Introduction au réseau CAN

(Z. MAMMERI – UPS)

1. Introduction

1.1 Origines et utilisations du réseau CAN

Pour satisfaire des 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 : systèmes anti-patinage, contrôle électronique du moteur et de l'air climatisé, fermeture centralisée des portes, etc. La complexité de ces systèmes et la nécessité d'échanger des données entre eux signifient de plus en plus de câbles. A côté du coût très important de ce câblage, la place qui lui est nécessaire pouvait le rendre tout simplement impossible à installer. Enfin, le nombre croissant de connexions et de câbles posait de sérieux problèmes de fiabilité et de réparation.

La société Robert Bosch GmbH (Allemagne), un important équipementier automobile, a fourni une solution dans les années 1980 avec le bus CAN (Controller Area Network). L'entreprise allemande a défini le protocole et a autorisé de nombreux autres fabricants à développer des composants compatibles CAN.

CAN est pensé et réalisé pour répondre à des impératifs de robustesse, de fiabilité, de simplicité et d'économie liés aux productions de masse de l'industrie automobile. CAN possède donc toutes les qualités pour séduire beaucoup d'industriels, soucieux de retrouver dans leurs installations ou leurs équipements, la fiabilité, la robustesse et le faible coût d'un système de communication normalisé et éprouvé.

CAN est un réseau de communication série qui supporte efficacement le contrôle en temps réel de systèmes distribués tels qu'on peut en trouver dans les automobiles, et ceci avec un très haut niveau d'intégrité au niveau des données. Avec le protocole CAN, les contrôleurs, capteurs et actionneurs communiquent entres eux à une vitesse pouvant aller jusqu'à 1 Mbits/s.

CAN est utilisé surtout pour la mise en réseau des organes de commande du moteur, de la boîte à vitesse, de la suspension et des freins (*figure 1*). Il s'agit là d'applications temps réel et critiques. Pour la mise en réseau des organes dits de carrosserie et de confort (commande des feux, des lève-vitres, de la climatisation, du verrouillage central, réglage de sièges et de rétroviseur), les constructeurs peuvent faire appel à CAN ou à d'autres réseaux de terrain comme VAN (Vehicle Area Network).

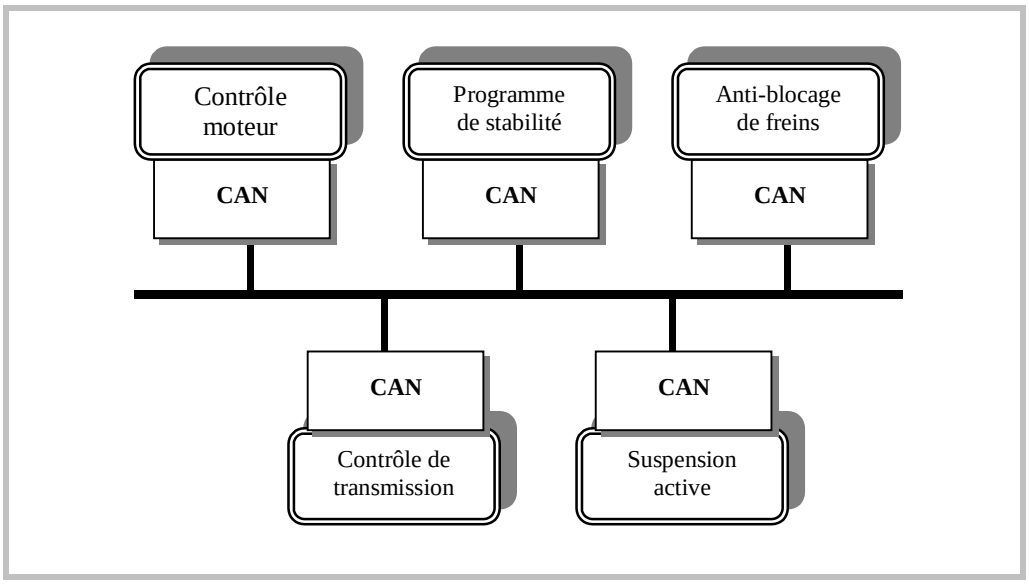


Figure 1 - CAN dans l'automobile.

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. C'est pourquoi les contrôleurs CAN ont été utilisés dans d'autres secteurs que l'automobile et des applications utilisant CAN sont aujourd'hui disponibles dans l'industrie, le bâtiment, l'agriculture, la marine, le matériel médical, les machines textiles, etc.

Les fondeurs de silicium, tels que Philips, Intel, NEC et Siemens, proposent aujourd'hui des composants et des contrôleurs CAN. La disponibilité d'outils d'accompagnement -tels que les analyseurs de réseau, les simulateurs de modules et les générateurs d'erreurs- facilitent le développement et la mise en œuvre des applications basées sur CAN.

