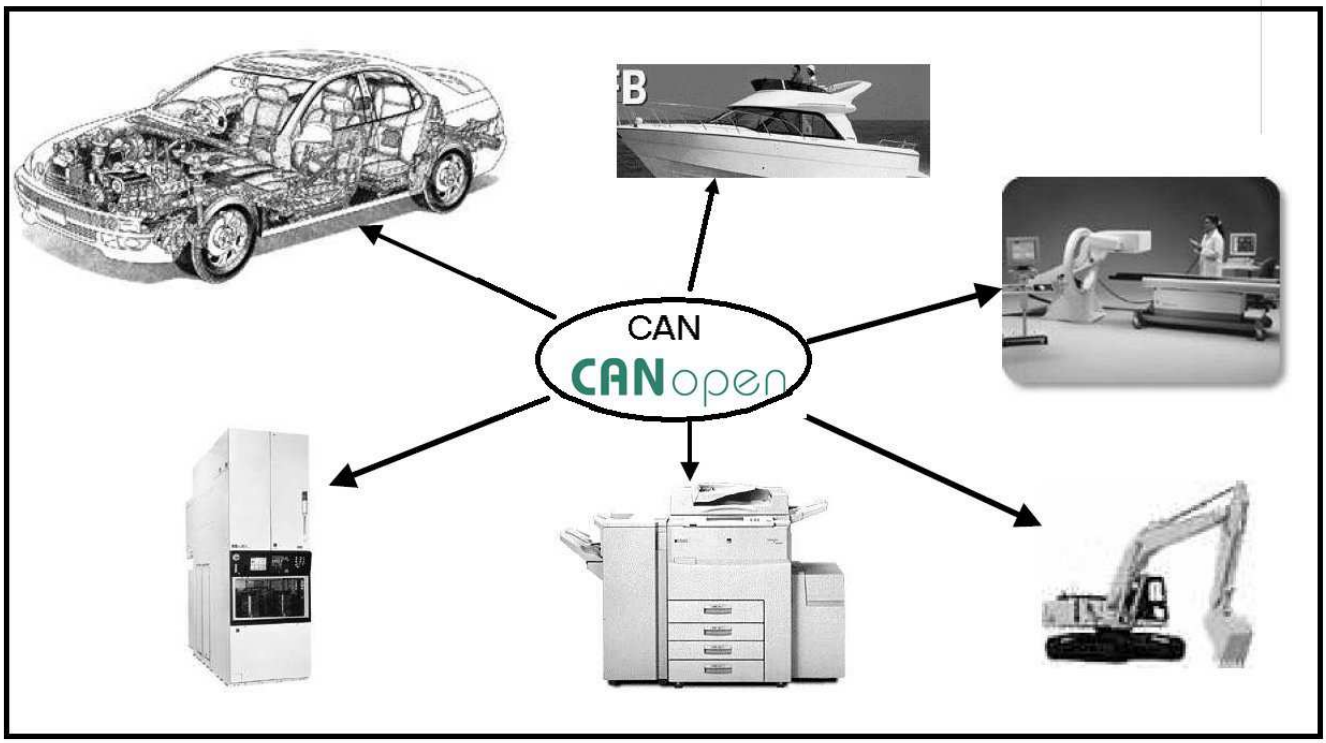
LE BUS DE TERRAIN

CAN

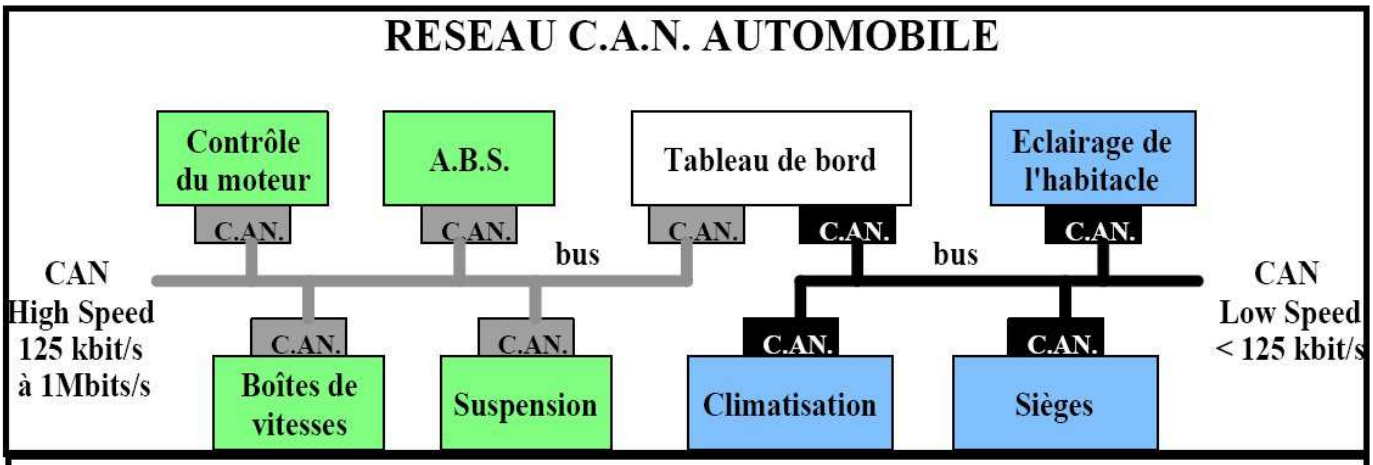


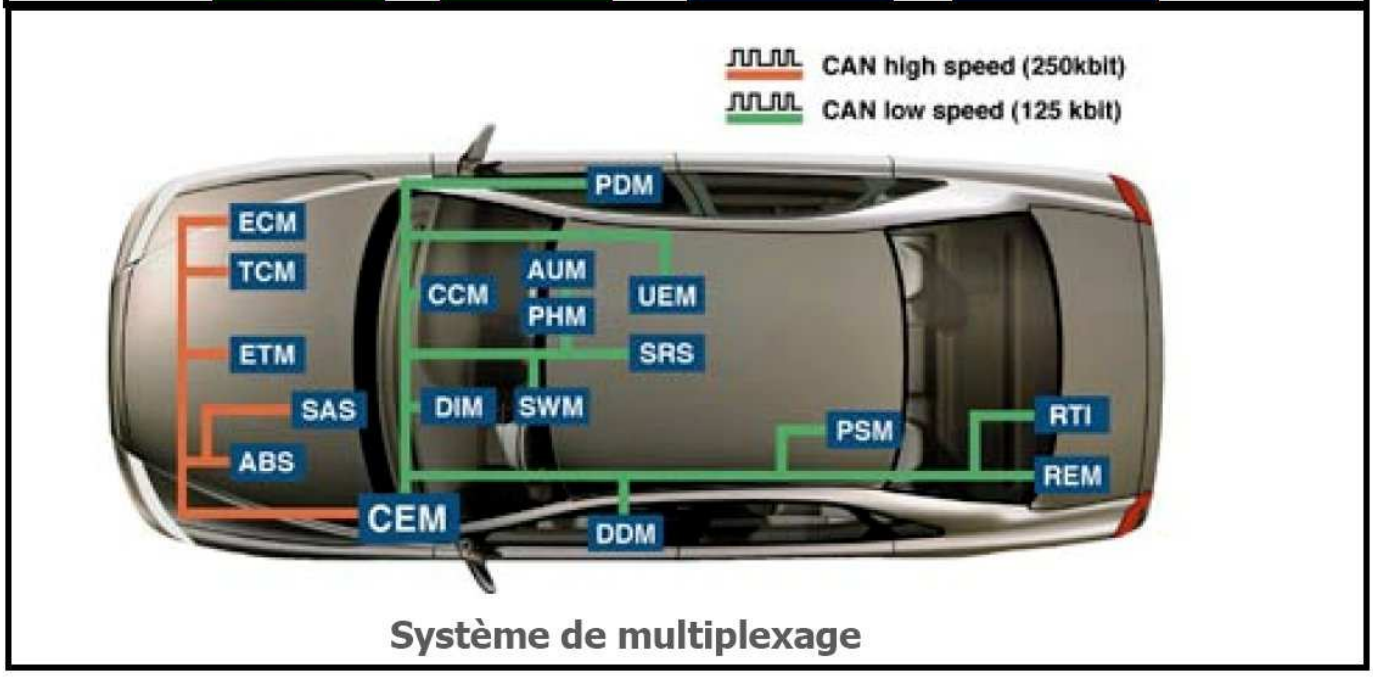
LE BUS CAN

1) Introduction

1.1) Position du réseau

Le Bus CAN, Controller Area Network, a été développé par l'équipementier Bosch dans les années 80 en partenariat avec la société Intel. Il a été utilisé la première fois par la Mercedes Classe S en 1991. Il est utilisé par la majorité des constructeurs européens et commence à s'imposer chez les constructeurs américains et japonais.
On les retrouve dans les systèmes embarqués Automobile, avionique, machines agricoles, batiment, electromenager...





Système de multiplexage

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 noeud.
- déconnexion automatique des noeuds défectueux.
- Réduction des couts de câblage et de maintenance

