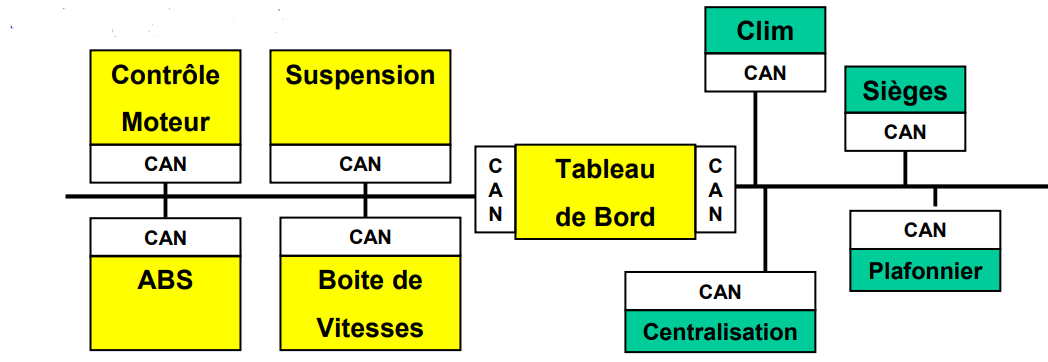
**Le bus CAN**

1. **Introduction**

Le CAN (Controller Area Network) est un bus de communication série développé à la fin des années 80 par l’entreprise allemande Robert Bosch. L’objectif était de fournir à l’industrie automobile un bus peu coûteux pour l’électronique embarquée des automobiles. Aujourd’hui, l’efficacité et la robustesse de ce protocole l’ont amené à être utilisé dans de nombreuses autres applications industrielles, en particulier celles nécessitant un débit important jusqu’à 1Mbits/s avec un très faible taux d’erreur.



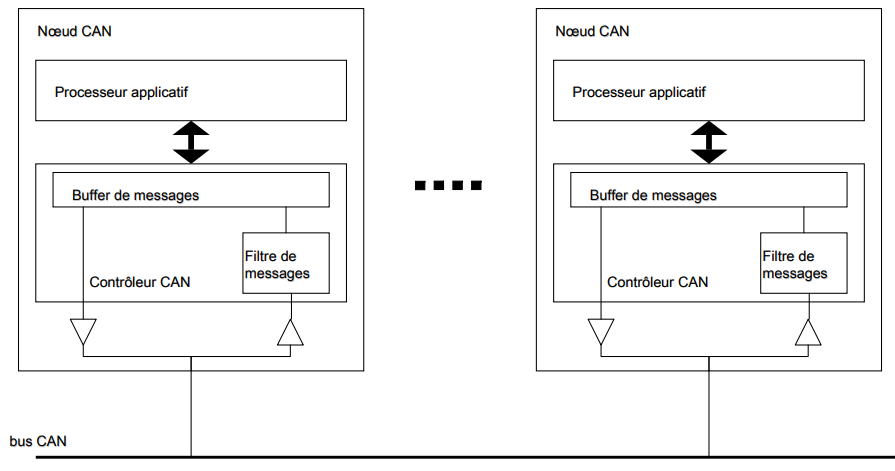
**CAN high speed CAN low speed**

1. **Domaines d’application**

* Industrial automation
* Home/Building automation
* Automotive (VL, PL)
* Transportation (Train, aviation, ...)
* Matériel agricole,
* Travaux public
* Maritime
* Médical
* Instrumentation

1. **Représentation d’un réseau CAN**

Le protocole est basé sur le principe de diffusion générale : lors de transmission, aucune station n'est adressée en particulier, mais le contenu de chaque message est explicité par une identification reçue de façon univoque par tous les abonnés. Grâce à cet identificateur, les stations, qui sont en permanence à l'écoute du réseau, reconnaissent et traitent les messages qui les concernent; elles ignorent simplement les autres. L'identificateur indique aussi la priorité du message, qui détermine l'assignation du bus lorsque plusieurs stations émettrices sont en concurrence.



