A2V - 2002 - Le CANOpen [Revenir à l'accueil]

Le réseau CAN et le protocole CAN Open

Nos Variateurs HARMONICA, BASSOON, CELLO, sur CANOpen DS301 /402 respectent tous les principes décrits ci-dessous.

I – Les bus de terrain

I - 1. Généralités sur les réseaux de terrain

Les réseaux de terrain permettent l'interconnexion entre plusieurs entités d'un même système. Cette communication se déroule sur une zone limitée et sur laquelle on désire une réduction maximale de la longueurs des liaisons entre les différents éléments grâce à un médium commun de transmission. Cette réduction est liée à une volonté de sécurité -sur des automates d'usine par exemple-, ainsi qu'un gain de place sur un terrain comme une automobile pour les systèmes dit d'électronique embarquée. Dans les industries lourdes comme dans les moyens de transport ces réseaux sont à l'origine de l'amélioration constante de tous les systèmes.

De nos jours les réseaux de terrain sont implantés dans tous les domaines de l'industrie : grâce à leur flexibilité d'extension et de raccordement de modules sur la même ligne tout d'abord, mais aussi en raison de l'assurance d'un transport fiable de données de n'importe quel élément informatisé vers un autre.

L'utilisateur ne se soucie pas du chemin suivi par les informations, de la conversion des formats, du type de l'interlocuteur ou bien du type du constructeur d'un élément, les techniques de réseaux facilitent grandement l'insertion ou la suppression d'éléments au sein d'un même système.

Un bus de terrain permet de transférer dans la plupart des cas les informations de manière séquentielle (c'est à dire bit par bit) ou bien par paquet de bits. Il faut savoir que le bus de terrain permet un échange de données qui serait difficile voire impossible par un autre moyen.

Derrière ce concept technologique se cachent de réels protocoles de communication qui n'ont fait qu'évoluer depuis maintenant près de quinze ans.

Basés sur l'optimisation de place et de temps, on a vu apparaître des contrôles-commandes de plus en plus perfectionnés. Ces technologies ne cessent d'être améliorées et sont de plus en plus utilisées pour des raisons de coûts, de fiabilité et de confort tant en ce qui concerne leur installation que leur entretien.

I - 2. Le système OSI

Afin de normaliser les protocoles, l'International Standard Organisation a développé le modèle Open System Interconnections, qui permit d'identifier et de séparer les différentes fonctions d'un système de communication.

Ce modèle divise en sept couches les fonctions d'un système de communication. Cependant il n'est pas indispensable de disposer de toutes les couches dans un système : selon les

fonctionnalités requises, certaines couches intermédiaires sont inutiles.

Numéro de la couche	Nom de la couche
7	Application
6	Présentation
5	Session
4	Transport
3	Réseau
2	Liaison de données
1	Physique

Nota : Les couches 1 à 3 fonctionnent entre machines ou éléments qui se suivent tandis que les couches 4 à 7 fonctionnent entre les machines aux extrémités.

I – 2 – 1. La couche physique

Rôle : Permet la transmission d'informations sous forme binaire.

On lui associe les notions de directionnalité (mono- ou bi-), de temps de propagation, de valeurs pour l'état haut, l'état bas.

I – 2 – 2. La couche de liaison de données

Rôle : Définit le format ainsi que le codage logique de la trame. Elle permet également la sécurisation du lien physique.

Cette couche découpe les séquences de bits transmis via la couche physique sous forme de trames dont la taille varie (de 10 à 1000 octets).

Ces trames sont protégées par un code détecteur d'erreur ainsi que par des trames dites d'acquittement. En effet si la trame reçue présente une anomalie quelconque, elle ne sera pas acquittée. La trame sera alors retransmise.

I – 2 – 3. La couche réseau

Rôle : Permet l'acheminement et le contrôle des données.

Les chemins peuvent être prédéfinis dans des tables de routage au sein même du hardware de certaines machines, mais le plus souvent ces chemins sont choisis dynamiquement pour chaque paquet de données.

Dans la technologie des réseaux on confie également à cette couche le rôle de contrôler les encombrements possibles au niveau des échanges des données.

Chaque réseau possède son propre protocole, ainsi lorsque l'on passe d'un réseau à un autre la couche réseau permet l'homogénéisation entre ces différents réseaux.

Notons que la couche réseau peut tout à fait demeurer absente dans certains protocoles comme celui des réseaux à diffusion.

I – 2 – 4. La couche transport

Rôle : Permet la sécurisation du lien.

Elle permet de sécuriser le lien de bout en bout. Cette couche est proche de la couche de liaisons de données, or cette dernière ne fonctionne que entre deux machines connectées aux extrémités du même lien physique (terminal et routeur, deux routeurs).

I – 2 – 5. La couche session

Rôle : Permet d'établir une session entre deux machines.

On associe à la couche session le fait que les machines peuvent désormais dialoguer et se synchroniser. Le dialogue demeure nécessaire pour certaines applications, ainsi à travers cette notion se développe la principale différence qu'il peut exister entre les différents protocoles.

I – 2 – 6. La couche présentation

Rôle : Permet de gérer la syntaxe et la sémantique de l'information transmise.

L'information transmise sous forme d'octets peut être de l'ASCII ou bien des résultats de calculs possédant un format spécial (virgule fixe, flottante) …

La couche présentation permet de coder cette information correctement.

I – 2 – 7. La couche application

A chaque application correspond son protocole comme par exemple :
FTP pour le transfert de fichiers
SMTP pour le transfert de courrier électronique

I – 3. Le temps réel

Le temps réel est une notion des plus difficiles à définir. En effet, en l'absence de définition normalisée, cette notion est souvent mal utilisée et soulève bon nombre de polémiques.

Nous ne voulions pas donner une énième définition, c'est pourquoi nous adopterons la définition de l'informaticien KAISER :

« On dit qu'il y a traitement temps réel lorsque le temps de réponse à des interrogations, généralement aléatoires, est soumis à des contraintes du système »

Ce qui ressort des définitions qui ont pu être écrites sur ce sujet est la notion de **synchronisation** entre le traitement de l'information et la génération issue du monde extérieur ; En effet ce qui prête souvent à confusion est le temps de réponse de la machine par rapport au temps effectif qui s'écoule pour l'environnement. Naturellement, on pense à des ordres de grandeur infinitésimale de l'ordre du millième de seconde. Ceci est en partie vrai, si l'on prend un système radar par exemple, mais la réponse peut être de l'ordre de centaines de minutes pour des systèmes des régulations ou de contrôle de réactions chimiques qui sont elles-même tributaires des conditions des températures et de pression.

On distingue globalement deux types de situations :

- Le système transactionnel où l'on tolère le dépassement d'un temps de réponse donné sur quelques échantillons. En d'autres termes la violation de la contrainte de temps n'entraîne pas de défaillance du système à condition tout de même qu'elle se produise avec une probabilité bornée. C'est le temps réel mou.
- La commande de processus où le respect d'un temps de réponse donné doit être garanti dans tous les cas sous peine de voir apparaître une dégradation, voire même un effondrement du système, c'est le temps réel dur.

On fait alors intervenir la notion de **déterminisme**.

« Un système est déterministe quand le comportement des sorties de celui ci est parfaitement maîtrisé et ce quelles que soient ses entrées »

On parlera de *déterminisme temporel* afin de parler du respect du timing, et du *déterminisme évènementiel* lorsque tous les évènements sont traités.

Découlent de cette notion plusieurs autres qui affinent le concept :

- La prévisibilité montre les possibilités que l'on a de prévoir comment le système va se comporter quel que soient les circonstances.
- L'urgence : il s'instaure une hiérarchie entre les différents traitements à effectuer ; certains étant plus importants que d'autres.

Le déterminisme est assuré par un certain nombre d'outils dont la préemption.

Afin de bien comprendre la notion de temps réel, il est important de la différencier avec celle de « multitâche ». Ces deux notions n'ont en commun qu'un nombre restreint de fonctionnalités comme l'exécution concurrente des tâches, la synchronisation et la communication en exclusion mutuelle entre elles. Le but recherché dans le multitâche est l'optimisation du temps de travail du processeur, alors que dans le temps réel, il s'agit du déterminisme des actions.

II – Le bus CAN

II – 1. Généralités et applications

A l'origine le CAN fut développé pour l'usage automobile par Bosch et aujourd'hui la plupart des constructeurs mettent au point des systèmes entièrement multiplexés utilisant la technologie CAN.

La technologie CAN trouve sa place dans de nombreuses industries notamment grâce à ses qualités de fiabilité et d'architecture temps réel. Toute transmission des données à travers le bus est bornée grâce à un temps maximal normalisé. De plus l'architecture CAN possède un système d'erreur simple et efficace qui permet de l'utiliser à des fins médicales ainsi que dans le cadre de toute application mettant en jeu la sécurité des personnes.

Les contrôleurs CAN sont physiquement petits, peu coûteux et entièrement intégrés. Ils sont utilisables à des débits importants, en temps réel et dans des environnements difficiles. Enfin, les transmissions ont un haut niveau de fiabilité. C'est pourquoi ils ont été utilisés dans d'autres industries que l'automobile et des applications utilisant le CAN sont aujourd'hui disponibles dans l'agriculture, la marine, le matériel médical, les machines textiles, etc…

II – 1 – 1. CAN et l'automobile

Pour satisfaire les exigences de plus en plus importantes du client en matière de sécurité et de confort, et pour se conformer aux lois de réduction de la pollution et de la consommation de plus en plus drastiques, l'industrie automobile a développé de nombreux systèmes électroniques tels que le système anti-patinage, le contrôle électronique du moteur, l'air climatisé, la fermeture centralisée des portes, etc...

La complexité de ces systèmes et la nécessité d'échanger des données nécessitent de plus en plus de câbles. En plus du coût très important de ce type de câblage, la place qui lui est nécessaire peut le rendre tout simplement impossible. Enfin, le nombre croissant de connections et de câbles pose de sérieux problèmes de fiabilité et de réparation.

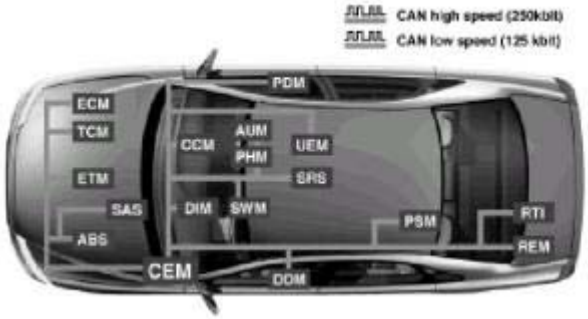
Bosch, un important équipementier automobile, a fourni la solution dans le milieu des années 80 avec le bus CAN. L'entreprise allemande a défini le protocole et a autorisé de nombreux autres fabricants à développer des composants compatibles CAN.