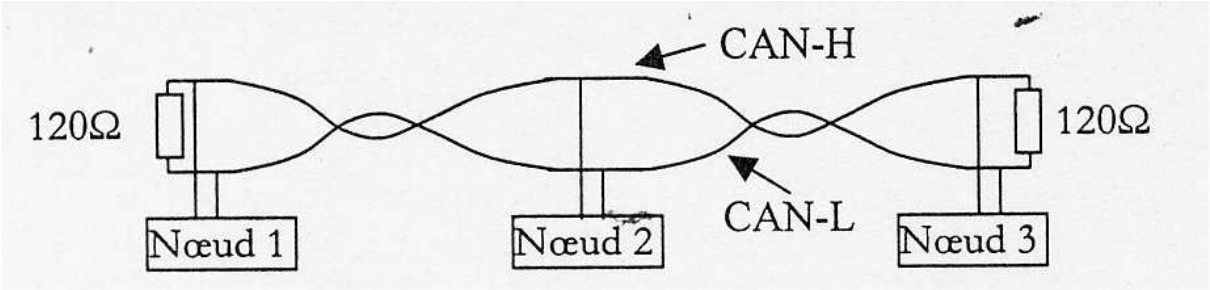
1.2) couches du modèle OSI utilisées

OSI	BUS CAN	BUS CANOPEN
Couche application	Spécifié par l'utilisateur	CAL:CAN application layer for industrial application
Couche présentation	-	-
Couche session	-	-
Couche transport	-	-
Couche réseau	-	-
Couche liaison données	MAC /LLC CAN 2.0A ou CAN2.0B	MAC /LLC CAN 2.0A ou CAN2.0B
Couche physique	Low ou high speed CAN	Low ou high speed CAN

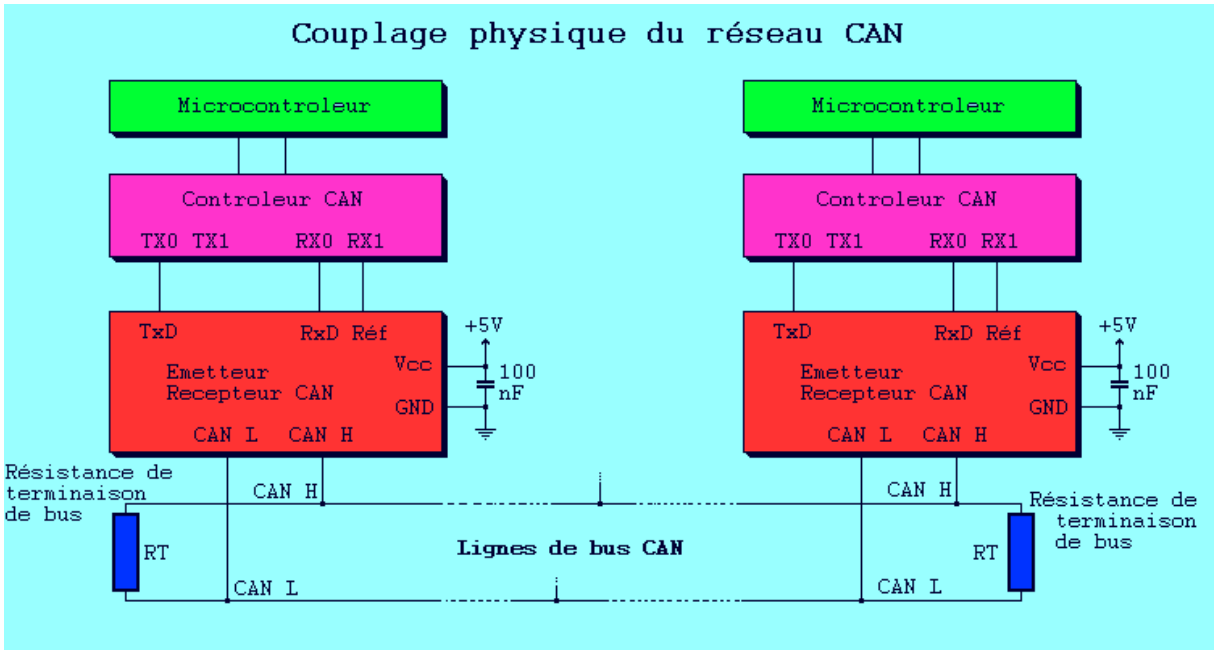
2) Support de transmission

CAN s'appuie sur une topologie de bus. Le medium de transmission principalement mis en oeuvre est la **paire torsadée** . Des solutions existent en fibre optique.

La longueur maximale est **de 40m** pour un débit de transmission de **1 Mbit/s**. Un bus CAN pent atteindre une distance de 1000 m en réduisant le débit a 50 kbits/s.