1. **Normes et particularités**

Les particularités de ce bus sont :

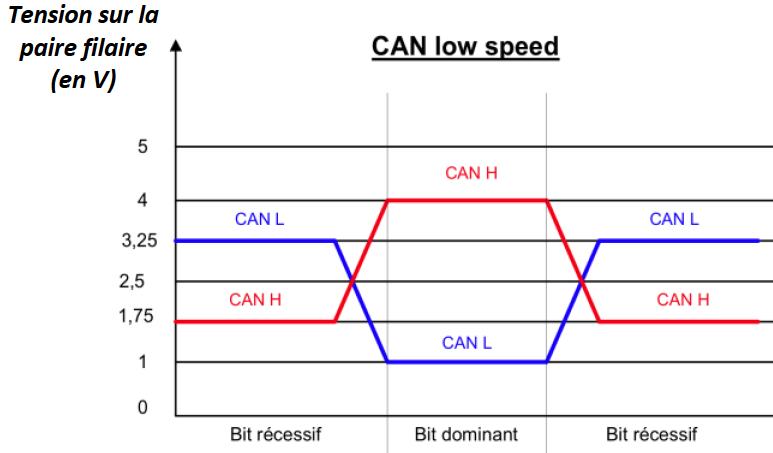
* bus multi maîtres où tous les participants ont les mêmes droits ;
* les nœuds (hôtes) de ce bus ne sont pas adressés seuls les messages transmis possèdent un identificateur. Chaque nœud scrute en permanence le bus et décide selon l’identificateur si le message lui est destiné ou pas ;
* fiabilité élevée des mécanismes de protection du protocole.

Il existe 2 protocoles CAN :

* le protocole CAN 2.0A dit CAN standard qui possède un ***identificateur de 11 bits*** ce qui permet de délivrer 2048 messages différents ;
* le protocole CAN 2.0B ou CAN étendu qui possède un ***identificateur de 29 bits*** soit plus de 536 millions de messages différents.

Pour chacun des 2 protocoles il existe 2 types d'interfaces (normes de transmission) :

* le CAN LowSpeed ou CAN-LS (norme ISO 11519). Ce bus présente un débit max de 125 kbits/s et peut raccorder jusqu’à 20 nœuds. Dans l'automobile, il est appelé bus confort et utilisé pour la climatisation, la radio, le tableau de bord … ;



CAN L

CAN H

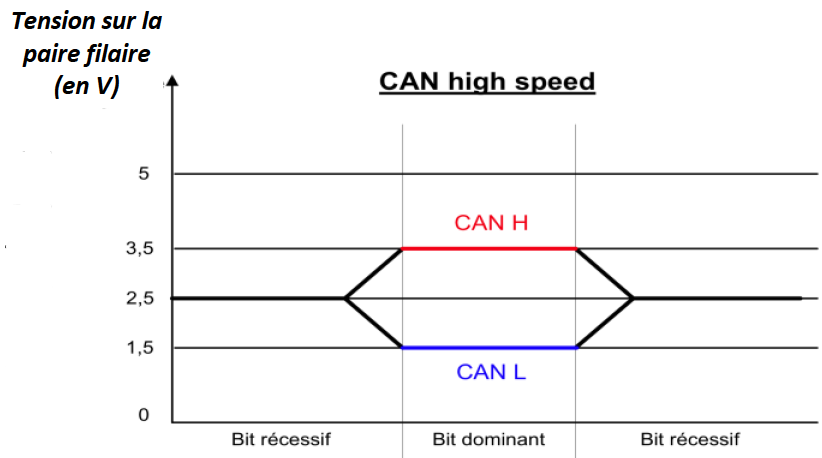
CAN L

CAN H

CAN L

CAN H

* le CAN Highspeed ou CAN-HS. Ce bus présente un débit max de 1 Mbits/s et peut raccorder jusqu’à 30 nœuds. Dans l'automobile, il est utilisé pour les équipements de sécurité (freinage, moteur … ).



1. **Codage et bit stuffing (Couche physique)**

Les bits transitant sur le bus sont codés avec la méthode NRZ (Non Return to Zero).

Le bus CAN étant une liaison asynchrone, il faut qu’un front soit régulièrement présent pour resynchroniser le récepteur. L’ajout systématique d’un bit de polarité inversée au bout de 5 bits de même polarité permet de résoudre le problème. L’émetteur ajoute donc ces bits de stuffing, le récepteur ne les considèrera pas comme de l’information utile.