Avec le protocole CAN, les contrôleurs, capteurs et actionneurs communiquent entre eux sur deux câbles à une vitesse pouvant aller jusqu'à 1 Mbits/s.

Contrôle du moteur :

L'architecture CAN a été utilisée pour la première fois dans une automobile chez le constructeur Mercedes Benz qui reliait ainsi plusieurs capteurs entre eux (température d'huile moteur, position du vilebrequin, détecteur de cliquetis, injection…). Le procédé a depuis été repris par de nombreux constructeurs tels que Renault, Saab, Audi, Peugeot ou Volkswagen.

Lorsque sont reliés entre eux de tels éléments de contrôle du moteur ou de sécurité (comme les capteurs d'ABS), le débit de transfert des données est généralement de 500 kbit/s.



Eléments de confort :

Le CAN utilisé dans une automobile permet de relier facilement plusieurs modules de divertissement et de confort comme la radio, le système de navigation GPS, le téléphone ou encore la climatisation.

Ces éléments peuvent également être interconnectés avec les modules de sécurité (par exemple le déclenchement des Airbags - coussins gonflables - qui provoque un appel téléphonique SOS envoyant ainsi aux secours via le GPS la position de l'automobile accidentée).

II – 2. La couche physique

Cette partie concerne les aspects physiques de la liaison entre les nœuds connectés sur un bus CAN.

La couche MAC (Médium Access Control), qui est une couche intermédiaire entre la liaison de données et la couche physique, définit l'arbitrage des bits sur le bus et donne à telle ou telle trame sa priorité.

II – 2 – 1. Le NRZ

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

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

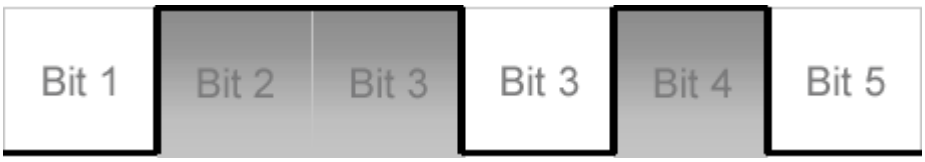


Figure 1 : transmission de le trame 011010

II – 2 – 2. Le bit stuffing

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

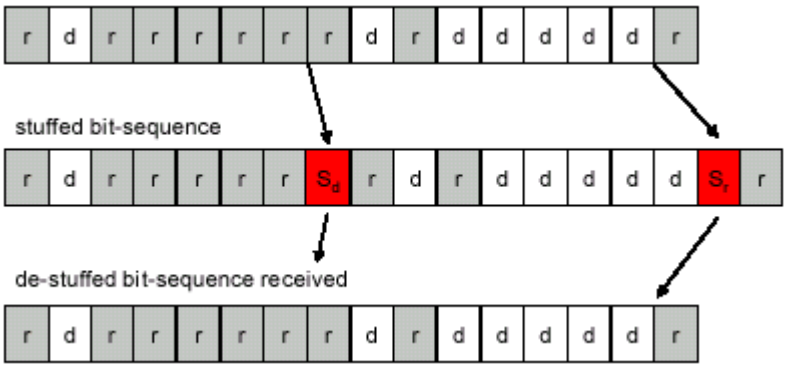


Figure 2 : Trame stuffée puis déstuffée

Le message comporte donc plus de transmissions ce qui permet une meilleure synchronisation des nœuds. Cette technique est appliquée pour les Data Frames et les Remote Frames, sur les bits depuis le Start of Frame jusqu'à la séquence de CRC.Les formats de trame seront détaillés un peu plus loin dans le rapport.

Nota : cette technique doit être appliquée à l'émission et à la réception pour un fonctionnement correct du réseau.

II – 2 – 3. Le bit timing

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

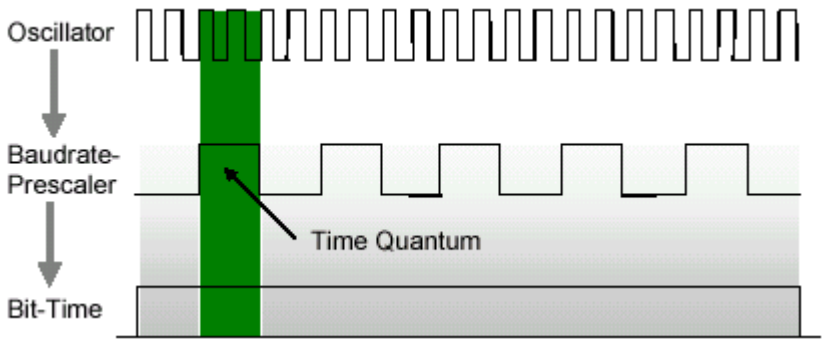


Figure 3 :Le bit timing

Nota : les spécifications de bit timing relatives au protocole CAN Open seront détaillées dans le chapitre III.

II – 2 – 4. Longueur du bus

La longueur du bus dépend des paramètres suivants :

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

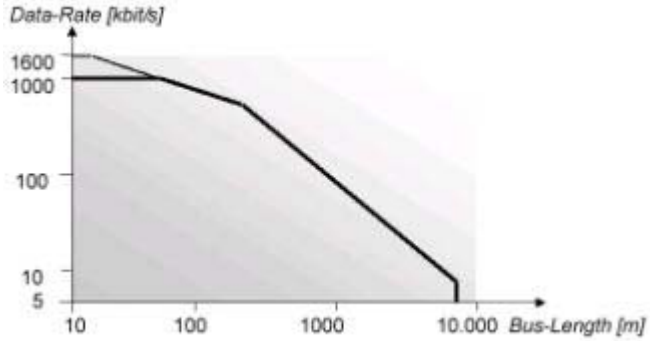


Figure 4 : Ratio Longueur/Vitesse

Bit Rate	Bus Length	Nominal Bit-Time
1 Mbit/s	30 m	1 µs
800 kbit/s	50 m	1,25 µs
500 kbit/s	100 m	2 µs
250 kbit/s	250 m	4 µs
125 kbit/s	500 m	8 µs
62,5 kbit/s	1000 m	20 µs
20 kbit/s	2500 m	50 µs
10 kbit/s	5000 m	100 µs

Figure 5 : Vitesse – Longueur – Bit time

Il est important de noter que n'importe quel module connecté sur un bus CAN doit pouvoir supporter un débit d'au moins 20 kbit/s

Pour une longueur de bus supérieure à 200 mètres il est nécessaire d'utiliser un optocoupleur, et pour une longueur de bus supérieure à 1 kilomètre il est nécessaire d'utiliser des systèmes d'interconnexion tels que des répéteurs ou des ponts.

II – 2 – 5. Le principal support de transmission et la norme ISO 11898-2

De nombreux standards industriels fonctionnent avec un bus CAN, et la normalisation première du CAN est la norme ISO 11898-2 spécifiant les caractéristiques du bus CAN High Speed.

On note que le CAN peut très bien utiliser la fibre optique ou la transmission hertzienne comme support de transmission mais ceci ne fait pas l'objet de notre étude.

La technologie de câblage du réseau CAN se rapproche de la structure simple ligne de manière à minimiser les effets de réflexion. La transmission des données s'effectue sur une paire par émission différentielle c'est à dire que l'on mesure la différence de tension entre les deux lignes (CAN H et CAN L). La ligne du bus doit se terminer par des résistances de 120 W (minimum 108W, maximum 132W) à chacun des bouts.

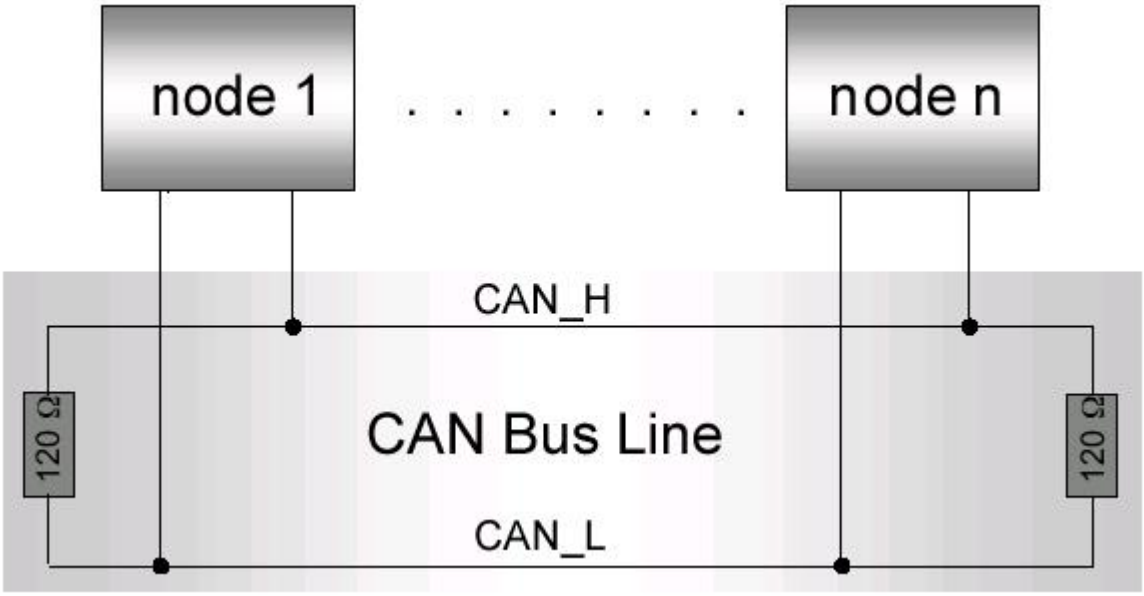


Figure 6 : Le câblage du réseau CAN

Un nœud du bus CAN requiert pour son fonctionnement au sein du réseau un microcontrôleur et un contrôleur CAN.Il est connecté à l'organe de transmission qui transforme les bits en tensions (le Transceiver), par l'intermédiaire d'une ligne de transmission Tx et d'une ligne de réception Rx. Les tension de référence d'alimentation du Transceiver sont de 5 volts.

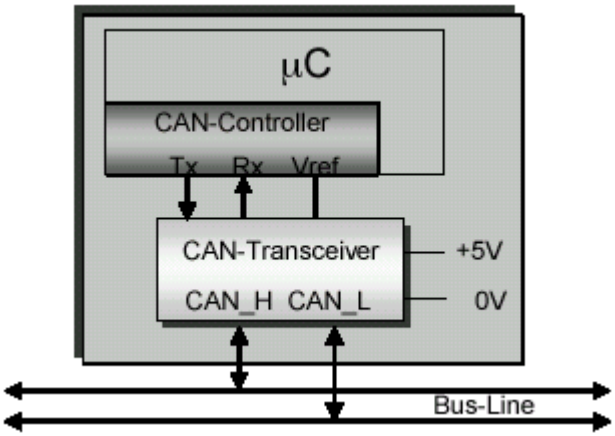


Figure 7 : Un nœud sur le bus CAN

Par défaut, c'est à dire sans transmission, la ligne CAN H est à 3.5 volts et la ligne CAN L est à 1.5 volt.