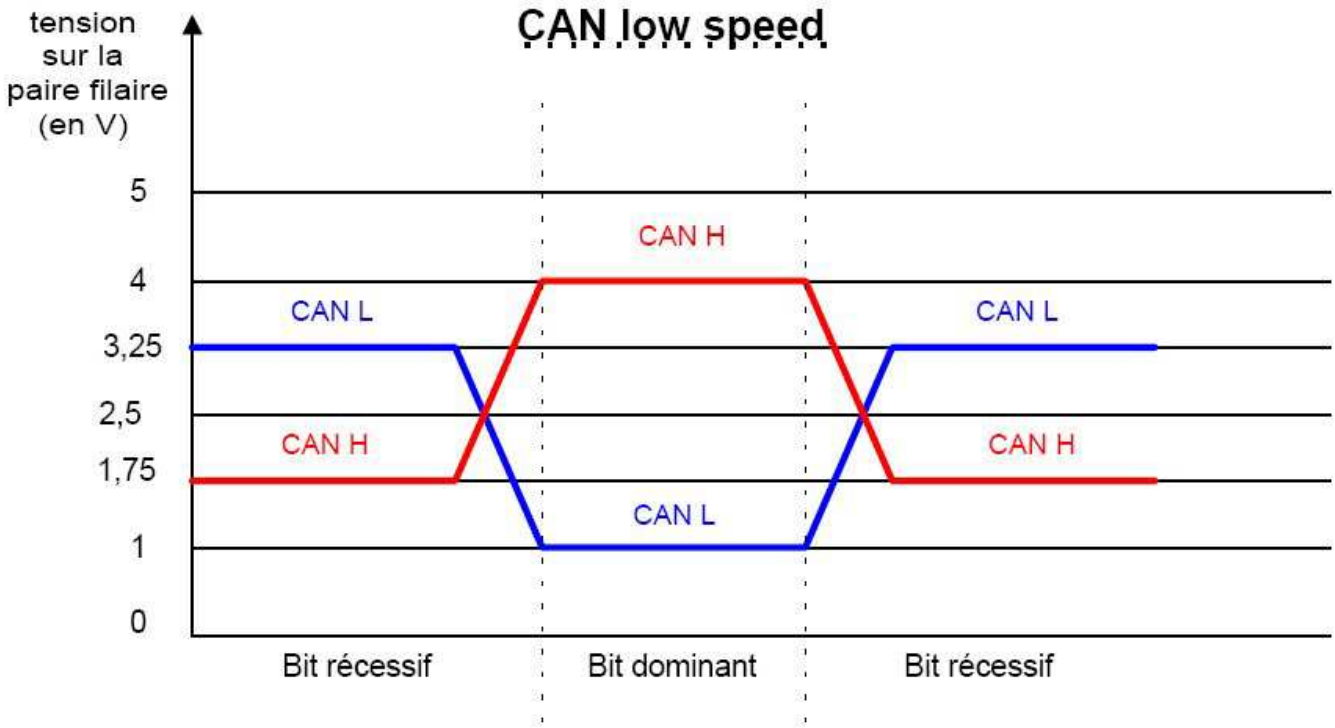


un nœud est composé le plus souvent d'un microcontrôleur, d'un contrôleur CAN et d'un émetteur récepteur CAN

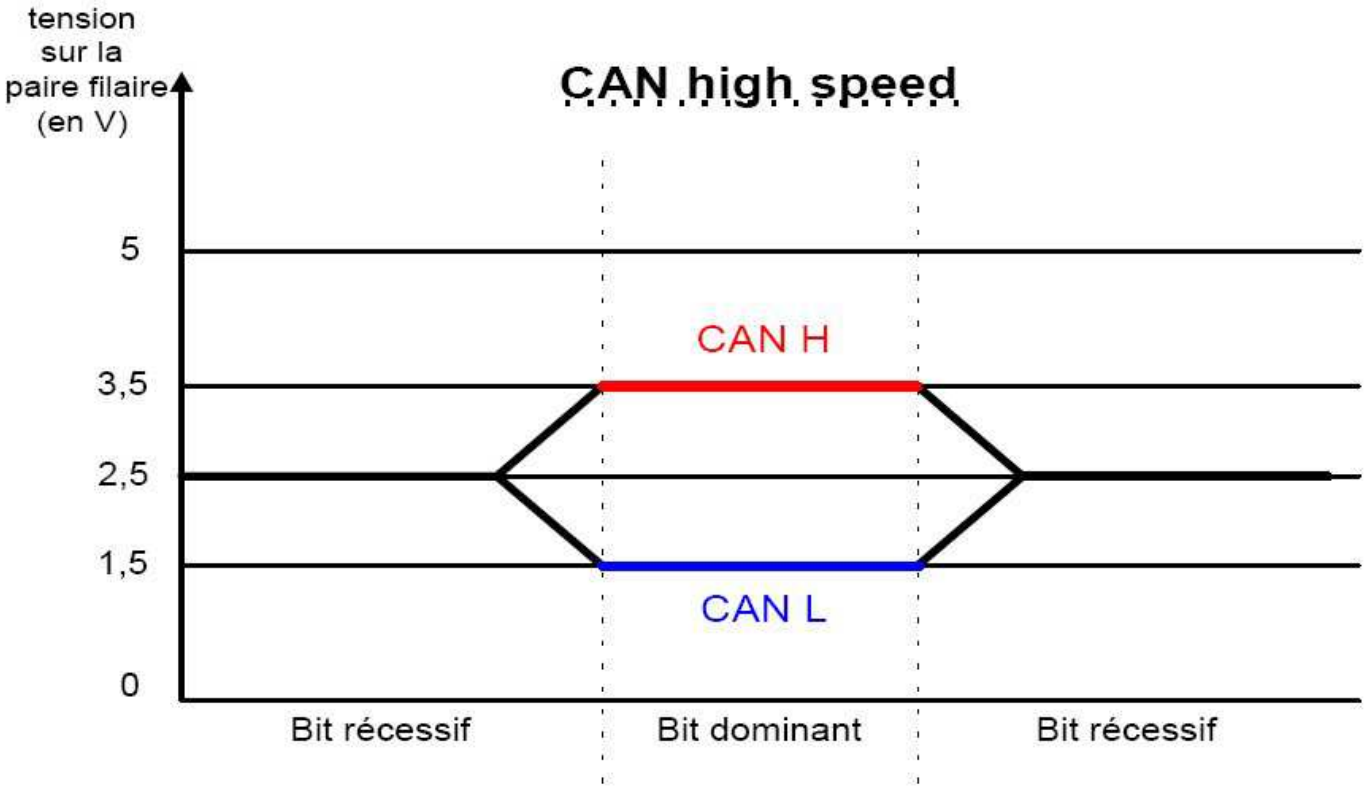


Les résistances aux deux extrémites de ligne sont nécessaire pour optimiser la qualité de la transmission (adaptation d'impédance).

codage NRZ avec des niveaux electriques différents suivant la vitesse du bus CAN (high speed, low speed).



Niveaux de tension du bus CAN *low speed*



3) Principes

3.1) principe de diffusion

Ce protocole est caractérisé par la diffusion d'informations d'un noeud vers les autres. Cette information est associée a un identificateur connu par certains noeud du reseau - le noeud dont est issue cette information : **le producteur** - les noeuds nécessitant cette information pour leur traitement : **les consommateurs**. On dit que ce protocole est orienté objet.

Ainsi une trame émise par un noeud ne contient pas véritablement d'adresses destination ou/et source. L'identificateur associé a la donnée contenue dans la trame est analysée par tous les noeuds du réseau. Ces derniers récuperent ou non cette trame.