Le CiA (CAN in Automation), créé à l'initiative des fournisseurs et utilisateurs allemands, est un club chargé de promouvoir le transfert d'une technologie automobile vers le monde industriel. Le CiA regroupe notamment des utilisateurs, des fabricants de semiconducteurs, des fabricants d'automates programmables ou de cartes industrielles.

Des services de niveau application (niveau 7 du modèle OSI) orientés systèmes d'automatismes industriels et s'appuyant sur CAN sont disponibles aujourd'hui : DeviceNet (de Allen Bradley), SDS (de Honeywell), CAL (de Philips), etc.

1.2. CAN dans le modèle ISO/OSI

CAN est un réseau compatible au modèle de référence ISO/OSI (ISO : *International Organization for Standards*, OSI : *Open Systems Interconnection*). CAN a été normalisé par l'ISO dans les normes 11898 pour les applications à haute vitesse (jusqu'à 1 Mb/s) et ISO 11519 pour les applications à basse vitesse (jusqu'à 125 kb/s). Comme le montre la figure 2, CAN (à l'image de la quasi-totalité des réseaux locaux industriels) a une architecture en trois couches ; les couches 3 à 6 du modèle OSI sont vides dans les architectures fondées sur CAN. Les spécifications CAN s'intéressent essentiellement aux couches MAC et physique.

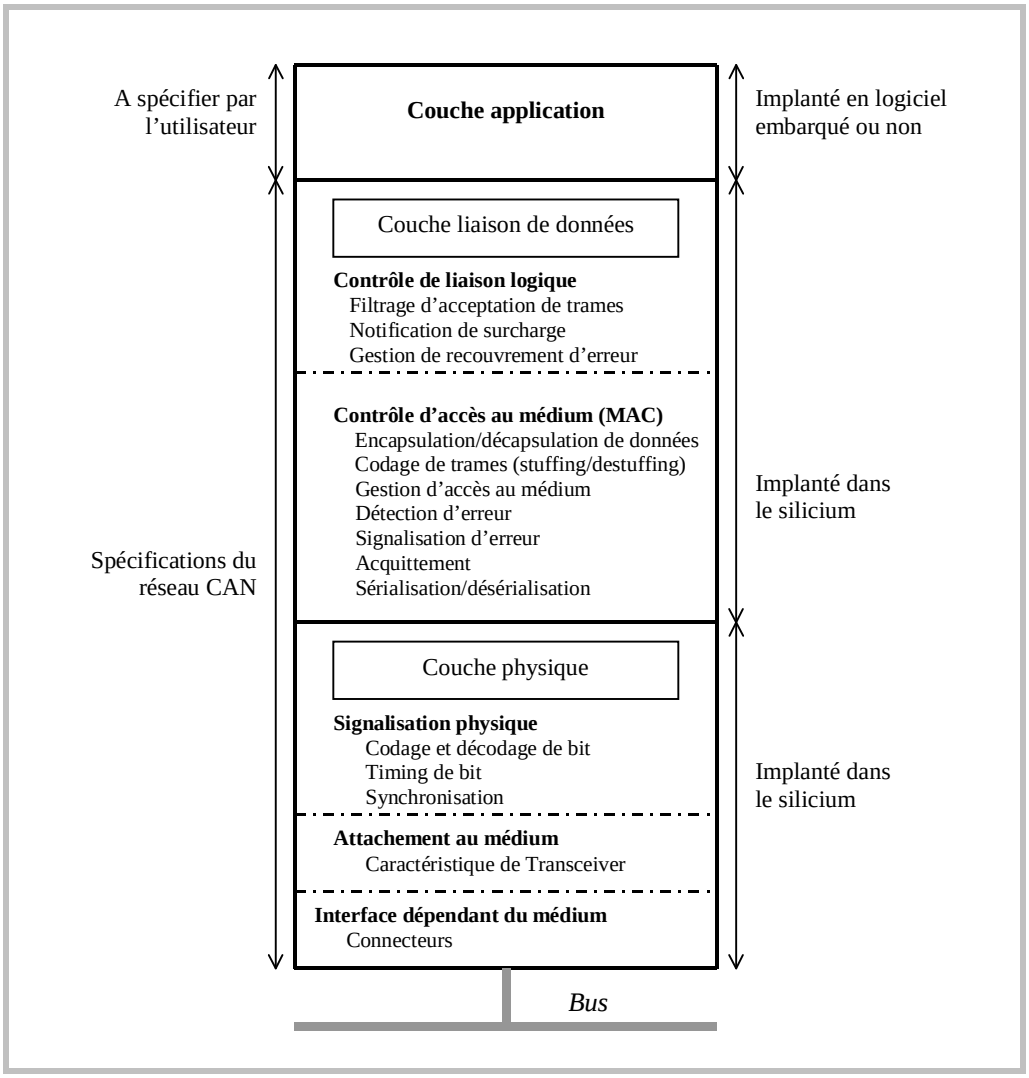


Figure 2 - Architecture CAN.