Chaîne de bits à l’émission (avant “stuffing”) **0 0 1 1 1 1 1 0 0 0 0 1 1 0**

Chaîne de bits transmis sur le médium (après “stuffing”) **0 0 1 1 1 1 1 0 0 0 0 0 1 1 1 0**

Chaîne de bits à la réception (après “destuffing”) **0 0 1 1 1 1 1 0 0 0 0 1 1 0**

1. **Trames de bus CAN**

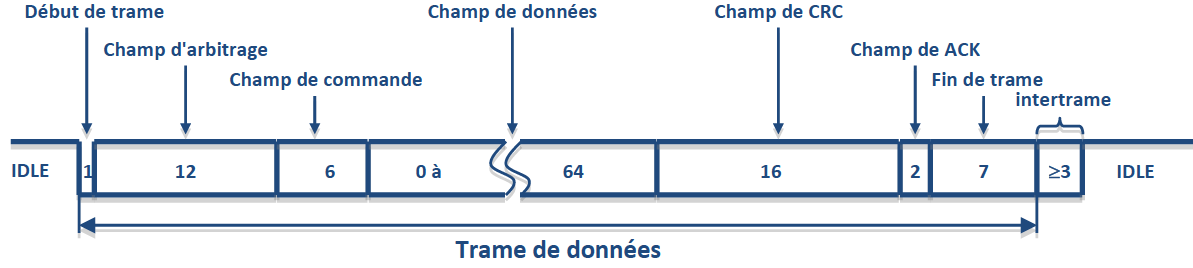
Il existe quatre types de trames pouvant être transmises sur un bus CAN :

* Trames de données : elles sont utilisées pour transporter des données (messages) de l’application (ou valeurs d’objets) sur le bus. C’est le **producteur** d’un identificateur qui émet des trames de données associées à cet identificateur.
* Trames de requête distante : elles sont utilisées par un nœud (un **consommateur**) pour demander la transmission de trames de données par d’autres nœuds (les producteurs) avec le même identificateur. Le bit RTR (Remote Transmission Request) permet de distinguer les trames de données des trames de requête. Le bit RTR est égal à 0 pour une trame de données et à 1 pour une trame de requête. On notera que les trames de données sont prioritaires par rapport aux trames de requête : quand un producteur et un consommateur d’un même objet entrent en conflit, c’est la trame émise par le producteur qui s’impose, ce qui est normal puisque la trame du producteur répond à la requête faite par le consommateur.
* Trames d’erreur : elles sont transmises par un nœud ayant détecté une erreur.
* Trame de surcharge : elles sont utilisées pour demander un délai entre deux trames de données ou de requête successives.

Les trames de données ou de requête sont séparées des trames qui les précèdent (de quelque type qu’elles soient) par un temps dit *intertrame* (ce temps doit correspondre à au moins le temps de 3 bits). Les trames d’erreur ou de surcharge ne sont pas séparées des autres trames par un intertame.

* 1. **Trame de données**

Les trames de données (Data Frames) sont composées de 7 parties détaillées ci-après. Le format est indiqué pour des trames respectant le protocole 2.0A.



* Le bit Start Of Frame (SOF) est un bit dominant. Un noeud ne peut débuter une transmission que si le bus est libre. Ensuite, tous les autres noeuds se synchronisent sur SOF du noeud ayant commencé une transmission.
* Arbitration field est constitué de l'identifieur (11 bits) et du bit RTR. Le bit RTR (Remote Transmission Request) caractérise les Remote Frames. Il est dominant dans les Data Frames et récessif dans les Remote Frames.
* Le Control field est composé de 6 bits. Les 2 premiers sont des bits réservés et les 4 suivants constituent le Data length code (DLC). La valeur du DLC est forcément comprise entre 0 et 8, soit 9 valeurs.
* Data field sont les données transmises par la Data frame. Il peut contenir de 0 à 8 octets.
* CRC field est composé de la séquence de CRC sur 15 bits suivi du CRC delimiter (1 bit récessif). La séquence de CRC (Cyclic redundancy code) permet de vérifier l'intégrité des données transmises. Les bits utilisés dans le calcul du CRC sont ceux du SOF, de l'Arbitration field, du Control field et du Data field.
* ACK field est composé de 2 bits, l'ACK Slot (niveau dominant pour positive acquittement) et le ACK Delimiter (1 bit récessif). Le noeud en train de transmettre envoie un bit récessif pour le ACK Slot. Un noeud ayant reçu correctement le message en informe le transmetteur en envoyant un bit dominant pendant le ACK Slot : il acquitte le message.
* End of frame (EOF) +interframe space : chaque Data frame et Remote frame est terminée par une séquence de 7 bits récessifs +NL1 bits (bus libre).
  1. **Trame de requête**

Une station nécessitant des données particulières peut initialiser la demande d'une transmission en envoyant une trame de requête (remote frame). La structure d'une trame de requête est identique à celle de donnée hormis le champ de données qui est optionnel dans ce type de trame.

Le bit RTR du champ d'arbitrage d'une trame de requête est un bit récessif. C'est donc ce bit qui différencie une trame de données d'une trame de requête. Alors la trame de donnée est prioritaire sur la trame de requête.