Une trame CAN de donnée Standard a 1'allure suivantes



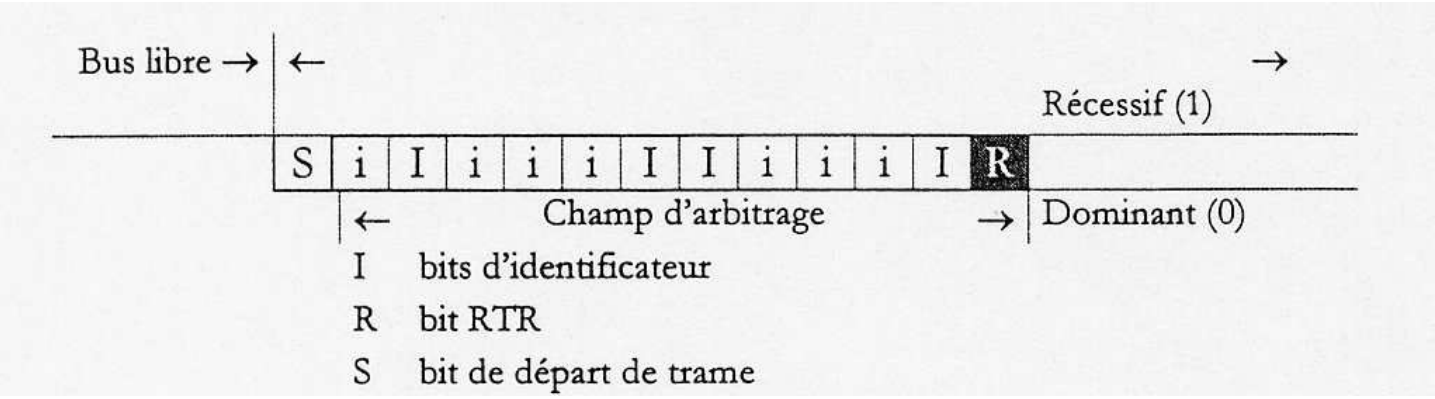
Cette trame se décompose en sept parties principales que l'on appelle des champs :

Début de trame SOF
le champ d'arbitrage
champ requête
champ de commande
champ de données
champ de CRC
champ d'acquittement
Fin de trame EOF

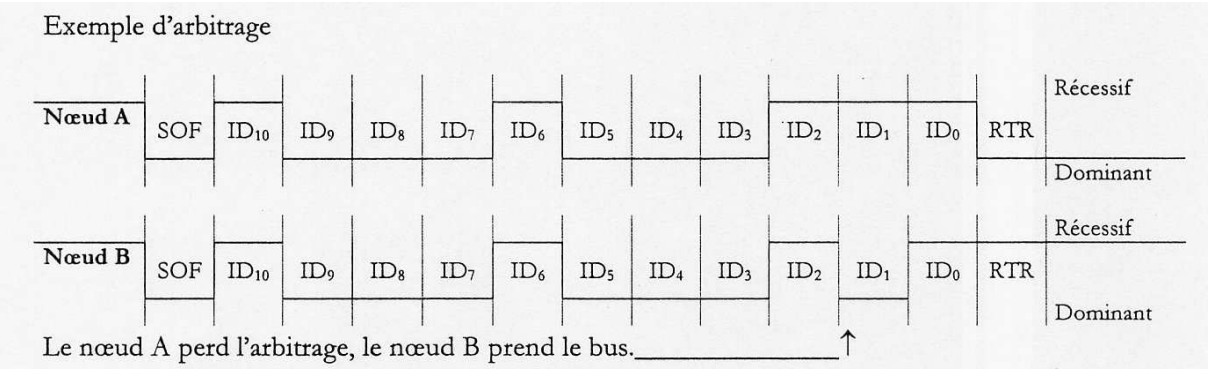
une zone 8 dite d'espace *interframe* (inter trame) qui fait partie intégrante de la trame.

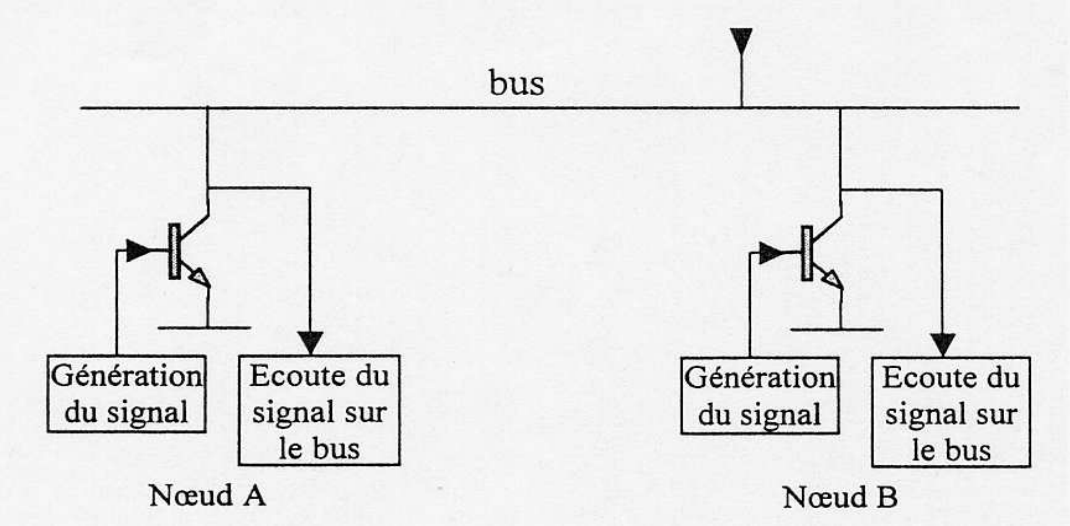
3.2) Méthode d'acces au support CSMA/CR

Les tentatives d'accès au support sont effectuées dès que-ce dernier est libre de tout signal. Le partage du support s'appuie sur 1'évaluation du champ d'arbitrage constitué des bits de l'identifier ainsi que du suivant dit *RTR (Remote Transmission request).*



Exemple d'arbitrage





Chaque noeud est constitué de deux fonctions : la génération du signal et 1'écoute du signal sur le bus. L'état du bus dépend de 1'état des transistors « ouverts » ou « fermés ».

Si tous les transistors sont ouverts, le bus est à 1'état 1 (récessif).

Si un des transistors est fermé, le bus est à 1'état 0 (dominant).

Chaque émetteur écoute le bus et bascule en réception dès qu'il détecte un bit dominant

Alors qu'il envoie un bit récessif le noeud dont 1'identificateur avait la plus haute priorité « gagne » 1'arbitrage du bus. Le noeud ayant perdu 1'arbitrage tente un accès au bus de suite après la libération de ce dernier.

Remarques sur les Identificateurs

1) La longueur de l'identificateur est de 11 bits et 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 a ID-4) ne doivent pas tous être récessifs.
2) L'objet avec la priorite la plus haute (c'est-a-dire celui dont l'identificateur est le plus faible) gagne 1'arbitrage et obtient l'accès au bus.

3.3) Trame de données

La trame de donnees transporte des données **d'un noeud source (producteur) vers les noeuds destination (consommateurs)**. Cet trame est repérée par 1'état dominant(0) du bit RTR Requête / données.

Le champ de commande (DLC) donne la longueur des données en octets.

Le champ données contient les valeurs affectées à 1'objet par le producteur. Il peut comporter de 0 à 8 octets. Le bit le plus significatif est envoyé en premier.

3.4)Trame de requète

La trame de requète est émise par un nœud A quand ce dernier a **besoin d'un certain nombre de données** issues d'un autre nœud B. Cette trame est capturée par le nœud B; ce nœud B renvoie sur le réseau les données par une trame de donnée.

La trame de requète est repérée **par 1'état récessif du bit RTR** requête /données . (Ainsi une trame de requête pour un même identificateur est moins prioritaire qu'une trame de données).

L'identificateur de cette trame est associé au nœud dont on souhaite récupérer des données.