2. Fonctionnement du réseau CAN

2.1. Couche physique

■ Support, topologie et codage

Généralement, CAN utilise comme support de transmission une paire torsadée blindée ou non blindée. Les nœuds sont reliés entre eux par l'intermédiaire d'un bus série équipé de terminateurs de lignes (résistances de terminaison). Les interfaces physiques sont de type différentiel en mode tension, proche du principe de la liaison RS485. La figure 3 montre un exemple de raccordement au bus CAN.

REMARQUE : La norme CAN ne spécifie pas de couche physique unique. Différentes implémentations sont donc possibles : fibre optique, câble coaxial, infrarouge, etc.

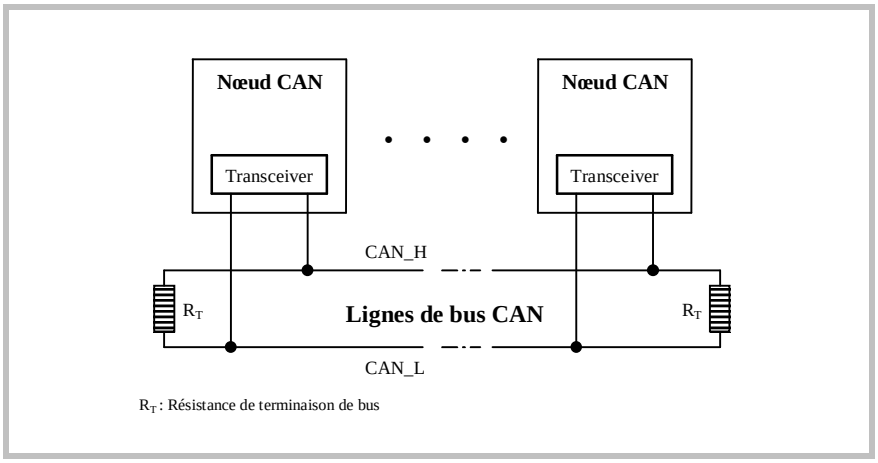


Figure 3 - Topologie en bus du réseau CAN .

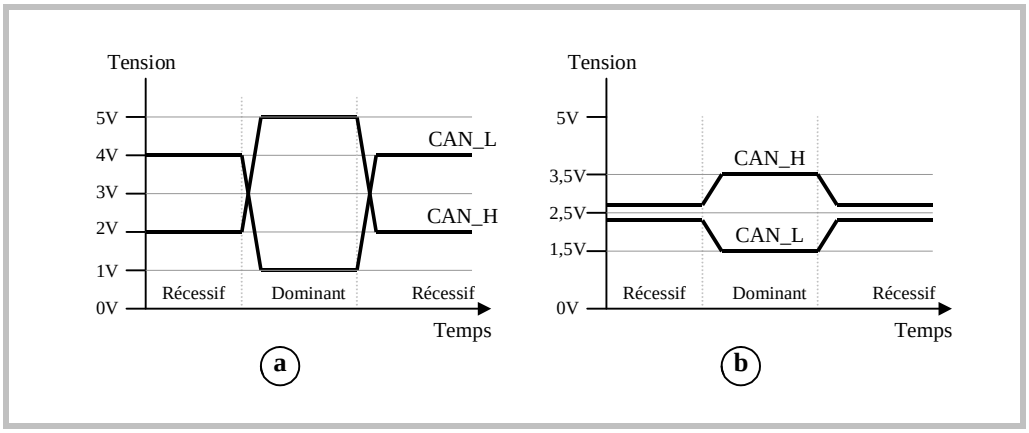


Figure C .4 - Représentation physique des bits :
a) cas de la norme ISO 11519 (basse vitesse)
b) cas de la norme ISO 11898 (haute vitesse)

Parmi les multiples techniques de codage existantes (NRZ, NRZI, Manchester simple, Manchester différentiel, etc.), CAN a retenu le code NRZ (Non Return to Zero) pour sa simplicité. Avec le code NRZ, la valeur du signal reste constante pendant toute la durée d'un bit.

Dans la norme ISO 11898 (haute vitesse), les nœuds détectent un bit récessif si la différence de tension entre les lignes CAN_H et CAN_L est inférieure ou égale à 0,5 V. Ils détectent un bit dominant si cette différence est supérieure ou égale à 0,9 V. La tension nominale pour le bit dominant est 3,5 V pour la ligne CAN_H et 1,5 V pour la ligne CAN_L.
Dans la norme ISO 11519 (basse vitesse), les nœuds détectent un bit récessif si la différence de tension entre CAN_L et CAN_H est de l'ordre de 1,5 V. Ils détectent un bit dominant si cette différence est de l'ordre de - 3 V.