CAN H	3,5 V
CAN L	1,5 V

Les niveaux de transmission du bus sur les lignes physiques sont comprises entre 0 et 5 volts, mais afin de garantir la transmission d'un bit dominant (0) ou d'un bit récessif (1), la norme spécifie le codage des niveaux de tensions suivants :

Nature du bit	Codage de la tension
1	*CAN H<CAN L+0.5 V*
0	*CAN H>CAN L+0.5V*

Nota : Nous verrons par la suite pourquoi le 0 est le bit dominant et 1 le bit récessif.

II – 2 – 6. L'immunité aux interférences électromagnétiques

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

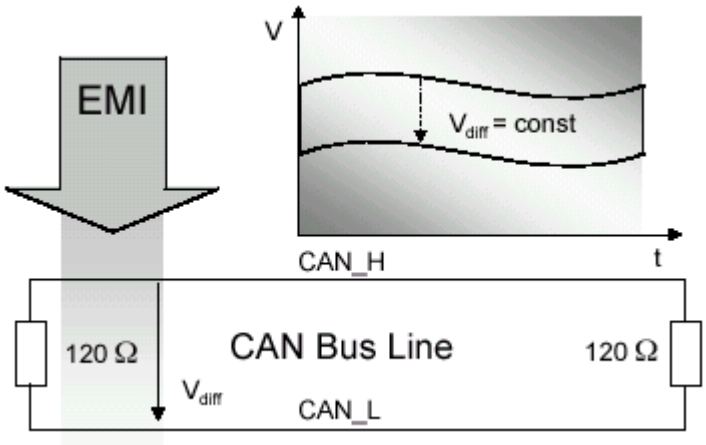


Figure 8 : L'immunité aux interférences électromagnétiques

II – 3. La communication sur le bus : l'architecture de liaison de données

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

Ce concept peut être schématisé par celui de la radio diffusion d'information routière: une fois l'état du trafic connu, un conducteur peut décider de changer son trajet, d'arrêter son véhicule, d'alerter un autre conducteur, ou de ne rien faire…

Le protocole CAN autorise différents nœuds à accéder simultanément au bus. C'est un procédé rapide et fiable d'arbitrage qui détermine le nœud qui émet en premier.

L'accès au bus est donc aléatoire car un nœud peut émettre à n'importe quel moment. Mais cet accès se fait par priorité ; cette méthode est appelée CSMA CD/AMP (Carrier Sense Multiple Acces with Collision Detection and Arbitration Message Priority).

II – 3 – 1. La méthode d'arbitrage

Afin d'être traitées en temps réel, les données doivent être transmises rapidement. Cela suppose non seulement une voie physique de transmission pouvant atteindre jusqu'à 1 Mbit/s, mais exige également une assignation rapide du bus dans les cas de conflits, lorsque plusieurs stations souhaitent transmettre simultanément des messages.

Lors de l'échange de données sur le bus, une hiérarchie est établie selon le type d'information. Par exemple une valeur variant rapidement, comme l'état d'un capteur ou l'asservissement d'un moteur, doit être transmise plus souvent avec un retard moindre que d'autres valeurs comme la température du moteur, qui évolue lentement. Sur le réseau CAN, l'identificateur de chaque message, qui est un mot de 11 bits (version 2.0 A) dans le cadre de notre application, détermine sa priorité.

Can Open va donc attribuer à chaque échange de données une priorité définie par le COB-ID qui se positionne en début de trame.

Le niveau de priorité est donné par l'ID sur 7 bits pour la version 2.0 A, ce qui donne 127

niveaux le 128ième étant le niveau 0000000, celui du NMT (Network Management) qui sera détaillé plus loin.

Les 4 bits du champ COB définiront le type de l'objet de communication de la trame.

Le procédé d'attribution du bus est basé sur le principe de l' «arbitrage bit à bit », selon lequel les nœuds en compétition, émettant simultanément sur le bus, comparent bit à bit l'identificateur de leur message avec celui des messages concurrents. Les stations de priorité moins élevée perdront la compétition face à celle qui a la priorité la plus élevé.

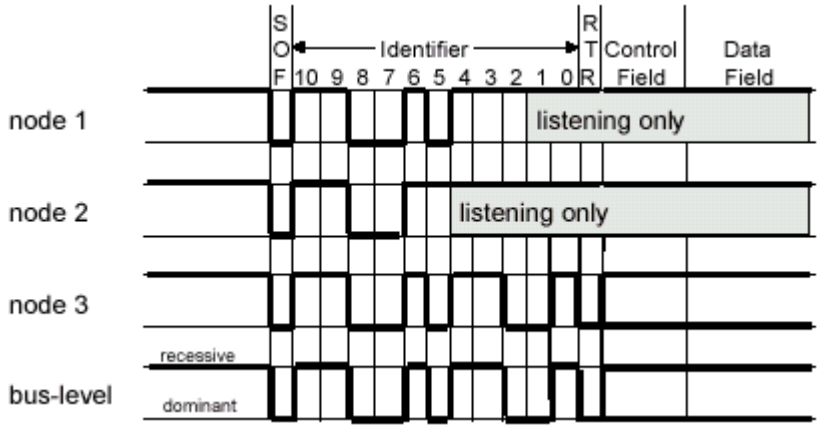


Figure 9 : L'arbitrage bit à bit

II – 3 – 2. Le format de la trame DATA FRAME (trame de données)

Le format d'une trame est le suivant :

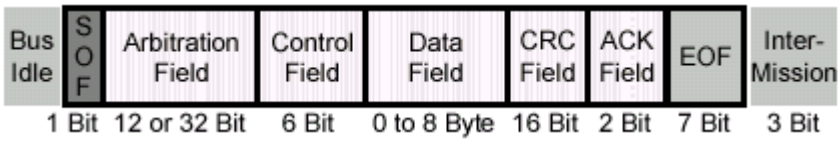


Figure 10 : Format de la trame

Détaillons maintenant les différents champs de la trame de données.

Start Of Frame

Le bit Start Of Frame (SOF) indique le début d'une Data Frame ou d'une Remote Frame. C'est un unique bit dominant.

Un nœud ne peut bien sûr débuter une transmission que si le bus est libre (cf. technique d'arbitrage dans le chapitre suivant). Ensuite, tous les autres nœuds se synchronisent sur le SOF du nœud ayant commencé une transmission.

Arbitration Field

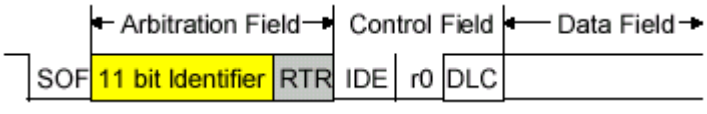


Figure 11 : Le champ d'arbitrage

L'Arbitration Field est constitué du COB, de l'identifier (ID) et du bit RTR.

Le COB-ID permet d'identifier le message et est utilisé lors de l'arbitrage. L'ID comporte ici 7 bits puisque nous utilisons la version 2.0A du protocole de communication sur le bus CAN.

Le bit RTR (Remote Transmission Request) caractérise les Remote Frames. Il est dominant dans les Data Frames et récessif dans les Remote Frames (ces dernières seront détaillées dans le chapitre suivant).

Control Field

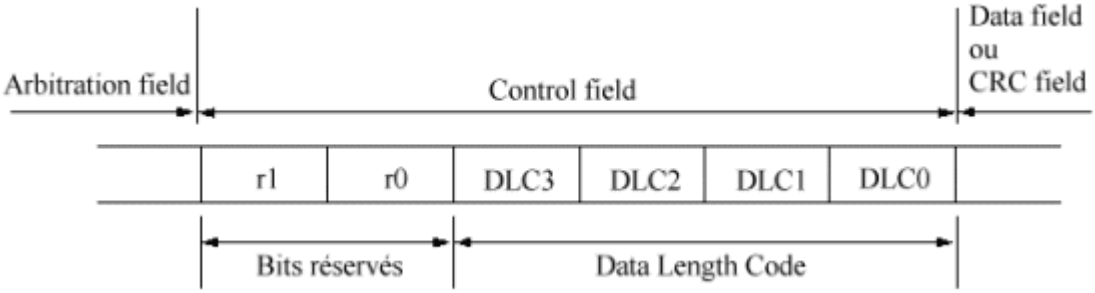


Figure 11 : Le champ de contrôle

Le Control Field est composé de 6 bits :

- les 2 premiers sont des bits réservés.
- es 4 suivants constituent le Data Length Code (DLC) qui indique le nombre d'octets du Data Field. 4 bits dominants (0000) correspondent à la valeur 0 pour le DLC, tandis que 1 bit récessif et 3 bits dominant (1000) correspondent à la valeur 8.Il y a donc 9 valeurs possibles pour le DLC

Data Field

Ce sont les données transmises par une Data Frame. Il peut contenir de 0 à 8 octets, et chaque octet est transmis avec le bit de poids fort en premier.

CRC field

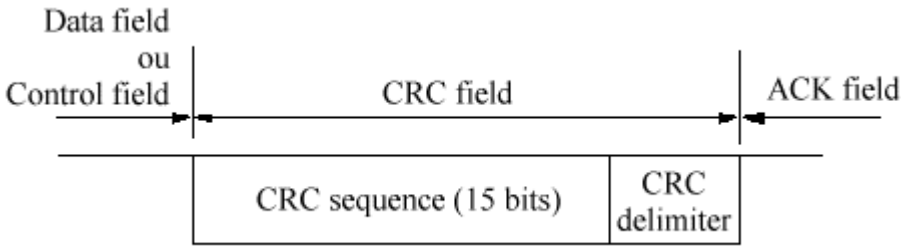


Figure 12 : Le champ CRC

CRC Field est composé de la séquence de CRC sur 15 bits suivi du CRC Delimiter (1 bit récessif).

a 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.

Le CRC est en fait un polynôme calculé de la même manière par l'émetteur et par le récepteur de la trame : le message est vu par l'algorithme comme un polynôme qui est divisé par X15+X14+X10+X8+X7+X4+X3+1 et le reste de cette division modulo[2] est la séquence CRC transmise avec le message.

Une erreur CRC est détectée si le résultat envoyé est différent du résultat reçu, dans ce cas le récepteur du message transmet un message d'erreur sous forme de Request Frame.

ACK Field

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

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

End Of Frame

Chaque Data Frame et Remote Frame est terminée par une séquence de 7 bits récessifs.