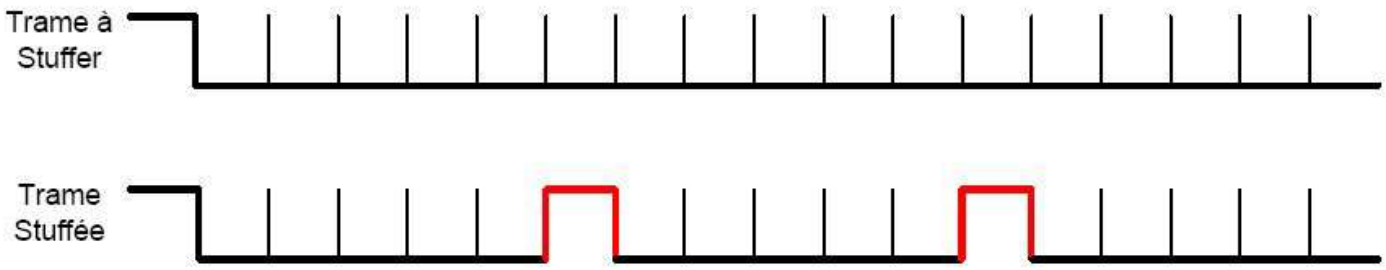
Le champ de commande **(DLC) donne la longueur des données** en octets qui devront être recues suite a la requète. Le champ données est inexistant

4) Sécurisation des transmissions

4.1) Bit stuffing

Une trame peut contenir beaucoup de bits de meme valeur; cet état peut s'apparenter a un dysfonctionnement. Pour distinguer un état normal d'un dysfonctionnement les noeuds CAN introduisent, dans la trame qu'ils émettent, après 5 bits de valeur identique, un bit supplementaire de valeur opposée. Ce bit est dit de remplissage (stuff). Un noeud recevant une trame retire ces bits. On parle de surcodage.

Les bits de stuffing permet d'insérer des transitions supplémentaires et facilite la synchronisation des bits malgré le type de codage bit NRZ.



Remarques

- seuls les champs début de trames, arbitrage, commande(DLC), données, CRC sont codés en utilisant la technique de bit stuffing.
- Le stuffing introduit une composante alternative dans le signal; ce qui facilite la transmission de ce type de signal.

4.2) Champ de CRC *Cyclic Redundancy Code* (code à redondance cyclique).

Ce champ se compose d'une séquence CRC (15 bits) qui est cloturée par le bit CRC delimiteur.

L'intégrité du message transmis est vérifiée par le récepteur au moyen d'une séquence CRC générée par 1'émetteur CAN en station avec le contenu du message transmis.

On forme un polynome à partir des champs suivant : le debut de trame, arbitrage, commande(DLC), données.

Ce polynome, multiplie par X^{15}, est divise (modulo 2) par le générateur polynomial suivant

$g(X)=X^{15}+X^{14}+X^{10}+X^{8}+X^{7}+X^{4}+X^{3}+1$

Soit, transcrit en binaire

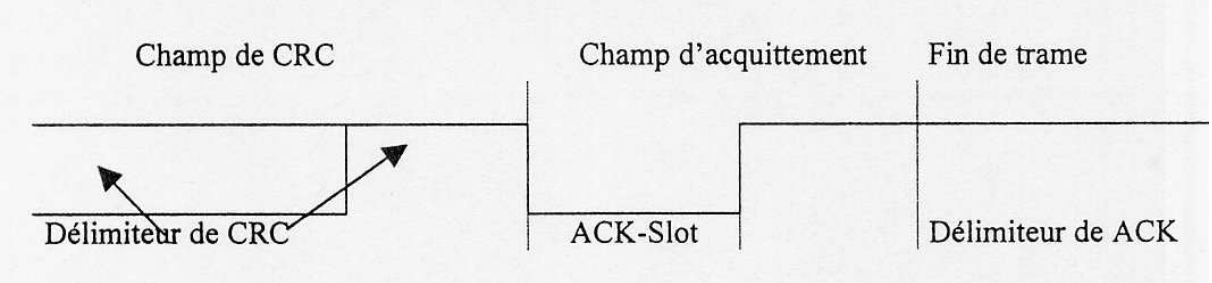
$g(X) = (110\ 0\ 0101\ 1001\ 1000)_2$ ou $(C\ 5\ 9\ 9)_{16}$

Remarques

- cinq erreurs bits independantes sont 100 % détectables a 1'intérieur du code transmis.
- La génération de la séquence CRC à 1'émission et la vérification à la réception sont réalisées sur un flots de données n'ayant pas subi de stuffing.

4.3) Champ d'acquittement

Ce champ se compose de 2 bits, les bits ACK slot et ACK delimiter.



ACK slot

un noeud ayant reçu un message valide(apres stuffing et CRC) 1'indique a 1'emetteur en superposant lors du *ACK* slot un bit dominant au bit récessif qui était présent a ce moment-là. Ce bit est un acquittement.

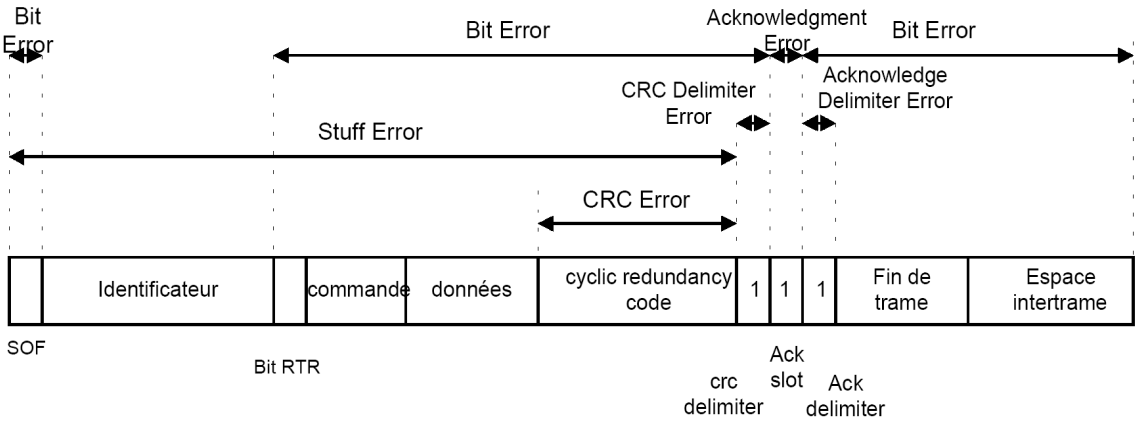
ACK delimiter

Ce deuxième bit doit toujours etre récessif, ce qui a pour conséquence que, lorsqu'un message a eté parfaitement reçu par toutes les stations présentes sur le bus, le bit de *ACK* slot (dominant) est entouré par deux bits récessifs (CRC delimiter et *ACK* delimiter).

4.4) Gestion 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 noeud, et empêcher le réseau de fonctionner correctement. Pour cela, des méthodes de détection d'erreurs de transmissions sont prévues par le protocole CAN.

a) 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.

• **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 ci-dessous résume les différents types d'erreurs et leur validité suivant l'endroit où l'on se trouve dans la trame.
Suivant le nombre d'erreur qu'un noeud comptabilise, l'état du mode de ce noeud 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.

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.