■ Longueur et débit

Le signal correspondant à un bit émis par un nœud se propage à une vitesse de l'ordre de 200 000 km/s (c'est-à-dire 2/3 de la vitesse de la lumière) sur les lignes électriques et les fibres optiques. La durée la plus longue de propagation est celle où un bit doit parcourir le bus d'une extrémité à l'autre (cette durée est notée *t_bus*). A cause du délai de propagation -qui est non nul-, deux nœuds ayant détecté que le bus est libre peuvent transmettre en même temps. En particulier, un nœud se trouvant à une extrémité du bus peut commencer sa transmission quelques « micro » instants avant l'arrivée du bit venant du nœud se trouvant à l'autre extrémité. Pour que chacun des nœuds puisse détecter s'il est en conflit, le temps que doit durer un bit, appelé *temps nominal de bit* (*TN_bit*), doit être supérieur à deux fois le temps de propagation d'un bit sur toute la longueur du bus (*t_bus*).

*TN_bit > 2*t_bus*

t_bus = longueur_du_bus/200 000 /* *longueur du bus* exprimée en km */

Des paramètres autres que la longueur du bus interviennent dans le calcul du débit maximum d'un réseau CAN. Ainsi, la norme CAN définit le temps nominal de bit comme étant composé des temps suivants exprimés à l'aide d'un quantum de base:

- *T_Syn* : un temps (1 quantum) de synchronisation des différents nœuds sur le bus ;
- *T_Prog* : un temps (qui peut varier de 1 à 8 quanta) pour compenser les retards introduits par le bus et les drivers de ligne. *T_Prog* est égal à deux fois la somme de *t_bus*, le retard d'entrée et de sorite de driver (transmetteur) de ligne.
- *T_Phase1* et *T_Phase2* : deux temps utilisés pour corriger les erreurs de phase. *T_Phase1* et *T_Phase2* dure chacun 1 à 8 quanta.

TN_bit = T_Syn + T_Prog + T_Phase1 + T_Phase2

Débit_du_réseau = 1/TN_bit.

La valeur du quantum de base est fixée en tenant compte de la fréquence de l'oscillateur qui pilote le contrôleur du réseau. Le choix de la longueur optimale pour un débit donné s'avère donc un peu délicate pour l'utilisateur. Les nombreuses années d'utilisation et d'expérimentation de CAN ont permis de mieux cerner (par la pratique) les bonnes valeurs du couple <débit, longueur>. Ainsi, le CiA et beaucoup d'utilisateurs industriels de CAN recommandent les configurations du tableau 1 :

Tableau 1 Configurations de débit et longueur préconisées par le CiA.

1 Mb/s sur 25 à 40 m
800 kb/s sur 50 m
500 kb/s sur 100 m
250 kb/s sur 250 m
125 kb/s sur 500 m
50 kb/s sur 1000 m
20 kb/s sur 2500 m
10 kb/s sur 5000 m

■ Notion de bit dominant et bit récessif

Plusieurs implantations de CAN sont possibles, mais toutes implantations doivent respecter le principe des bits dominants/récessifs. Chaque nœud doit pouvoir présenter sur le bus un bit appelé « dominant » (0 logique) et un bit appelé « récessif » (1 logique). Les implantations doivent aussi respecter la règle suivante : si deux nœuds présentent des niveaux logiques différents, le bit dominant s'impose. Le bus CAN se comporte donc comme un ET logique.

2.2. Protocole de niveau MAC

2.2.1 Identificateurs

Les trames de données (c'est-à-dire les trames qui contiennent les messages destinés à la couche application) transmises par un nœud sur le bus ne contiennent ni adresse d'un nœud destinataire, ni adresse du nœud source. Elles contiennent des identificateurs d'objets. Il y a un identificteur par trame. Par exemple, l'identificateur 100 désigne un angle de volant d'automobile et l'identificateur 101 désigne la température du moteur. Un identificateur est unique et spécifie, sans ambiguïté, l'information (ou l'objet) qu'une trame transporte. Les identificateurs sont donc attribués (par le concepteur d'application) aux objets selon l'urgence et l'importance de ces objets pour l'ensemble de l'application. En général, les objets associés aux grandeurs issues des capteurs ou à destination des actionneurs ont des identificateurs avec des valeurs faibles pour leur permettre d'être échangés en priorité en cas de conflit d'accès au bus (§ 2.2.2). En d'autres termes, l'identificateur contenu dans une trame spécifie la priorité de cette trame en cas de conflit d'accès. Plus la valeur de l'identificateur est faible, plus la trame est prioritaire.

Chaque nœud relié au réseau est producteur des valeurs d'un ou de plusieurs objets identifiés ou consommateur des valeurs d'un ou de plusieurs objets identifiés. Un nœud peut être à la fois producteur et consommateur de valeurs d'objets identifiés.

Grâce à l'identificateur contenu dans une trame, les nœuds connectés au réseau, et qui sont en permanence à l'écoute du bus, reconnaissent les objets qui les intéressent (ceux qu'ils consomment ou qu'ils produisent) et traitent les trames ; tout nœud fait une copie de la trame courante (s'il est consommateur de l'objet diffusé) ou envoie sa valeur (s'il est producteur de l'objet diffusé). C'est au niveau de la sous-couche Contrôle de liaison logique que les messages reçus sont filtrés.