* 1. **Types d’erreurs**

Le CAN implémente cinq mécanismes de détection des erreurs, 2 au niveau bits (le ”bit monitoring” et le ”bit stuffing”) et 3 au niveau messages (vérification du CRC, de la forme des trames et de l’acquittement). Ces cinq types d’erreurs différents qui peuvent être détectée par un nœud sont:

* **Bit error**: Un nœud envoyant un bit sur le bus regarde aussi en même temps les bits qu’il reçoit (Bit monitoring). Il considère comme une erreur de bit lorsque le bit envoyé est différent du bit reçu, à l’exception de l’envoi d’un bit récessif durant l’arbitrage (cas de la perte d’arbitrage) ou pendant le ACK Slot (trame acquittée).
* **Stuff error**: Le nœud détecte une erreur de stuffing lorsqu’il reçoit 6 bits consécutifs de même valeur dans une partie d’un message qui devrait être codé avec la méthode du bit stuffing.
* **CRC error**: Une erreur de CRC est détectée lorsque le CRC calculé par un récepteur est différent de la valeur du CRC contenu dans la trame.
* **Form error**: Une ”Form error” est détectée lorsqu’un bit qui devrait être à une certaine valeur est à une valeur différente (un délimiteur par exemple).
* **ACK error** : Le transmetteur détecte une erreur d’acquittement lorsqu’il ne reçoit pas de bit dominant pendant le ACK Slot.
  1. **Gestion des erreurs dans le bus CAN**

Dans tous ces cas de figures, la présence d'erreurs sera signalée par une **trame d'erreur** - *error frame*- qui sera générée sur le bus pour en informer qui de droit.

Lors de **perturbations persistantes**, la partie contrôleur de bus passe en mode dite de ***bus off***et la partie CPU locale peut alors prendre des valeurs convenues par défaut.

La présence de perturbations mineures (ou de courtes durées) sur le bus n'affecte pas la partie contrôleur de bus.

Tous les microcontrôleurs conformes au protocole CAN doivent obligatoirement posséder deux compteurs internes bien distincts:

* le transmit error counter (TEC),
* le receive error counter (REC),

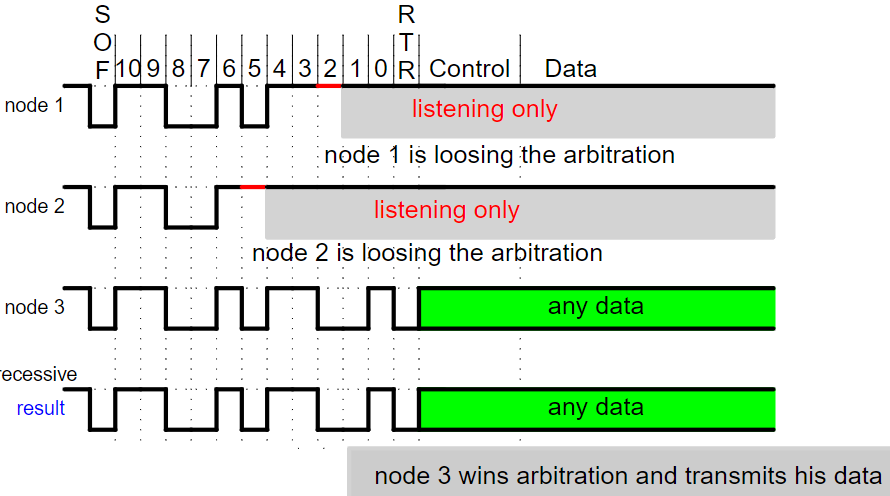
TEC<128 et REC<128 : état error active.

TEC>127 et REC>127 : état error passif.

TEC>255 : Bus off.

**6.5. Arbitrage**

CSMA CD/AMP (Carrier Sense Multiple Acces with Collision Detection and Arbitration Message Priority). Avant d’Emettre sur la ligne, le nœud écoute et vérifie qu’aucun transfert n’est en court. Si c’est le cas il attend. Lorsque le bus est libre, plusieurs nœuds peuvent vouloir émettre. Tous ces nœuds commencent à placer l’identifiant sur le bus. Le niveau 0 étant dominant, si un nœud tente de forcer le niveau 1 sur la ligne, mais qu’il entend 0, c’est qu’il a perdu l’arbitrage. Le nœud envoyant un message avec l’identifiant le plus petit gagnera donc le droit d’émettre.



**6.6. Exemple de circuit d’interface de ligne (SN65HVD251) :**