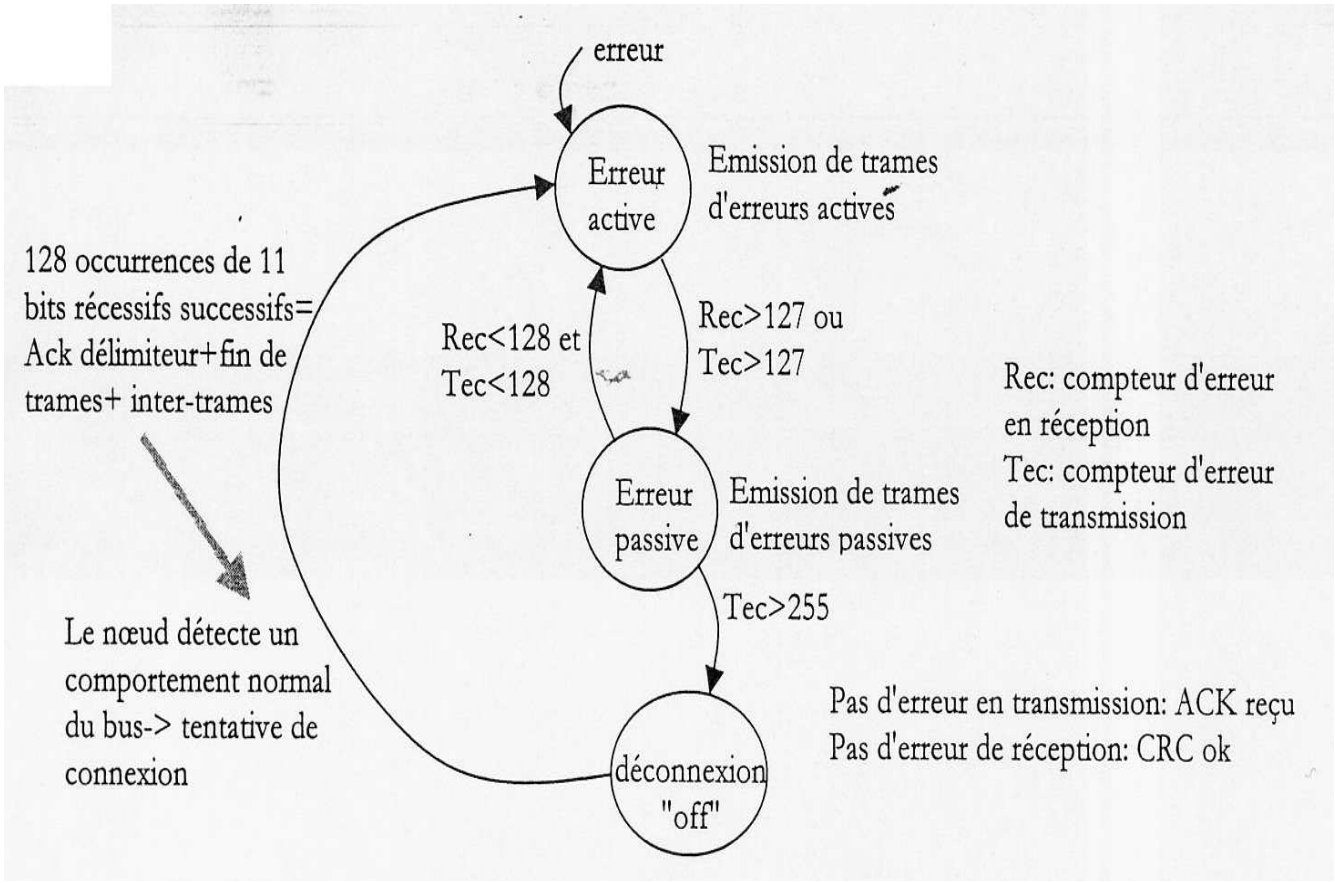
b) Les modes d'erreurs :

• **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.
Ceci signifie que le noeud en question continuera de recevoir et d'émettre normalement mais aussi que dans le cas où une erreur est détectée, il transmettra un active error flag pendant la trame d'erreur. Remarque : Il est recommandé dans le protocole de commencer à « s'affoler » au moment où le compteur atteint 96 points (warning limit), indiquant une accumulation significative de conditions d'erreurs. Généralement, à ce niveau il se produit à l'intérieur du (micro)contrôleur CAN une interruption dite de error status ou de error interrupt provoquant une requête de gestion de ces perturbations.

• **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 noeud est alors totalement déconnecté du bus mais toujours à son écoute.

Il sort de cet état de *Bus Off* après avoir reçu 127 trames de onze bits récessifs montrant ainsi que de nombreux messages dont les bits de ACKnowledge delimiter + end of frame + ceux de l'intermission sont bien passés et qu'apparemment le bus a retrouvé une bonne santé). Lorsque le nombre d'erreur devient trop important et que le gestionnaire est déjà en erreur passive, le noeud 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 ci-dessous.



c) Les trames d'erreurs

Trame d'erreur active

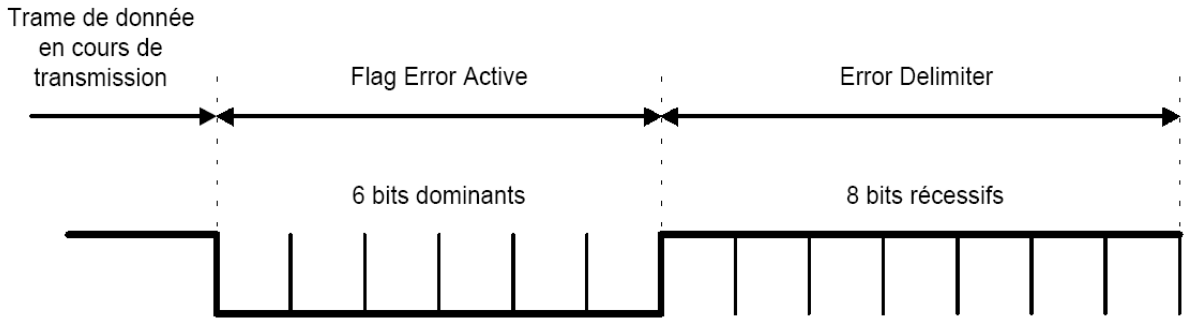
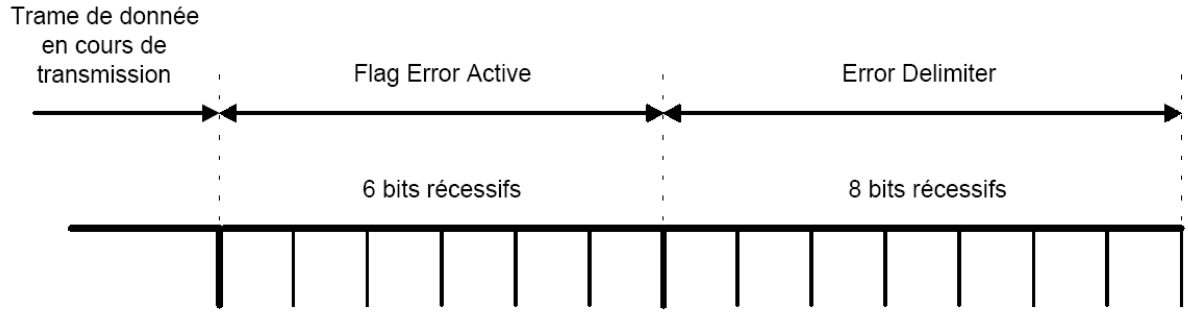