Les identificateurs sont codés sur 11 bits en version standard (CAN 2.0.A) et sur 29 bits en version étendue (CAN 2.0.B).

2.2.2 Arbitrage bit à bit

Dans un système typique, certains paramètres vont changer plus rapidement que d'autres. Ce sera par exemple la vitesse d'un moteur d'automobile, tandis qu'un paramètre plus lent pourra être la température de l'habitacle. Il est donc naturel que les paramètres qui varient le plus soient transmis le plus souvent et par conséquent doivent avoir une plus grande priorité. Dans les applications temps réel, ceci nécessite non seulement une vitesse de transmission importante, mais aussi un mécanisme d'allocation du bus efficace qui soit capable de traiter les cas où deux ou plusieurs nœuds cherchent à transmettre en même temps. C'est la raison pour laquelle, CAN intègre une méthode simple et efficace pour arbitrer l'accès au bus. Cette méthode est appelée CSMA/CR ("Carrier Sense, Multiple Access with Collision Resolution") et a la capacité de l'arbitrage non destructif (dit "Non-Destructive Bitwise Arbitration").

REMARQUE : L'adjectif « non destructif » est utilisé pour distinguer la méthode CSMA/CR de sa "cousine" CSMA/CD ("CSMA with Collision Detection") pour laquelle toutes les trames sont détruites en cas de conflit d'accès au bus. On notera que CSMA/CD est la technique utilisée sur le réseau local Ethernet (qui est le réseau local le plus répandu).

Les caractéristiques électriques définies par la norme CAN, font que en cas de conflit d'accès (c'est-à-dire quand deux ou plusieurs nœuds commencent à transmettre en même temps), la valeur 0 écrase la valeur 1. Lors de l'arbitrage, dès qu'un nœud émetteur détecte un bit à 0 sur le bus, alors qu'il émet un bit à 1, il abandonne sa tentative d'accès au bus. L'un après l'autre, les nœuds en compétition abandonnent leur transmission au profit du nœud le plus prioritaire (c'est-à-dire celui qui transmet la trame ayant l'identificateur le plus petit). Tout se passe donc comme si la trame de plus haute priorité était la seule à être transmise. Lorsqu'un nœud perd l'arbitrage, il devient automatiquement un récepteur de la trame en cours de transmission. Les nœuds qui abandonnent tentent leur transmission plus tard une fois que le nœud le plus prioritaire en cours termine sa transmission et après un silence sur le bus équivalent à au moins 3 fois le temps de transmission d'un bit.

Comme l'arbitrage d'accès se fait sur la base des identificateurs contenus dans les trames, le choix de ces identificateurs est important pour le bon fonctionnement d'une application.

Dans l'exemple illustré par la figure 5, trois nœuds tentent de transmettre en même temps. Les deux premiers bits des identificateurs transmis sont identiques pour les trois nœuds, par conséquent les nœuds ne peuvent être départagés qu'à partir du troisième bit. Le troisième bit vaut 1, pour le nœud 2, et vaut 0, pour les nœuds 1 et 3. Par conséquent, le nœud 2 abandonne sa transmission au troisième bit. Ensuite, après un bit à 1 pour les deux nœuds restants, le nœud 1 abandonne, au cinquième bit, au profit du nœud 3 qui poursuit la transmission de sa trame jusqu'à sa fin (le nœud 3 a gagné la compétition d'accès au bus).

On dit aussi que CAN est un réseau *multi-maîtres*, car contrairement au réseau FIP, il n'y a pas de nœud privilégié (ou central) qui attribue le droit d'accès au support de transmission. Chaque nœud a le droit d'accéder au support dès qu'il détecte qu'il est libre sans attendre l'autorisation d'un quiconque autre nœud.

2.2.3 Types d'échanges d'objets identifiés

Selon les besoins de l'application, les objets identifiés peuvent être échangés par les nœuds de manière périodique, de manière sporadique ou sur demande d'un consommateur.

Il faut noter que la transmission d'une trame prend un certain temps (qui dépend principalement du débit du réseau) et que des nœuds sont contraints d'attendre avant de voir leurs tentatives de transmission réussir. Pour un fonctionnement de l'application sans violation des contraintes de temps, un calcul minutieux doit être fait pour montrer que les temps d'attente avant transmission n'affectent pas la validité des données échangées. Par exemple, le temps d'attente d'une trame contenant la valeur d'une température qui doit être échangée toutes les *P* millisecondes doit être inférieur à *P* millisecondes. La phase pendant laquelle le développeur d'une application temps réel doit vérifier a priori le bon fonctionnement de son application est dite phase d'analyse d'ordonnançabilité. Cette phase est d'une importance capitale pour les applications critiques (comme c'est le cas des applications embarquées dans l'automobile).