II – 3 – 3. Le format de la trame REMOTE FRAME (trame de requête)

Il se peut qu'un nœud ait besoin d'information d'un certain type dont il ne dispose pas pour assurer la mission dont il a la charge. Donc une station peut demander une transmission de données à une autre station en lui envoyant une Remote Frame.

Il existe deux différences entre une Remote Frame et un Data Frame :

- le bit RTR est transmis comme un dominant dans la Data Frame et comme un récessif dans la Remote Frame.
- la Remote Frame ne contient pas de données dans le champ Data Field

Ceci ne présente de l'intérêt que si une Data Frame et une Remote Frame sont transmises en même temps avec le même identifiant. En effet la Data Frame prendra la main grâce à son bit RTR dominant suivi de son identifiant. Donc le nœud qui a transmis la Remote Frame recevra son information tout de suite.

II – 3 – 4. La gestion des erreurs

L'error frame (trame d'erreur)

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

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

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

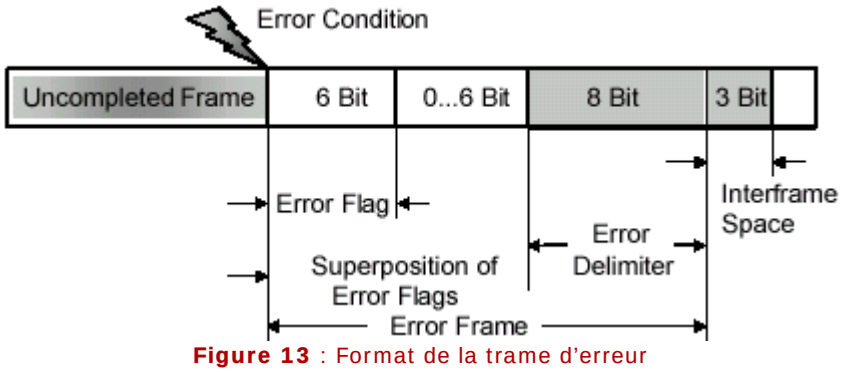


Figure 13 : Format de la trame d'erreur

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

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

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

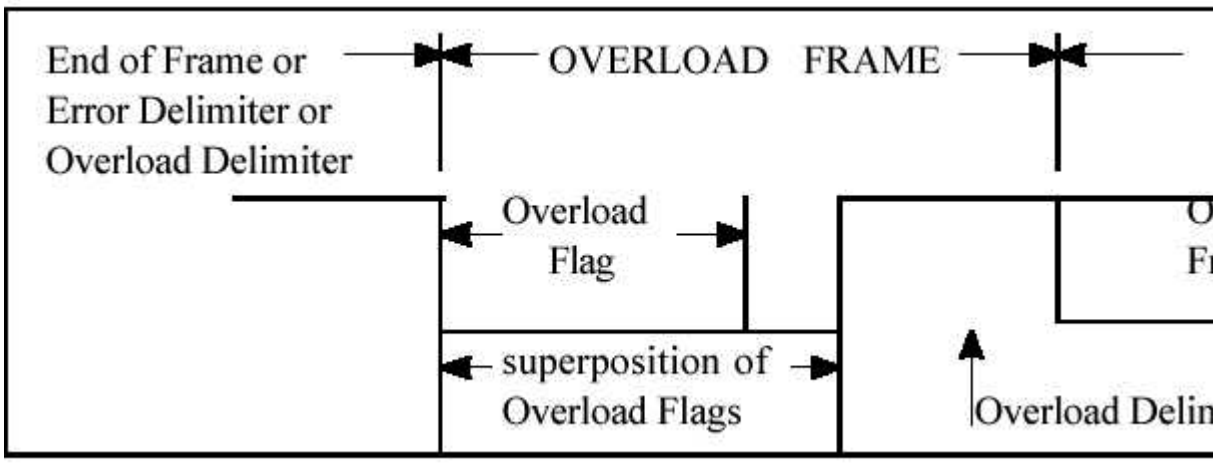
L'overload frame (trame de surcharge)

L' overload frame contient deux zones de données ; l'une contient l'overload flag et l'autre l'overload delimiter.

Il y a deux conditions pour transmettre un overload flag:

- les conditions internes d'un récepteur, qui exige le retard de la data frame suivante ou de la remote frame.
- les conditions internes d'un récepteur, qui exige le retard de la data frame suivante ou de la remote frame.

La première condition implique qu'une overload frame ne peut être transmise qu'à l'émission du premier bit de l'interruption demandée, tandis que les overload frames dues à la seconde condition attendent le temps d'un bit après avoir détecté le bit dominant.



Deux overload frames peuvent être produites afin de retarder les données suivantes ou la remote frames.

L'overload flag se composent de six bits dominants. Sa forme globale correspond quant à elle à celle de l'active error flag.

La forme de l' overload flag détruit la forme fixe de la zone d'interruption. Cela a pour conséquence que toutes les autres stations détectent un overload et un overload flag.

Dans le cas où un bit dominant est détecté pendant la lecture du 3ème bit de l'interruption à un certain nœud, les autres nœuds n'interpréteront pas l'overload flag correctement, mais interprèteront le premier des six bits dominants en tant que début de trame. Le sixième bit dominant viole la règle du bit stuffing et provoque ainsi une erreur.

L' overload delimitter se compose de huit bits récessifs.

Il est de la même forme que le error delimiter. Après transmission d'un overload flag, la station surveille le bus jusqu'à ce qu'elle détecte une transition d'un bit 'dominant' vers un bit récessif'. A ce moment là chaque station du bus a fini d'envoyer son overload flag et elles commencent la transmission de sept bits 'récessifs'.

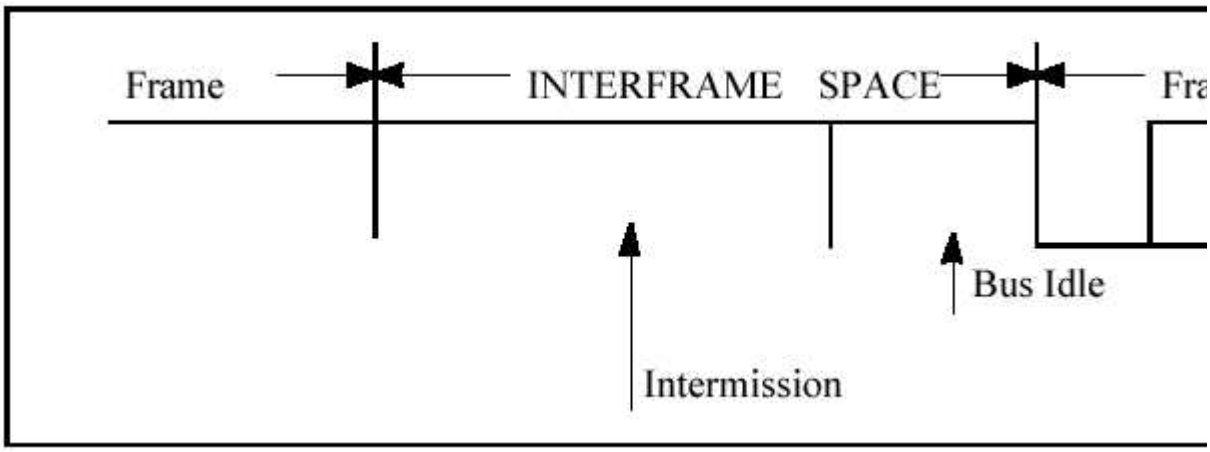
L'interframe spacing

Des trames de données et des remote frames sont séparées des trames qui les précédent (qui peuvent être des data frames, remote frames ,error frame, overload frame), par une zone de bit appelée interframe spacing. En revanche, les overload frames et les error frames ne sont pas précédées par un interframe space.

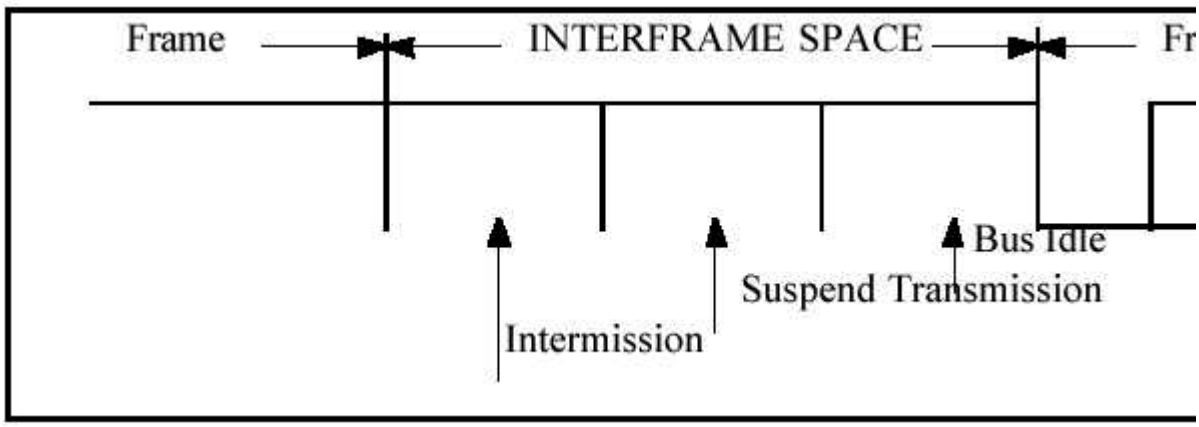
Plusieurs overload frames ne peuvent être séparées par un interframe space.

L'interframe space contient la zone de bit d'interruption et de bus IDLE.

Pour les stations qui ne sont pas 'error passive ' ou qui sont réceptrices du message précédent on a :



Pour les stations 'error passive' qui ont été émettrices du message précédent on a:



L'interruption se compose de trois bits récessifs.

Pendant l'interruption on ne permet à aucune station de commencer la transmission d'une trame de données ou d'une remote frame. L'unique action possible est de signaler une condition d'overload.

La période du bus IDLE est arbitraire. Ce bus est reconnu pour être libre et n'importe quelle station ayant quelque chose à transmettre peut avoir accès à celui-ci. Un message, qui est en suspend pendant la transmission d' un autre, est émis au premier bit suivant l'interruption.

La détection d'un bit dominant sur le bus est interprétée comme début de trame start of frame.

Après une 'error passive', une station transmet un message, elle envoie huit bits récessifs après l'interruption avant de commencer la transmission d'un autre message. Si elles se trouvent en attente d'un autre message (provoquée par une autre station), la station deviendra réceptrice de ce message.

II – 4. Le format étendu CAN 2.0 B

Le format étendu CAN 2.0 B diffère du format de base CAN 2.0 A par le nombre de bits de son identifiant.

En effet le champ Arbitration Field n'est pas codé sur 11 bits mais sur 29 bits, à savoir 18 bits supplémentaires (ID-17 à ID-0). Notons qu'il reste toujours un identifiant de base appelé Base Id qui est codé sur 11 bits.

