC), les commandes de robots industriels, les commandes intelligentes de moteurs, les organes d'entrées / sorties - capteurs, comp- teurs, actionneurs / actionneurs - intelligents, les systèmes hydrauliques, les commandes d'ascenseurs (exemples : KONE, OTIS), les équipements pour bateaux (exemple : Chantiers de Saint Nazaire), en passant par les équipements médicaux
(exemples : Philips Médical, Général Electric), etc.

Citons, en vrac, les principaux centres d'applications que sont :
- manutention sur chaîne de production,
- machines d'assemblage,
- palettisation,
- traitement de produits alimentaires,
- machines spéciales,
- commandes de machines pour l'industrie textile (broderie, bobinage, ...),
- fabrication de verre,
- tri de marchandises,
- industrie sucrière,
- équipements de bâtiments (climatisation de buildings, de tours, ...),
- jouets (commande de réseaux de trains électriques minia- tures, ...),

etc.

Applications automobiles

A son origine, les applications CAN ont été développées de façon prédominante par son concepteur, la société R. Bosch, pour réaliser des commandes dites de " temps réel " dans l'Automobile.

Malgré ses origines, la pénétration du CAN dans l'électro- nique du marché de l'Automobile a été effectuée un peu plus lentement que celle dans le milieu Industriel du fait, d'une part, de la forte pression sur le coûts des équipements et, d'autre part, du fait du caractère critique de l'aspect sécuritai- re que doivent posséder certaines des applications (ABS par exemple !).

Il est aussi à noter que, techniquement, le secteur Automobile a dû apprendre à s'adapter aux systèmes de commandes distri- buées en temps réel, ce qui n'était pas son lot quotidien, contrairement au secteur Industriel qui lui était déjà longue- ment habitué à ce style d'architecture et pouvait utiliser immé- diatement CAN sans aucune réorientation culturelle.

CAN permet l'implantation de boucles de commandes et de contrôles dans la plupart des unités de commandes électro- niques des véhicules. Cependant, les utilisations de telles boucles se trouvent limitées à des applications dans lesquelles les temps de réponse réalisables avec CAN sont suffisants courts. La valeur minimale de ce temps de réponse est limitée en raison du débit maximal du CAN de 1 Mbit/s lorsque la lon- gueur du bus ne dépassent pas 40 mètres. De plus, la valeur de ce temps de réponse peut être plus longue du fait de la présen- ce possible de conflits d'accès lorsque plusieurs stations ten- tent de démarrer simultanément la transmission de messages qui, par principe, ne peuvent avoir le même niveau de priorité.

De futures implémentations de fonctions améliorées telles que les commandes dynamiques et systèmes automatiques de co- pilotage et aide à la navigation nécessitent des actionneurs sur les freins, sur la direction, ... avec des interfaces électriques. Dans de telles applications, il peut devenir nécessaire d'implé- menter des boucles de commandes sur le bus avec des temps de réponse plus courts qu'il n'est possible de réaliser avec des solutions de communications habituelles. Ceci peut conduire à un développement de bus de communication temps réel plus puissant avec des temps de réponse déterministes.

Pendant le développement de systèmes bâtis autour du CAN, il est aussi apparu qu'en plus des commandes temps réel, celui- ci pouvait être utilisé pour mettre en réseau d'autres parties applicatives du véhicule, notamment une partie importante de la communication entre les différents éléments des parties châssis, habitacle et confort.

Dans ces zones du véhicule - souvent appelées zones de «multiplexage» - la complexité, la longueur, le poids (donc la consommation du véhicule) du harnais de câbles (de l'ordre de 1à 2,5 km et jusqu'à 30 à 50 kg parfois) peuvent être for- tement réduits par une mise en réseau des stations. Ceci rend plus aisée l'installation de l'électronique et peut, dans cer- tains cas, réduire aussi les coûts. De plus CAN est parfait pour assurer la solution aux problèmes de communication entre éléments pour les diagnostics, l'intégration d'interfaces utilisateurs (écran, ...) des différentes unités de communica- tions mobiles.

Pour conclure sur ce point indiquons qu'à ce jour, ces disposi- tifs sont appliqués aussi bien aux véhicules particuliers qu'aux camions (tracteurs et remorques), autobus, autocars, bateaux, machines agricoles, excavatrices, machines des services rou- tiers (désenneigeuses d'aéroport internationaux, ...).

concerne les modes de communication des messages.

Communication point à point

Résumons en quelques mots les grandes particularités des principales couches applicatives concernant leurs possibilités d'établir des communications dites «point à point».

CAL

Pour cet usage, CAL fournit des «Configuration Services» à l'aide de canaux d'administration pour chacun des éléments faisant partie du NMT (Network Management) du CAL. Deux identificateurs particuliers sont réservés à cet usage.

CANopen

De son côté CANopen fournit des «Service Channels» au travers desquels des SDO peuvent être échangés - à l'aide du protocole des domaines multiplexés - entre n'importe lesquels des noeuds disposés sur le réseau. Rappelons que ce protocole d'échange fournit un acquittement explicite aux trames transmises sur le réseau.