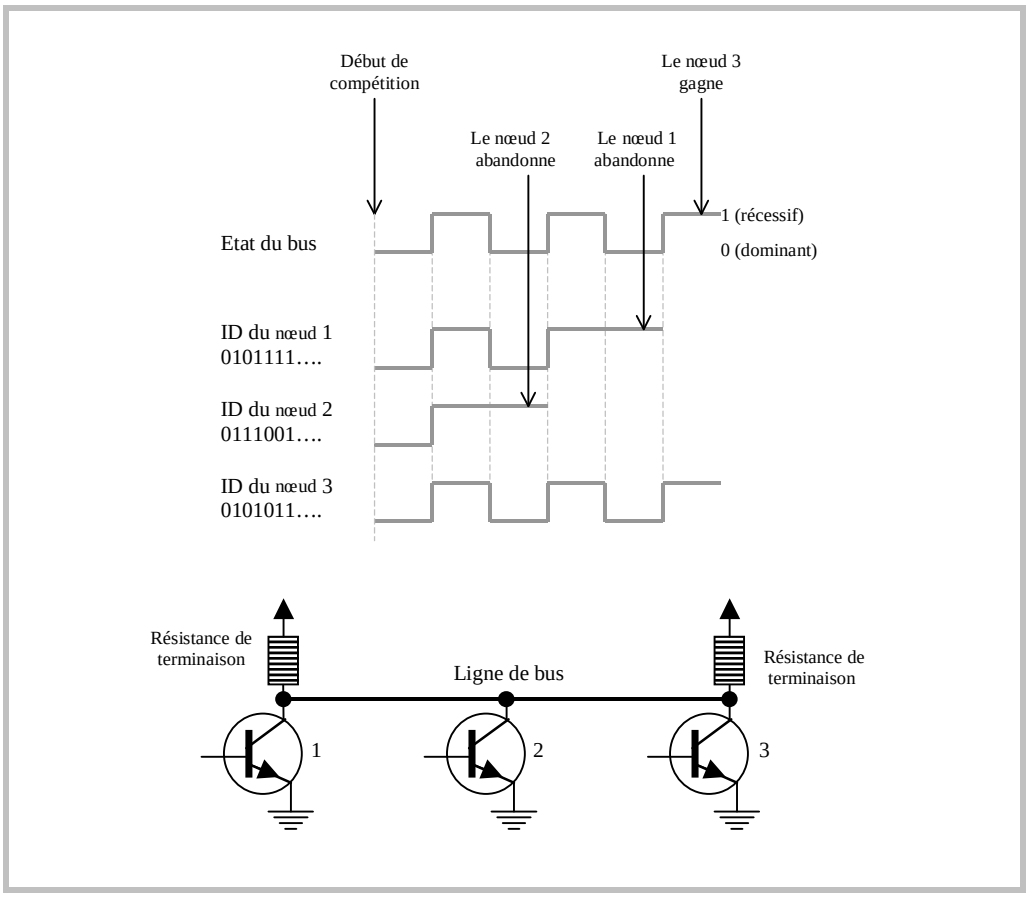


Figure 5 - Principe d'arbitrage du réseau CAN.

2.2.4 Formats de trames de CAN

■ Protocoles 2.0A et 2.0B

Le protocole CAN 2.0 comporte deux spécifications qui diffèrent uniquement au niveau de la longueur de l'identificateur (*figure 6*). La version 2.0A définit des identificateurs de 11 bits (on parle dans ce cas de trame de *format standard*) et la version 2.0B des identificateurs de 29 bits (on parle dans ce cas de trame de *format étendu*).

Pour permettre le développement de contrôleurs assez simples, le support complet du format étendu n'est pas requis pour être conforme au protocole 2.0. Les contrôleurs sont considérés conformes au protocole 2.0 s'ils respectent les deux conditions suivantes :

- Le format standard doit être totalement supporté.
- Ils doivent être capables de recevoir des trames de format étendu, mais sans forcément être capables de les traiter ; elles ne doivent seulement pas être détruites.

■ Types de trames

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. Leur format et utilisation seront détaillés par la suite (§ 2.2.6).
- Trame de surcharge : elles sont utilisées pour demander un délai entre deux trames de données ou de requête successives (§ 2.2.7).

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.

■ Format de trames de données et de requête

Comme le montre la figure 6, les trames CAN sont composées des éléments suivants (attention certains bits sont valables pour la version 2.0B seulement) :

- *Bit SOH* (Start of Heading) : il marque le début d'une trame de données ou de requête. C'est un bit dominant. Un nœud ne peut bien sûr débuter une transmission que si le bus est libre. Ensuite, tous les autres nœuds se synchronisent sur le bit SOH du nœud ayant commencé une transmission.
- *Champ d'arbitrage* : ce champ diffère selon le format (standard ou étendu) de trames.
  - Pour le format standard : le champ d'arbitrage est composé d'un identificateur sur 11 bits et d'un bit RTR.
  - Pour le format étendu : le champ d'arbitrage est composé d'un identificateur sur 29 bits, d'un bit SRR, d'un bit IDE et d'un bit RTR. L'identificateur est décomposé en deux parties séparées de 11 bits et 18 bits. Le bit IDE ("Identifier Extension") est un bit récessif. Le bit SRR ("Substitute Remote Request") est bit récessif.