Par rapport au format classique viennent se rajouter deux bits de contrôle (SSRbit et IDEbit)

Le bit RTR suit toujours l'identifiant comme dans le format 2.0 A et le format de la trame est le suivant :

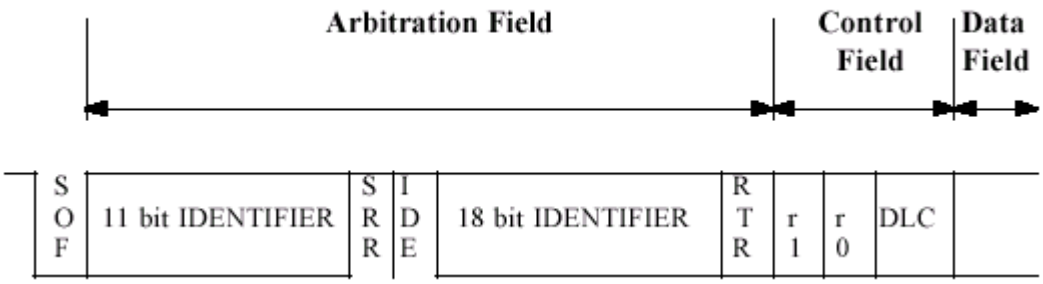


Figure 14 : Format de la trame dans la version 2.0 B

Lors de la transmission, l'ID est transmis et donc reçu en premier, suivi du bit IDE,du bit SRR et finalement du bit RTR.

Les fonctions attribuées à ces bits sont les suivantes :

Bit SRR (Substitute Remote Request Bit) : c'est un bit récessif (1) qui est transmis à la place du bit RTR du format classique. Il le substitue d'où son nom. Ce bit est utile lorsqu'une trame au format étendu circule sur un bus CAN et rencontre une trame classique qui possède le même Base Id. Le récepteur tiendra alors compte de cette dernière.

Bit IDE (Identifier Bit) : c'est un bit dominant (0) qui a la même fonction que celle du bit IDE dans le format classique mais il appartient dans ce format à l'Arbitration Field.

III – Le protocole CAN Open

III – 1. Généralités

III – 1 – 1. Introduction

Can Open est un protocole qui utilise le bus série CAN et qui respecte la norme ISO 11898. Ce protocole impose des mécanismes de communication standardisés qui seront décris par la suite.

A sa création, CAN Open était destiné à des systèmes industriels de contrôle de mouvement ou de manipulation tandis qu'aujourd'hui il est utilisé dans de nombreuses applications comme les véhicules, les transports publics, les équipements médicaux ou l'électronique maritime.

Le premier avantage du protocole CAN Open est qu'il supporte des systèmes temps réel car un temps maximal entre l'émission et la réception des trames pour un processus quelconque peut être défini.

Le second avantage est la programmation objet.

III – 1 – 2. Le modèle

Le modèle de communication est similaire au modèle ISO de référence traité précédemment. De plus, CAN Open est basé sur le CAN Data Link Layer et sur l'émission et la réception à haute vitesse comme elle est décrite dans la norme ISO 11898. CAN Open spécifie en outre le Bit-Timing et recommande des câblages pour les connecteurs.

Le CAN Protocol Layer Interaction décrit la communication sur les différentes couches. Sur la couche application les nœuds échangent des objets de communication et d'application. Ces objets sont accessibles via un index sur 16 bits et un sous-index sur 8 bits. Ces objets de communication (COB) sont mappés.

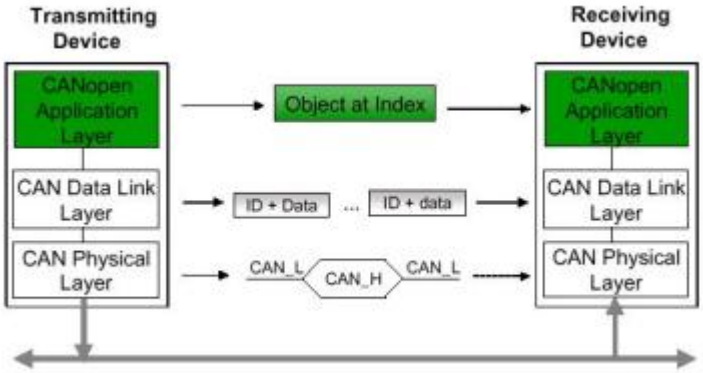


Figure15 : Protocol Layer Interaction

III – 1 – 3. Le bit timing

Précisons le bit timing déjà évoqué dans le chapitre précédent.

Lors d'une transmission à 1 Mbit/s, un bit fait 8 quantums de temps, tandis qu'à 10 kbits/s un bit en fait 16.

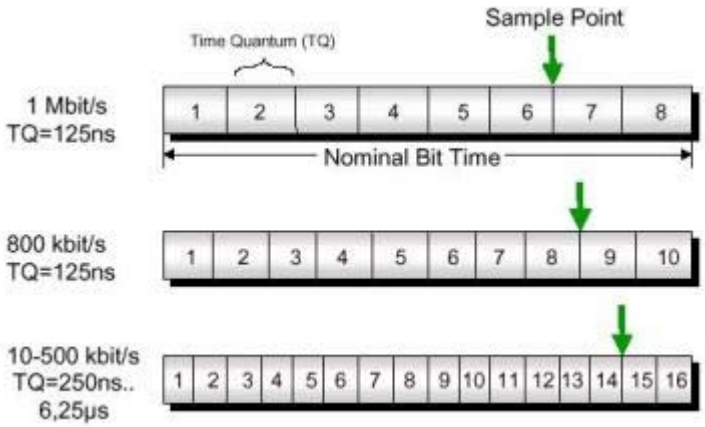


Figure 16 : Description du bit timing(1)

Chaque nœud doit supporter une des vitesses de transmission spécifiées dans le tableau suivant :

Bit rate Bus length	Nominal bit time t_b	Number of time quanta per bit	Length of time quantum t_q	Location of sample point	Index
1 Mbit/s 25 m	1 µs	8	125 ns	6 t_q (750 ns)	0
800 kbit/s 50 m	1.25 µs	10	125 ns	8 t_q (1 µs)	1
500 kbit/s 100 m	2 µs	16	125 ns	14 t_q (1.75 µs)	2
250 kbit/s 250 m	4 µs	16	250 ns	14 t_q (3.5 µs)	3
125 kbit/s 500 m	8 µs	16	500 ns	14 t_q (7 µs)	4
50 kbit/s 1000 m	20 µs	16	1.25 µs	14 t_q (17.5 µs)	5
20 kbit/s 2500 m	50 µs	16	3.125 µs	14 t_q (43.75 µs)	6
10 kbit/s 5000 m	100 µs	16	6.25 µs	14 t_q (87.5 µs)	7

Figure 17 : Description du bit timing(2)

III – 2. Les types de données

III – 2 – 1. Les données statiques

Les données statiques sont placées dans l'Object Dictionnary afin d'être définies entre l'index 0001h et 0016h puis entre 0018h et 001Bh.

Le type boolean

Ces données prennent la valeur TRUE ou FALSE.

Le type integer

Ces données prennent des valeurs entières entre –2^(n-1) et 2^(n-1)-1.

Le type unsigned

Ces données prennent des valeurs entières positives entre 0 et 2^(n-1)-1.

Le type visible string

Le type octet string

Le type date

Ces données prennent une valeur codée sur 56 bits et incluent les ms, les minutes, l'heure, l'heure standard ou d'été, le jour, le mois, la semaine et l'année.

Le type time of day

Ces données représentent le temps absolu, c'est à dire le temps écoulé en ms et en nombre de jours depuis le 1er janvier 1984.

Le type time difference

Ces données représentent la différence de temps en jours ou en ms.

III – 2 – 2. Les données complexes

Les données de ce type sont prédéfinies pour les paramètres des PDO et des SDO. De plus,

l'Object Dictionnary réserve des entrées spécifiques pour les nœuds qui en auraient besoin.

Elles sont définies dans l'Object Dictionnary entre l'index 0020h et 0023h et entre 0040h et 025Fh (pour les entrées spécifiques).

III – 3. Objets et Object Dictionnary

La partie la plus importante d'un élément Can Open est l'Object Dictionnary. Il s'agit en fait d'une table regroupant tous les objets accessibles via le bus. Chaque objet du dictionnaire est adressé en utilisant un index sur 16 bits et un sous-index sur 8 bits.

Index (hex)	Object
0000	Reserved
0001-001F	Static Data Types
0020-003F	Complex Data Types
0040-005F	Manufacturer Specific Data Types
0060-007F	Device Profile Specific Static Data Types
0080-009F	Device Profile Specific Complex Data Types
00A0-0FFF	Reserved for further use
1000-1FFF	Communication Profile Area
2000-5FFF	Manufacturer Specific Profile Area
6000-9FFF	Standardized Device Profile Area
A000-FFFF	Reserved for further use

Figure 18 : L'Object Dictionnary

Il est intéressant de remarquer que des zones sont réservées au fabricants. Elles sont utilisées lors de l'implémentation d'un nouveau nœud sur le réseau.

Nota : des exemples d'objets seront donnés dans les paragraphes suivants.

III – 4. Protocoles de communication

Les objets de communication qui transitent sur le bus sont décrits par des services et des protocoles.

- Les transferts de données en temps réel sont réalisés à l'aide de PDO (Process Data Object).
- La lecture ou l'écriture d'entrées dans l'Object Dictionnary d'un élément sont réalisés à l'aide de SDO (Service Data Object).
- Les applications telles que la Synchronisation, le Time Stamp, l'Emergency ont des protocoles spéciaux.
- Le protocole du Network Management (NMT) donne des services pour l'Initialisation, le Contrôle d'Erreurs, et l'Etat d'un Nœud.

III – 4 – 1. COB et COB-ID

Lors de l'explication de la méthode d'arbitrage, nous avons vu que la priorité est attribuée selon la valeur du champ d'arbitrage sur 11 bits car nous sommes dans la version 2.0 A.

Les 4 premiers bits sont le COB (objet de communication) et les 7 suivants sont l'ID (identifiant).

III – 4 – 2. Client/Server, Producer/Consumer, Master/Slave

Le modèle Client/Server

Dans ce modèle, le client transmet un message auquel le serveur répond. Le client obtient ainsi une confirmation.

Ce modèle est utilisé pour transmettre des données dont la longueur est supérieure à 8 octets. En fait la donnée originale est segmentée et transmise segment par segment. L'élément recevant les segments peut renvoyer une confirmation après avoir reçu un ou plusieurs segments. Cette communication est donc du type peer-to-peer.