Figure 11 : Trame d'erreur active

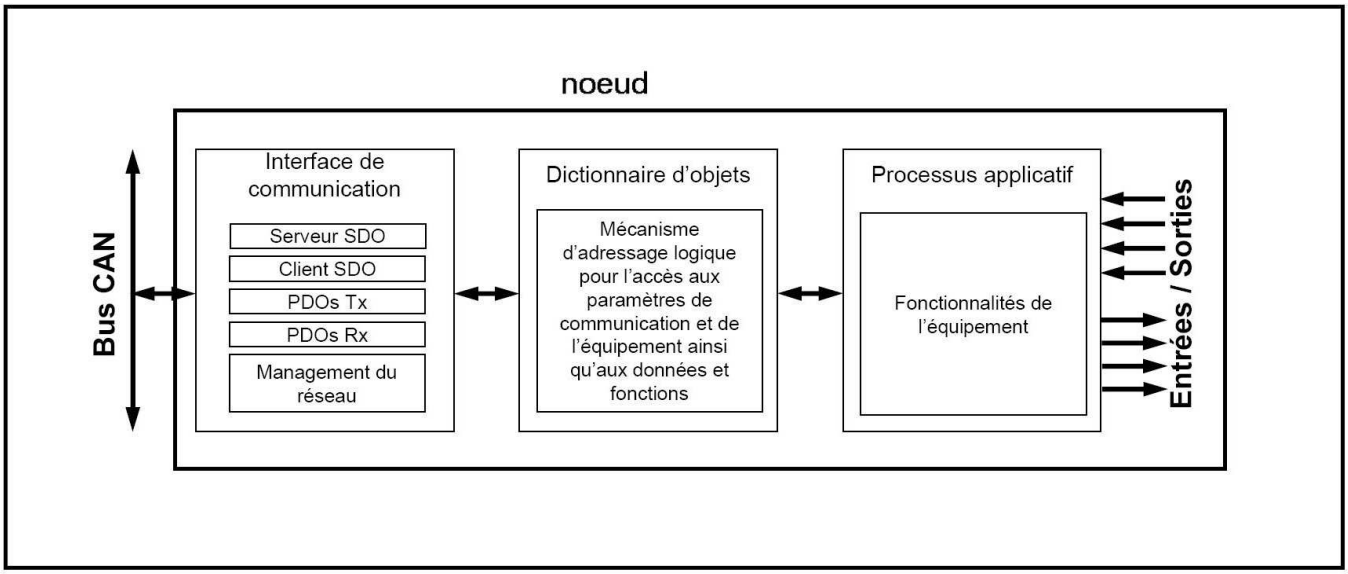
Trame d'erreur passive



Le bus CANOPEN

1) Présentation

CANOpen est une couche applicative (couche 7 du modèle OSI) pour les bus de terrain du type CAN (Controller area network) .
CANOpen est une reprise de la couche applicative CAL développée par Philips Medical Systems ; il reprend les services et protocoles de gestion de bus et de messages de la couche CAL tout en définissant le contenu des messages.
Un nœud CANOpen est constitué des éléments suivants :



Un dictionnaire d'objet définit, pour chacun des périphérique CANOpen (appelé nœud), l'information sur le format de la donnée ainsi que sur le moyen d'y accéder.

2) Le dictionnaire d'objet (Object Dictionnary)

La partie la plus importante d'un noeud Can Open est l'Object Dictionnary. Il s'agit en fait d'une table regroupant tous les objets accessibles via le bus pour ce noeud. Chaque objet du dictionnaire est adressé en utilisant un index sur 16 bits et un sous-index sur 8 bits.
Chaque entrée de cette table (suivant la valeur de son index) peut contenir :

- une variable d'un type quelconque (y compris une structure complexe), une indication de type pour chacune des variables ou des objets

- Des données du process, des paramètres de communication pour les protocoles de communication…
- Les différents types d'équipements et leurs natures…

Index OD	description	
0000h	réservée	
0001h-0FFFh	Types de données	→ Les différents types de données traitées par chaque noeud
1000h-1FFFh	Communication	→ paramètres de communication
2000h-5FFFh	Spécifique constructeur	→ Différentes informations stockées pour chaque constructeur
6000h-9FFFh	Paramètres profils equip.	→ Paramètre permettant de décrire pour chaque constructeur leur profil d'équipement
A000h-FFFFh	réservée	

La section *types de données*

Index OD	description
0000h	réservée
0001h-0FFFh	Types de données
1000h-1FFFh	Communication
2000h-5FFFh	Spécifique constructeur
6000h-9FFFh	Paramètres profils equip.
A000h-FFFFh	réservée

Index	Type standard
0001h	booléen
0002h	Integer8
0006h	Unsigned int16
0008h	Flottant 32 bits
0009h	Chaîne ASCII
000Fh	Bloc de données

Remarque: la lecture de ces entrées retourne la taille en bits

Types de données

Index	Types de données
0001h-001Fh	standard
0020h-0023h	Complexe -predéfini
0040h-005Fh	Complexe-constructeur
0060h-007Fh	Standard-profil équip.
0080h-009Fh	Complexe-profil équip
00A0h-025Fh	Equip multi-modules

0020h		PDO_COMMUNICATION_PARAM	
	00h	unsigned8	Nb. entrées
	01h	unsigned32	COB ID
	02h	unsigned8	Type transmission
	03h	unsigned16	Inhibit time
	04h	unsigned8	reservé
	05h	unsigned16	timer

0021h	PDO_MAPPING
0022h	SDO_PARAMETER
0023h	IDENTITY

La section *spécifique constructeur*

Dictionnaire

Index OD	description
0000h	réservée
0001h-0FFFh	Types de données
1000h-1FFFh	Communication
2000h-5FFFh	Spécifique constructeur
6000h-9FFFh	Paramètres profils equip.
A000h-FFFFh	réservée