	CANopen	DeviceNet	SDS
Nom	Service Data Channel	Explicit Message	Peer to Peer Channel
Nombre maximal de canaux	128 clients SDO 128 serveurs SDO par élément	27 messages explicit transmis 1701 messages explicit reçus par élément	4 canaux par objet embarqué 32 objets embarqués par élément logique
Protocole	<5 octets : avec acquittement non fragmenté transmission fragmentée (7 octets par fragment) chaque trame est acquittée longueur indifférente	<7 octets : avec acquittement non fragmenté transmission fragmentée (6 octets par fragment) chaque trame est acquittée longueur indifférente	<6 octets : avec acquittement non fragmenté transmission fragmentée (3 octets par fragment) acquittement de chaque bloc 255 octets au maximum
Etablissement des connexions	établissement dynamique au moyen du manager des SDO connexions prédéfinies par défaut	établissement dynamique au moyen du manager des messages non connectés pour élément du groupe 2 allocation de message explicites pour le jeu de connexions prédéfinies	établissement dynamique au moyen du manager de connexions jeu de maître/esclave du jeu de connexions
Services de connexions et arguments	initier, avorter charger, décharger segment, domain index, sous-index du dictionnaire	ouverture, fermeture service pour : - création - configuration - démarrage - arrêt - reset - etc. code service chemin accès aux attributs arguments de services	ouverture, fermeture lecture écriture événemen t action code service numéro du canal attribut/action/évènemen t identificateur

Tableau 3

DeviceNet

DeviceNet fournit des services et canaux multi-applications orientés objets. L'emploi des «Explicit Messages» s'effectuent à l'aide des «Explicit Messaging Connections».

SDS

Pour sa part, SDS fournit des canaux de communication directe entre n'importe lesquels des «Embbeded Objects» (objets embarqués) dans les différents «Éléments Logiques» contenus dans un élément.

Le tableau 3 résume les propriétés fondamentales de connexion point à point de CANopen, DeviceNet et SDS.

Méthodes d'établissement de connexions de données

Ce point est très lié à la distribution statique et/ou dynamique des valeurs des identificateurs aux différents participants du réseau. Toutes les couches possèdent leurs propres spécificités concernant ce point. CAL et CANopen sont basées sur le DBT du CAL et, DeviceNet est basée sur une philosophie orientée sur le fait que les éléments du réseau sont propriétaires d'un lot d'identificateurs de messages.

Administration du réseau

L'administration des réseaux CAL et CANopen est basée sur le «NMT» des CAL et utilise le «Node Guarding» pour la détection des nœuds défectueux. De son coté DeviceNet supervise chaque nœud à l'aide des mécanismes de recherche de «Duplicate MAC ID» et de «inactivity / watchdog timer».

Modélisation et profiles des éléments

Les trois couches emploient des techniques de modélisation et de profiles dont les approches reflètent des sensibilités assez différentes.

Dominique PARET
Automotive & Identification
Innovation & System
Senior Technical Support
Philips Semiconducteurs

Cet article est extrait du deuxième tome consacré au bus Can chez Dunod
Tome 1 : Bus Can - Description, de la théorie à la pratique
Tome 2 : Bus Can - Applications.

Afin de vous présenter dans ce chapitre une synthèse constructive des principales proposi - tions industrielles existant sur le marché, de nombreux éléments ont été empruntés à une excellente étude de notre ami, M. Konrad Etschberger, Directeur Technique de la société alle - mande STZP, présentés lors des conférences ICC 97 du CiA.

Petit inventaire des requêtes et nécessités des applications

En quelques lignes, voici les principales requêtes et nécessités spécifiques des applications Industrielles et Automobile. applications Industrielles

Il y a quelques années, une enquête effectuée en Allemagne par la société VDMA auprès de très nombreux fabricants d'équipements industriels (automatismes, ...) avait donné les résultats suivants en ce qui concernait leurs souhaits : **communication de type synchrone** en %

- requise 64
- non requise 36

mode d'échantillonnage pour les nœuds du réseau

- échantillonnage cyclique 73
- échantillonnage commandé par un événement 27

périodicité des cycles

- 1 ms 26
- 10 ms 48
- 100 ms 26

temps de réponse du signal

- 1 ms 40
- 10 ms 34
- 100 ms 26

longueur du bus

- inf. à 100 m 66
- inf. à 1000 m 31
- sup. à 1000m 3

Même si ces chiffres ont légèrement évolué depuis l'époque de l'enquête, les tendances indiquées dans ce tableau restent sensiblement identiques.

Applications Automobile

En ce qui concerne les applications Automobile, il est néces- saire de distinguer au moins deux grands centres d'intérêt que sont le «contrôle moteur» et «l'habitacle» du véhicule.

contrôle moteur

- distance courte : quelques mètres
- rapidité de communication : débit de l'ordre de 50 à 500 kbits/s
- applications temps réel : commandes ordres en quelques micro secondes
- périodicité des cycles : 10 à 100 ms
- haute sécurité des connexions physiques
- possibilités de diagnostics
- possibilités de configuration des stations

habitacle

- distance plus longue : quelques dizaines, centaines de mètres
- communication plus lente : quelques dizaines de kbits/s à 125 kbits/s
- protection améliorée des lignes : principe de «bus failure managements»

Les autres performances requises sont très similaires de celles nécessaires aux applications Industrielles.