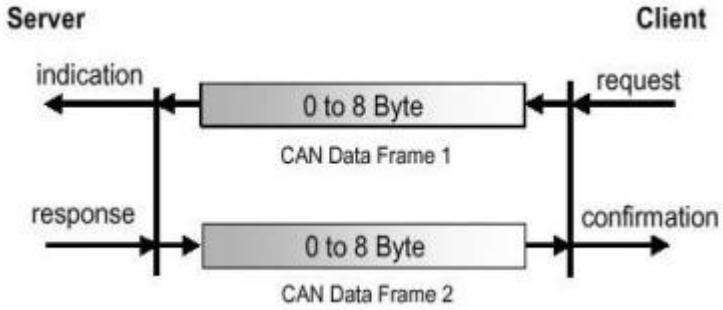


Figure 19 : Le modèle Client/Server

Le modèle Master/Slave

Dans ce modèle, seul le maître peut commencer une communication et l'esclave est toujours en train d'attendre une requête de communication du maître.

Dans les réseaux CAN, ce modèle peut être implanté en allouant des identifiants appropriés aux différents éléments.

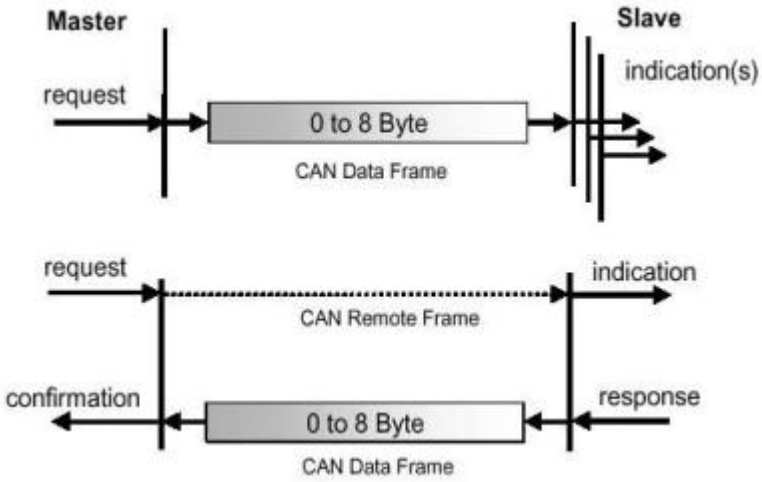


Figure 20 : Le modèle Master/Slave

Le modèle Producer/Consumer

Ce modèle décrit parfaitement les possibilités de broadcast du réseau CAN. En effet, chaque élément du réseau peut écouter les messages qui transitent sur le bus.C'est après avoir reçu le message que l'élément décide si il l'accepte ou non.

Le modèle Producer/Consumer permet :

- de transmettre des messages (modèle Push)
- de demander des messages (modèle Pull)

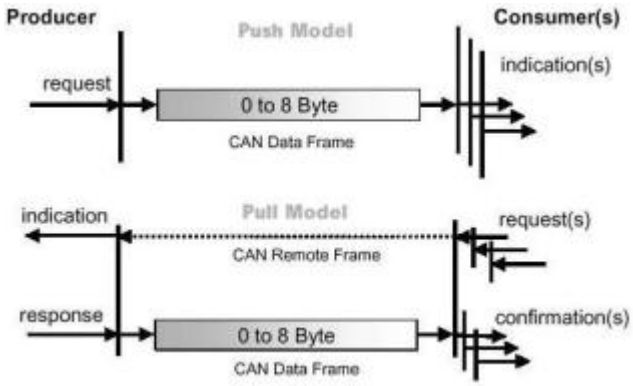


Figure 21 : Le modèle Producer/Consumer

III – 4 – 3. PDO (Process Data Object) et PDO Mapping

Les messages PDO sont utilisés pour manipuler des objets de l'Object Dictionnary sans se référencer explicitement à leurs identifiants, les types des données et la liste des objets à transmettre étant définis dans une structure appelée PDO mapping.

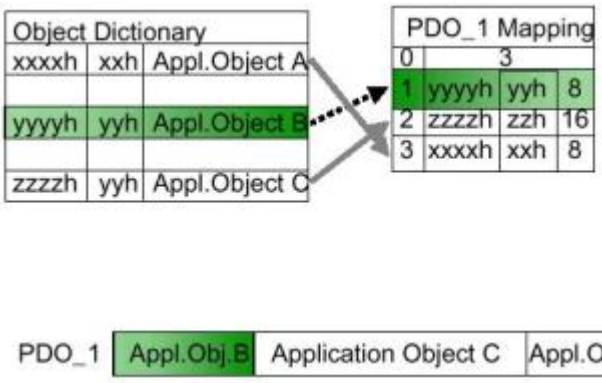


Figure 22 : Le PDO mapping

Dans le schéma ci-dessus, le PDO mapping PDO_1 va provoquer la réalisation successives des applications décrites par les objets yyyyh, puis zzzzh et enfin xxxxh

Une communication par PDO ressemble au modèle Producer/Consumer. La donnée est transmise sans demande de confirmation et est émise d'un nœud (Producer) vers un, plusieurs ou tous les nœuds (Consumers).

Le nœud Producer envoie un Transmit-PDO qui correspond, pour le nœud consommateur, à un Receive-PDO.

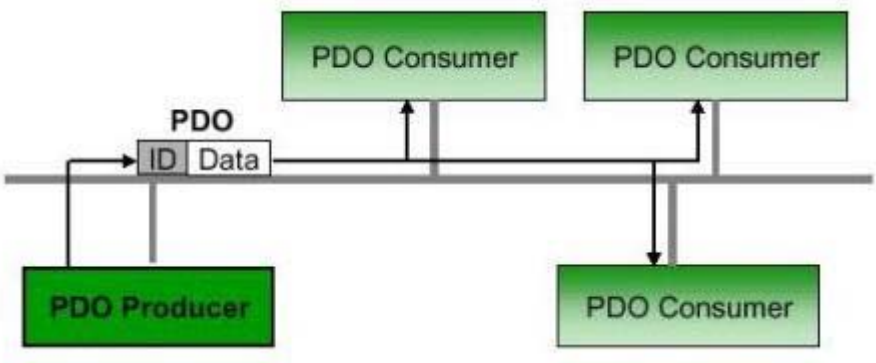


Figure 23 : Le PDO

Les messages PDO correspondent aux entrées de l'Object Dictionnary et fourniront à l'application concernée les données nécessaires à son fonctionnement.

Il existe principalement deux types de message PDO:

Ecrire un PDO

Ce service est contenue dans une seule Data Frame.

Lire un PDO

Ce service est contenu dans une Remote Frame. La réponse à cette dernière est la Data Frame correspondante.

Dans les deux cas, le champ de données de la trame contient la donnée du process sur 8 octets.

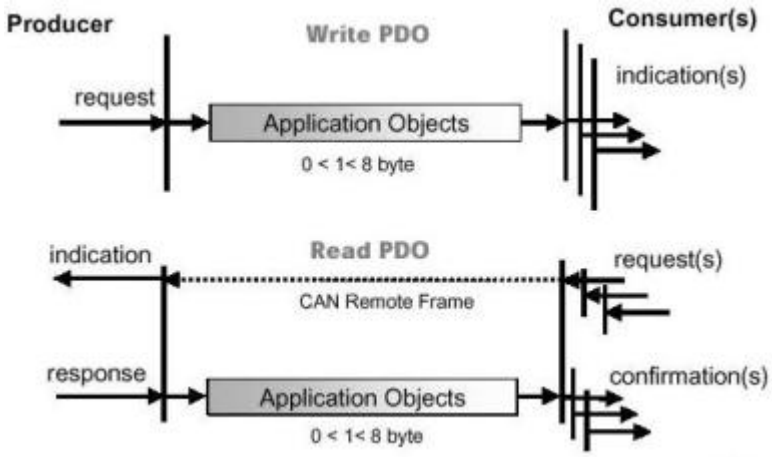


Figure 24 : Lecture et écriture d'un PDO

Deux paramètres permettent de décrire un PDO :

Le PDO communication parameter qui décrit les possibilités de communication d'un PDO

Le PDO mapping parameter qui contient les informations relatives aux contenus des PDO.

Index	Sub-Index	Description	Data Type
1XXXh	0h	number of entries	Unsigned8
	1h	COB-ID	Unsigned32
	2h	transmission type	Unsigned8
	3h	inhibit time	Unsigned16
	4h	reserved	Unsigned8
	5h	event timer	Unsigned16

Figure 25 : Le PDO communication parameter

Index	Sub-Index	Description	Data Type
1XXXh	0h	number of mapped	Unsigned8
	1h	1st object	Unsigned32
	2h	2nd object	Unsigned32
	..	..	..
	40h	64th object	Unsigned32

31 16	15 8	7 0
16-bit Index	8-bit Sub-index	length

Figure 26 : Le PDO mapping parameter

nota 1 : avant d'effectuer une transmission de type PDO, le numéro des PDO ainsi que la cartographie des objets (mapping) doivent être transmis aux éléments du réseau intéressés pendant la phase de configuration (Device Configuration) c'est à dire pendant l'initialisation. Ceci s'effectue d'ailleurs à travers un message de type SDO.

nota 2 : la relation Producer-Consumer implique l'existence d'un Producer et de zéro ou plus Consumers. Du Producer vers le Consumer c'est le Push model qui est un service dit unconfirmed.Du Consumer vers les Producer c'est le Pull model qui est un service dit confirmed.

Il existe des PDO synchrones et asynchrones (les PDO synchrones sont exécutés à chaque transmission du SYNC –voir plus loin- tandis que les asynchrones peuvent être exécutés à n'importe quel moment du moment que la priorité est respectée).

Il existe 4 types de PDO correspondant à différents niveaux de priorité (PDO1 ayant la priorité la plus élevée) et les objets sont :

Transmettre un PDO1

Le COB-ID de cet objet vaut 180h + Node ID. Il varie donc suivant le nœud considéré de 181h à 1FFh. L'index de cet objet est 1800h.

Recevoir un PDO1

Le COB-ID de cet objet 200h + Node ID. Il varie donc suivant le nœud considéré de 201h à 27Fh. L'index de cet objet est 1400h.

Transmettre un PDO2

Le COB-ID de cet objet vaut 280h + Node ID. Il varie donc suivant le nœud considéré de 281h à 2FFh. L'index de cet objet est 1801h.

Recevoir un PDO2

Le COB-ID de cet objet vaut 300h + Node ID. Il varie donc suivant le nœud considéré de 301h à 37Fh. L'index de cet objet est 1401h.

Transmettre un PDO3

Le COB-ID de cet objet vaut 380h + Node ID. Il varie donc suivant le nœud considéré de 381h à 3FFh. L'index de cet objet est 1802h.

Recevoir un PDO3

Le COB-ID de cet objet vaut 400h + Node ID. Il varie donc suivant le nœud considéré de 401h à 47Fh. L'index de cet objet est 1402h.