  Le bit RTR est dominant dans les trames de données et récessif dans les trames de requête, pour les deux formats de trames.
- *Champ de contrôle* : ce champ est composé de 6 bits. Le premier bit dépend du format de trame : il s'appelle IDE et il est dominant dans le format standard et il est noté *r1* (réservé) dans le format étendu. Le deuxième bit (r0) est réservé pour des extensions futures. Les quatre autres bits indiquent la taille des données (DLC : Data Length Code). La valeur du DLC est forcément comprise entre 0 et 8, soit 9 valeurs. La valeur binaire 0000 correspond à une longueur nulle et la valeur 1000 correspond à la taille maximale (8 octets).
- *Champ de données* : ce champ véhicule les données de la couche application qui contiennent la valeur de l'objet désigné par le champ d'arbitrage. Il peut contenir de 0 à 8 octets.
- *Champ de CRC* (Cyclic Redundancy Check) : ce champ est composé de la séquence de CRC sur 15 bits suivis du délimiteur de CRC (1 bit récessif). La séquence de CRC permet de détecter les erreurs de transmission. Les bits utilisés dans le calcul du CRC sont ceux des champs SOH, arbitrage, contrôle et données. Les bits du CRC sont calculés par le polynôme générateur suivant :

  $X^{15} + X^{14} + X^{10} + X^8 + X^7 + X^4 + X^3 + 1$.

- *Champ ACK* (acknowledgement) : ce champ est composé de deux bits : un bit ACK et un bit délimiteur d'acquittement (ce dernier est toujours un bit récessif). A l'émission d'une trame, le bit ACK est mis à 1 (bit récessif). Ensuite ce bit est écrasé (remplacé par un bit dominant) par tout récepteur qui a reçu correctement la trame. Ce mécanisme évite d'utiliser des trames séparées pour faire les acquittements. Il faut faire attention : un acquittement positif (c'est-à-dire, bit ACK = 0) signifie qu'au moins un des récepteurs a reçu correctement la trame. La présence du bit d'acquittement n'indique en rien que le récepteur a accepté ou a rejeté les données contenues dans la trame. Par contre, l'absence d'acquittement (bit ACK = 1), indique que les nœuds récepteurs ont détecté une erreur et qu'il y a une forte chance que l'erreur soit due au nœud émetteur. Le temps pendant lequel le bit ACK peut être modifié par les récepteurs est appelé par la suite *fenêtre d'acquittement*.
- *Champ de fin de trame* : chaque trame de données ou de requête est terminée par une séquence de 7 bits récessifs. Après émission des 7 bits de fin de trame, le bus doit rester libre au moins pendant un temps égal au temps de transmission de trois bits.

REMARQUE : Dans un réseau CAN tous les nœuds acceptent simultanément les mêmes données ou aucun d'eux ne les accepte en cas d'erreur. Grâce à ses mécanismes de diffusion et de contrôle d'erreurs, CAN garantit la consistance des données échangées sur le bus : tous les consommateurs (non défaillants) d'un même objet ont la même image de cet objet.