- Exemple simple: le nœud CANopen comporte une horloge temps réel

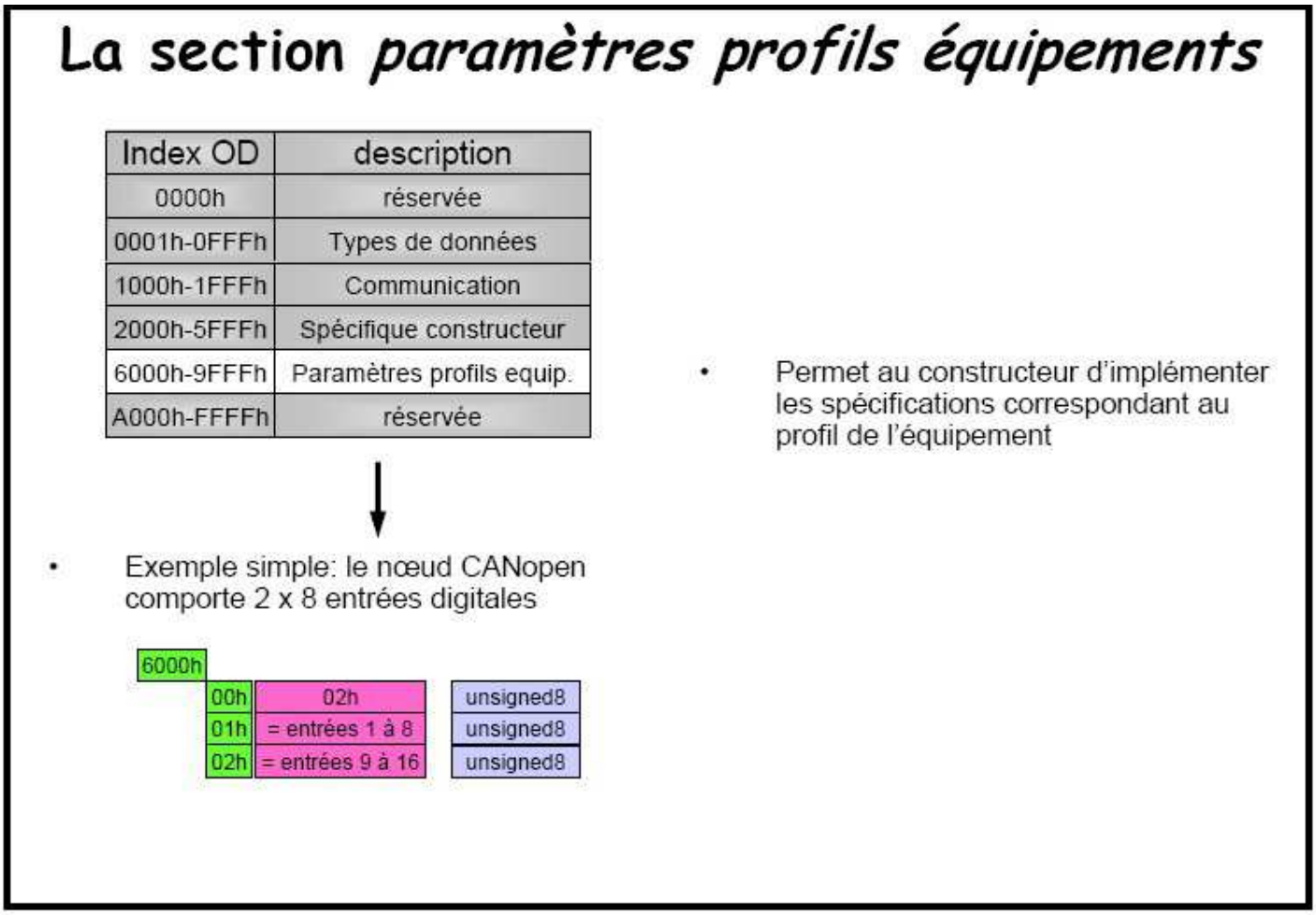
2000h			
00h	03h	unsigned8
01h	= Heure	unsigned16
02h	= Minute	unsigned8
03h	= Seconde	unsigned8

- Soit à définir un type complexe pour une série de messages d'erreurs

0040h		ERROR_MESSAGE	
00h	02h	unsigned8	Nb. entrées
01h	06h	unsigned16	Numéro de l'erreur
02h	09h	Visible_string	Message d'erreur

- Exemple de rangement de 2 messages dans la section spécifique constructeur (les entrées de cette section auront le type ERROR_MESSAGE)

2000h		
00h	02h	
01h	= n° de l'erreur	
02h	= le message correspondant	
2001h		
00h	02h	
01h	= n° de l'erreur	
02h	= le message correspondant	



3) Les protocoles de communication sur CANOPEN

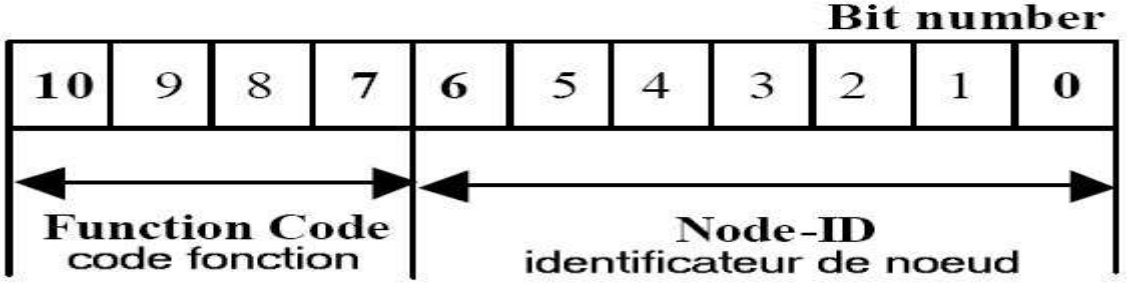
Les différents protocoles de communication permettent :

- L'accès aux objets du dictionnaire par **SDO le protocole (Service Data Object).** Ce service permet l'accès aux paramètres des périphériques (ou nœuds) présents dans le dictionnaire d'objet, la communication est alors du type question/réponse et chaque message est confirmé par une sorte d'acknowledge : Le maître accède a l'objet de l'OD grâce au couple [index,sub-index] et l'esclave répond soit en envoyant les données désirées soit en confirmant le bon déroulement de l'écriture de l'objet.
- L'Accès performant aux E/S et aux entrées de la table par le **protocole PDO (Process Data Object)**. Il est destiné au transfert de données temps réel, ce protocole permet de mettre dans un même message un ou plusieurs champs de l'OD. L'autre avantage des PDO est que la lecture se fait au travers d'un message court de type synchrone ou asynchrone qui est reçu par l'ensemble des nœuds présents sur le bus. Ce mécanisme permet de lire

une information donnée au même instant sur l'ensemble des capteurs/nœuds du bus.

- Le Management du réseau **NMT (Network ManagemenT).** Un maître NMT permet la surveillance du fonctionnement des différents noeuds d'un réseau. Ce maître NMT peut effectuer les actions appropriées de dépannage ou d'arrêt des échanges.

En CANopen, le champ d'arbitrage est décomposé de la façon suivante :



Le code fonction va définir la priorité du message sur le bus CAN. Le node ID va définir le destinataire du message (de 1 à 127)

La norme a défini pour chaque protocole de communication une priorité :

Broadcast objects of the CANopen Predefined Master/Slave Connection Set			
Object	**Function code (ID-bits 10-7)**	**COB-ID**	**Communication parameters at OD index**
NMT Module Control	0000	000h	–
SYNC	0001	080h	1005h, 1006h, 1007h
TIME STAMP	0010	100h	1012h, 1013h

Peer-to-Peer objects of the CANopen Predefined Master/Slave Connection Set			
Object	**Function code (ID-bits 10-7)**	**COB-ID ***	**Communication parameters at OD index**
EMERGENCY	0001	081h - 0FFh	1024h, 1015h
PDO 1 (transmit)	0011	181h - 1FFh	1800h
PDO 1 (receive)	0100	201h - 27Fh	1400h
PDO 2 (transmit)	0101	281h - 2FFh	1801h
PDO 2 (receive)	0110	301h - 37Fh	1401h
PDO 3 (transmit)	0111	381h - 3FFh	1802h
PDO 3 (receive)	1000	401h - 47Fh	1402h
PDO 4 (transmit)	1001	481h - 4FFh	1803h
PDO 4 (receive)	1010	501h - 57Fh	1403h
SDO (transmit/server)	1011	581h - 5FFh	1200h
SDO (receive/client)	1100	601h - 67Fh	1200h
NMT Error Control	1110	701h - 77Fh	1016h, 1017h

4) Exemples d'application du CANOPEN