Transmettre un PDO4

Le COB-ID de cet objet vaut 480h + Node ID. Il varie donc suivant le nœud considéré de 481h à 4FFh. L'index de cet objet est 1803h.

Recevoir un PDO4

Le COB-ID de cet objet vaut 500h + Node ID. Il varie donc suivant le nœud considéré de 501h à 57Fh. L'index de cet objet est 1403h.

Le detail des objets est:

1800h : Transmit PDO Communication Parameter

DATA TYPE	PDO CommPar
OBJECT CODE	RECORD
SUB INDEX 0	Largest sub-index supported
SUB INDEX 1	COBID used by PDO
SUB INDEX 2	Transmission Type
SUB INDEX 3	Inhibit time
SUB INDEX 4	Reserved
SUB INDEX 5	Event timer

1400h : Receive PDO Communication Parameter

DATA TYPE	PDO CommPar
OBJECT CODE	RECORD
SUB INDEX 0	Largest sub-index supported
SUB INDEX 1	COBID used by PDO
SUB INDEX 2	Transmission Type
SUB INDEX 3	Inhibit time
SUB INDEX 4	Compatibility entry
SUB INDEX 5	Event timer

Transmettre un PDO Mapping

1A00h :Transmit PDO Mapping

DATA TYPE	PDO Mapping
OBJECT CODE	RECORD
SUB INDEX 0	Number of mapped applications objects in PDO
SUB INDEX 1-40h	PDO Mapping for the nth application object to be mapped

Recevoir un PDO Mapping

1600h :Receive PDO Mapping

DATA TYPE	PDO Mapping
OBJECT CODE	RECORD
SUB INDEX 0	Number of mapped applications objects in PDO
SUB INDEX 1-40h	PDO Mapping for the nth application object to be mapped

Ces objets sont regroupés dans le tableau suivant :

Receive PDO Communication Parameter (20H)					
1400	RECORD	1st receive PDO Parameter	PDOCommPar	rw	M/O*
1401	RECORD	2nd receive PDO Parameter	PDOCommPar	rw	M/O*
.....	.....	.....	.....	.....	.....
15FF	RECORD	512th receive PDO Parameter	PDOCommPar	rw	M/O*
Receive PDO Mapping Parameter (21H)					
1600	ARRAY	1st receive PDO mapping	PDOMapping	rw	M/O*
1601	ARRAY	2nd receive PDO mapping	PDOMapping	rw	M/O*
.....	.....	.....	.....	.....	.....
17FF	ARRAY	512th receive PDO mapping	PDOMapping	rw	M/O*
Transmit PDO Communication Parameter (20H)					
1800	RECORD	1st transmit PDO Parameter	PDOCommPar	rw	M/O*
1801	RECORD	2nd transmit PDO Parameter	PDOCommPar	rw	M/O*
.....	.....	.....	.....	.....	.....
19FF	RECORD	512th transmit PDO Parameter	PDOCommPar	rw	M/O*
Transmit PDO Mapping Parameter (21H)					
1A00	ARRAY	1st transmit PDO mapping	PDOMapping	rw	M/O*
1A01	ARRAY	2nd transmit PDO mapping	PDOMapping	rw	M/O*
.....	.....	.....	.....	.....	.....
1BFF	ARRAY	512th transmit PDO mapping	PDOMapping	rw	M/O*

Figure 27 : Les objets des PDO

III – 4 – 4. SDO (Service Data Object)

Le système de message SDO suit l'architecture Client-Server. Les SDO permettent d'avoir accès aux entrées de l'Object Dictionnary. Et comme ces entrées peuvent être de grande taille les SDO sont amenés à transférer plusieurs données appelées data sets, elles mêmes contenant de nombreux blocs de données ; le Client contrôle quelle data set doit être transmise via un multiplexer (index et sub-index de l'Object Dictionnary).

C'est le Client qui prendra l'initiative d'un tel transfert et le Server aura accès aux données de l'Object Dictionnary.

Le Client comme le Server pourra mettre fin à la transmission.

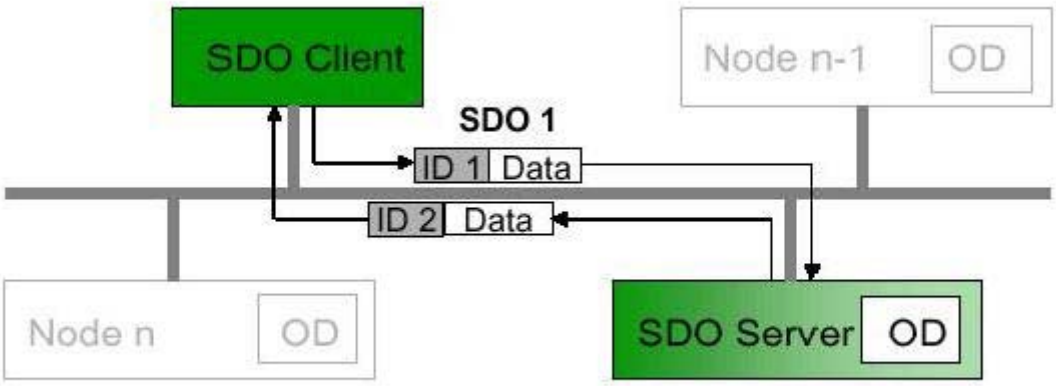


Figure 28 : Le SDO

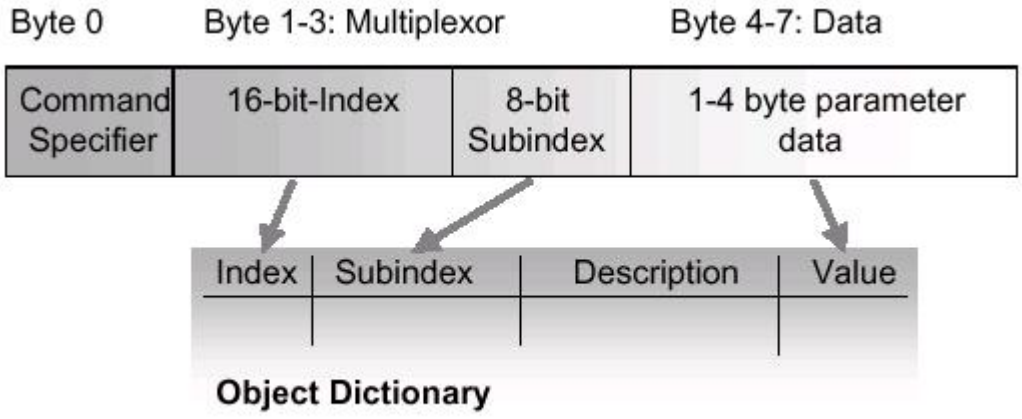


Figure 29 : L'accès à l'Object Dictionnary

Il existe deux types de message SDO : Le SSDO (Server-SDO) qui est le SDO par défaut, et le CSDO (Client-SDO).

Dans le cas d'un message SDO on se positionne par rapport au Server. Ainsi un SDO Download est la transmission de données du Client vers le Server et un SDO Upload consiste à transmettre des données d'un Server vers le Client.

Nous avons écrit précédemment que le SDO permet de transférer des données de n'importe quelle taille. Dans ce cas, on dit que le transfert est segmenté. Il se fait de la manière suivante :

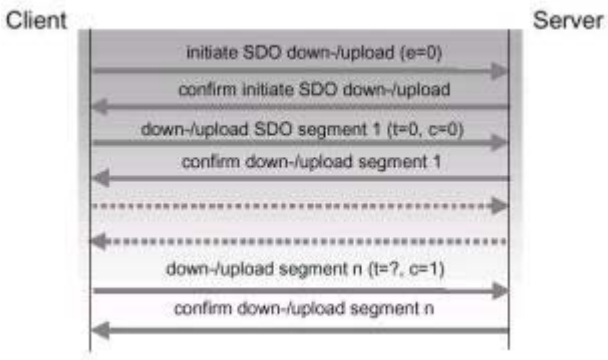
Les objets sont :

Transmettre un SDO

Le COB-ID de cet objet vaut 580h + Node ID. Il varie donc suivant le nœud considéré de 581h à 5FFh. L'index de cet objet est 1200h.

Recevoir un SDO

Le COB-ID de cet objet vaut 600h + Node ID. Il varie donc suivant le nœud considéré de 601h à 67Fh. L'index de cet objet est 1200h.



Le detail des objets est:

1200h : Server SDO Parameter

DATA TYPE	SDO Parameter
OBJECT CODE	RECORD

SUB INDEX 0	Number of entries
SUB INDEX 1	COBID RX
SUB INDEX 2	COBID TX
SUB INDEX 3	Node ID of the SDO client

1280h : Client SDO Parameter

DATA TYPE	SDO Parameter
OBJECT CODE	RECORD
SUB INDEX 0	Number of entries
SUB INDEX 1	COBID RX
SUB INDEX 2	COBID TX
SUB INDEX 3	Node ID of the SDO server

Ces objets sont regroupés dans le tableau suivant :

Server SDO Parameter (22H)					
1200	RECORD	1st Server SDO parameter	SDOParameter	ro	O
1201	RECORD	2nd Server SDO parameter	SDOParameter	rw	M/O**
.....	.....	.....	.....	.....	.....
127F	RECORD	128th Server SDO parameter	SDOParameter	rw	M/O**
Client SDO Parameter (22H)					
1280	RECORD	1st Client SDO parameter	SDOParameter	rw	M/O**
1281	RECORD	2nd Client SDO parameter	SDOParameter	rw	M/O**
.....	.....	.....	.....	.....	.....
12FF	RECORD	128th Server SDO parameter	SDOParameter	rw	M/O**
1300		reserved			
.....	.....	.....	.....	.....	.....

Figure 30 : Les objets des SDO

III – 4 – 5. La synchronisation

Le SYNC est broadcasté périodiquement sur le réseau par le SYNC Producer. C'est l'horloge de base du réseau. Le temps d'une période est défini par le paramètre standard Communication Cycle Period (1006h) qui peut être configuré par un outil de configuration et envoyé à l'application pendant le boot-up. La fréquence de cette horloge peut être temporairement accélérée lorsque certains messages doivent parcourir le réseau avant même que le signal SYNC ne soit transmis. Pour garantir l'accès de SYNC au réseau ce message est doté d'un haut niveau de priorité (1005h).

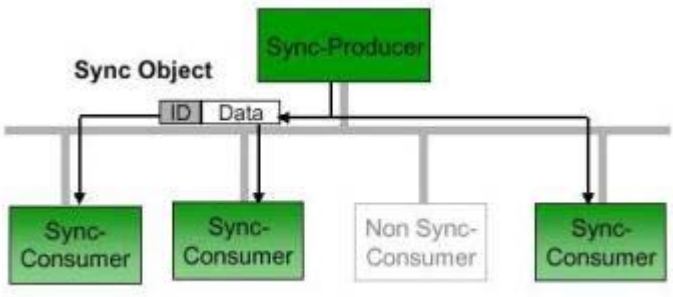


Figure 31 : La synchronisation

Le premier intérêt de la synchronisation est que lorsque chaque nœud reçoit le signal SYNC, il

exécute automatiquement son PDO mapping et envoie donc des données au contrôleur ou exécute une action. Cela permet donc de paramétrer les nœuds pour qu'ils effectuent périodiquement telle ou telle action.

Le second intérêt de la synchronisation est qu'elle permet de synchroniser l'horloge du nœud avec celle du contrôleur : c'est le rôle du Time Stamp (décrit dans le paragraphe suivant) qui contient la valeur de l'horloge maître codée sur 32 bits.

III – 4 – 6. Le time stamp

Dans la pratique, l'objet Time Stamp représente en ms le temps écoulé depuis le 1er janvier 1984.C'est une séquence de 48 bits (6 octets). Le producteur de Time Stamp reçoit une demande et envoie une Data Frame vers les Consommateurs.

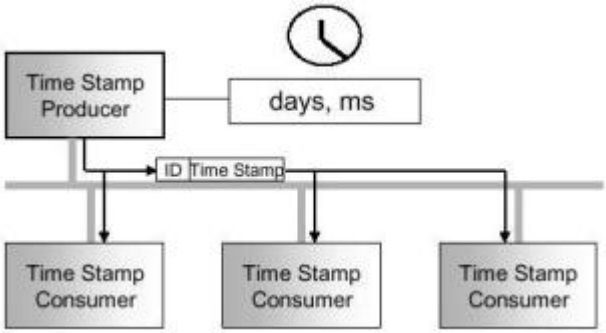


Figure 32 : Le time stamp

III – 4 – 7. L'emergency

L'emergency permet à un nœud d'envoyer avec une priorité élevée un message contenant le code d'erreur correspondant au problème auquel il est confronté. La table des codes d'erreurs est donnée dans la norme DS-301 page 9-38.

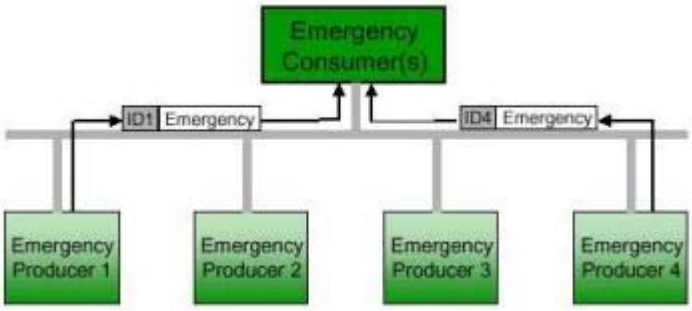


Figure 33 : L'emmergency

Les 8 bits de data dans la trame sont répartis de la manière suivante :

- Les deux premiers octets contiennent le code d'erreur cité ci dessus.
- Le second contient le registre d'erreur qui rend compte d'une erreur interne. Il correspond à l'objet 1001h.
- Les 5 derniers constituent le champ d'erreur spécifique au constructeur.

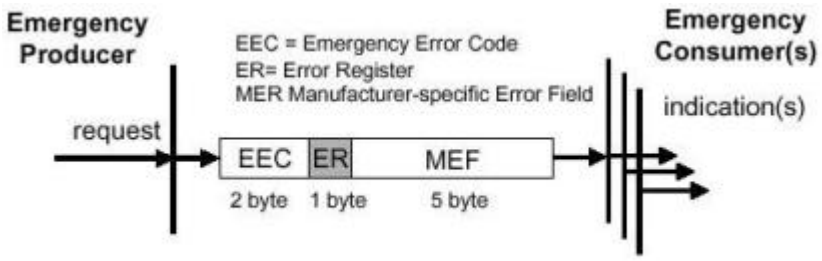


Figure 34 : La trame d'urgence

III – 4 – 8. Le network management (NMT)

Le NMT est un nœud dit orienté et suit la structure maître esclave.

A travers les services du NMT il est possible d'initialiser un nœud, de le démarrer, de faire un reset, de le stopper, de l'activer, de le contrôler.

Tous les nœuds sont considérés comme des esclaves du NMT et un tel esclave est uniquement identifié sur le réseau par son Node-Id, qui est une valeur comprise entre 1 et 127.

Le principe du NMT demande à ce que un élément du réseau remplisse les fonctions de maître NMT.

Les services NMT sont le MCS (Module Control Services) et l'ECS(Error Control Service).

Module Control Service (MCS)

C'est grâce au MCS que le maître NMT contrôle les différents états des esclaves NMT.

Le rôle de MCS peut être appliqué à un nœud particulier ou à tous les nœuds simultanément.

On remarquera que le maître NMT contrôle son propre état via un service local et que le MCS peut être activé par l'application locale (sauf dans le cas du Start Remote Mode)

Error Control Service (ECS)

A travers ce service le NMT détecte les erreurs du réseau CAN.

Les ECS interviennent principalement lors de défaut dans la périodicité de transmission de message d'un élément du réseau. Il existe deux cas où l'ECS va jouer son rôle :

- Le maître NMT établit une surveillance des nœuds du réseau (node guarding protocol) : si un esclave NMT n'a pas répondu dans un laps de temps imparti (node life time) ou encore si ce même esclave change sa politique de communication, le maître en informe sa propre application.

Principe de la surveillance de l'esclave par le maître: si le Life Time est supporté, l'esclave utilise le Life Time précisé dans son Object Dictionnary pour déterminer le Life Time du nœud sur lequel il est connecté, et dans le cas d'une erreur relative à ce temps il en informe sa propre application. Cette surveillance est généralement activée lors de la phase du boot-up.

- Le mécanisme de heartbeat consiste pour un élément de transmettre de façon cyclique un message de présence généré par un Heartbeat Producer. Un ou plusieurs éléments connectés au réseau reçoivent ce message et si sa périodicité change par rapport à celle du Heartbeat Producer alors l'application locale en est informée.

nota : l'implémentation du Heartbeat OU du Node Guarding est obligatoire.

Détaillons maintenant plus précisément les mécanismes de Heartbeat et de Node guarding.

III – 4 – 9. Le node guarding

Le Node Guarding est la surveillance des nœuds du réseau. Le maître du NMT (Network Management) envoie une RTR (Remote Transmission Request) à intervalles réguliers (cette période est appelée le Guard Time) et le nœud concerné doit répondre dans un laps de temps imparti(c'est le Node Life Time égal au Guard Time multiplié par le Life Time Factor).

La valeur du Life Time est 0 par défaut (c'est a dire qu'il n'est pas utilisé).

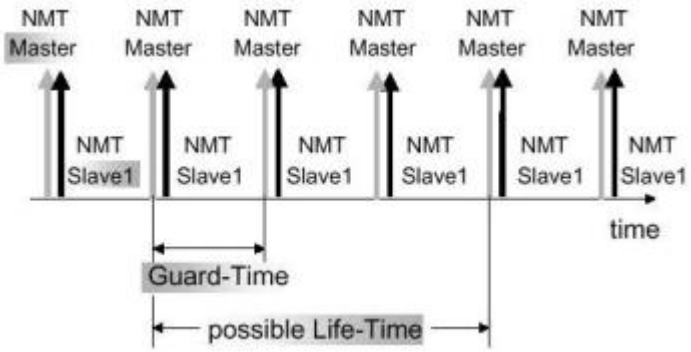


Figure 35 : Le guard time et le life time

Le bit de poids fort (7) est appelé Toggle bit. Sa valeur doit changer entre deux réponses consécutives. Lors de la première réponse il est à 0, puis 1, puis 0,etc…Si sa valeur est constante sur deux réponses consécutives, le maître du NMT considèrera qu'il n'a pas reçu de réponse.

Nota : Il est interdit d'utiliser en même temps le Node Guarding et le Heartbeat. Si le Heartbeat producer time n'est pas égal à 0, alors le mécanisme du Heartbeat est utilisé.

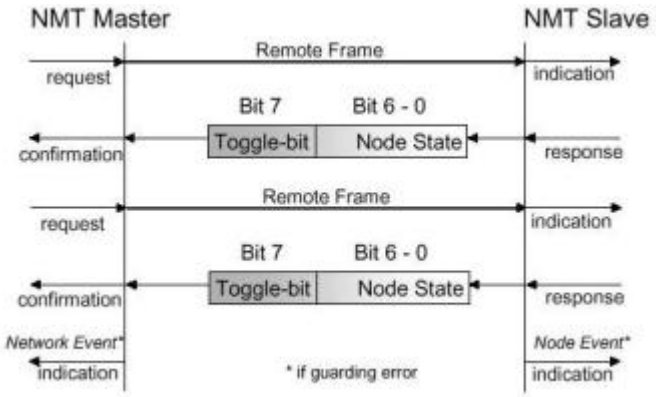


Figure 36 : Le node guarding

III – 4 – 10. Le Heartbeat

Le Heartbeat est généré périodiquement (la période est le Heartbeat Producer Time) par un nœud sans qu'il ait reçu au préalable une RTR (Remote Transmission Request). Il permet au nœud d'indiquer aux autres son état.

La data est contenue dans un seul octet :

- La data est contenue dans un seul octet :
- La data est contenue dans un seul octet :
  - 0 : Boot-up
  - 4 : Stopped

- 4 : Stopped
- 127 : Pre-Operational

Si un nœud n'a pas reçu de message de Heartbeat pendant un certain temps (appelé Heartbeat Consumer Time), un événement Heartbeat sera généré.

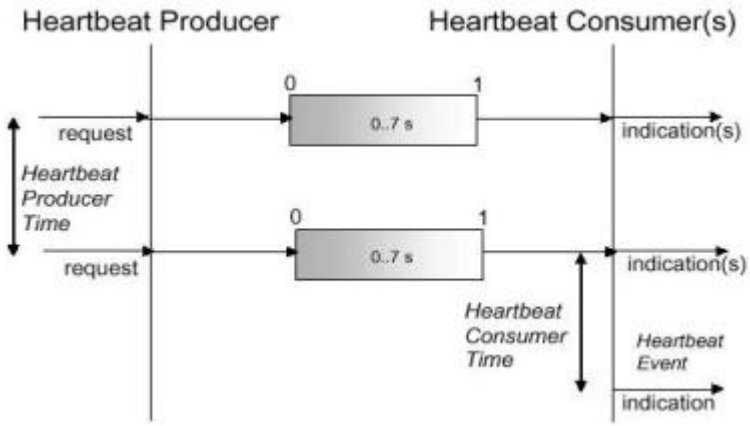


Figure 37 : Le Heartbeat

Auteurs : A2V - (01/2002)