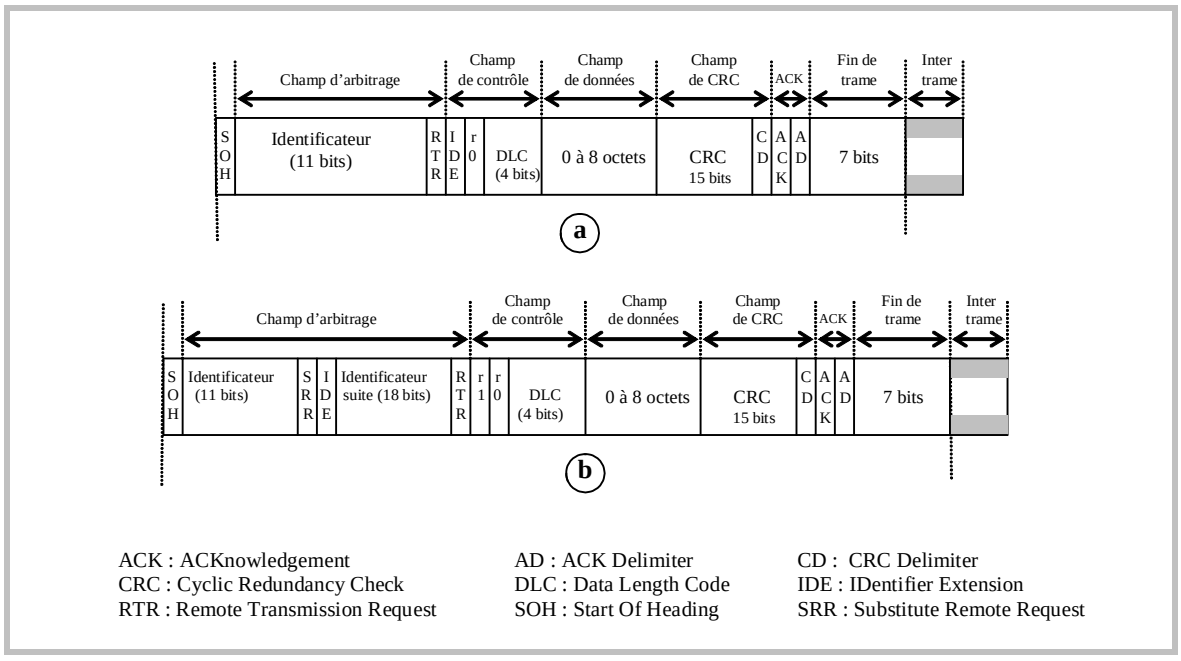


Figure 6 - Formats de trames de données et requête :
a) Format de trame standard (CAN 2.0A)
b) Format de trame étendu (CAN 2.0B).

2.2.5 Bit-stuffing

Pour renforcer la détection d'erreurs de transmission, CAN intègre un mécanisme dit de "bit stuffing" (ou bourrage de bits) qui se décrit de la manière suivante : quand l'émetteur détecte cinq bits identiques qui se suivent, il insère automatiquement un bit de valeur complémentaire qui sera supprimé à la réception. Les bits insérés sont pris en compte aussi pour détecter les suites de cinq bits consécutifs et identiques (voir l'exemple de la figure 7).

Il faut signaler aussi que l'opération de stuffing introduit dans le spectre du signal initial NRZ (comprenant la composante continue) une composante alternative fondamentale minimale que le signal ne possédait pas précédemment. Cet artifice permet aussi que l'on puisse assurer des liaisons isolées galvaniquement par transformateur. L'opération de stuffing permet aussi de créer (artificiellement) plus de transitions dans le signal, malgré l'utilisation du codage NRZ, ce qui permet de faciliter la synchronisation au niveau réception.

L'opération de "bit stuffing" s'applique seulement aux champs *début de trame* (*SOH*), *arbitrage*, *contrôle*, *données* et *CRC* ; les champs restants ont une forme fixe et ne sont pas codés par l'opération de "bit stuffing".

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 1 0 0 0 0 0 1 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**

Figure 7 - Exemple d'application de l'opération de "bit stuffing".

L'opération de "bit stuffing" permet d'améliorer les capacités de détection d'erreurs, mais elle réduit le débit utile du réseau. En effet, pour une trame qui contient *n* octets de données, l'opération de "bit stuffing" peut conduire à transmettre $44 + 8n + \lfloor (34 + 8n) / 4) \rfloor$ bits au lieu de *44 + 8n* bits. Dans le pire cas, l'opération de "bit stuffing" rajoute $\lfloor (34 + 8n) / 4 \rfloor$ bits. $\lfloor x \rfloor$ désigne la partie entière de *x*.

2.2.6 Détection et gestion des erreurs

Le réseau CAN a été créé pour opérer dans des environnements difficiles/agressifs (l'automobile, les procédés industriels, etc.) et c'est pourquoi il comprend de nombreux mécanismes de détection d'erreurs de transmission causées par les différents bruits inhérents à ces environnements.
CAN implante cinq mécanismes de détection des erreurs : deux au niveau bits (contrôle de bit, contrôle du "bit stuffing") et trois au niveau trame (vérification du CRC, de la forme des trames et de l'acquittement).
La norme CAN établit une distinction entre les erreurs temporaires et les erreurs permanentes sur le bus.

■ Types d'erreurs

Les cinq types d'erreurs pouvant être détectées sont les suivants :

- *Erreur de bit* : un nœud envoyant un bit sur le bus "observe" aussi en même temps le bit qu'il reçoit. Il détecte une erreur 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 la fenêtre d'acquittement (c'est-à-dire l'intervalle de temps pendant lequel le bit ACK peut être modifié par les récepteurs).
- *Erreur de stuffing* : un nœud détecte une erreur de stuffing lorsqu'il reçoit six bits consécutifs identiques.
- *Erreur de CRC* : une erreur de CRC est détectée lorsque le CRC calculé par un récepteur est différent de la valeur du CRC contenu dans la trame reçue.
- *Erreur d'acquittement* : un nœud émetteur détecte une erreur d'acquittement lorsqu'il ne reçoit pas de bit dominant pendant la fenêtre d'acquittement.
- *Erreur de forme* : une erreur de forme est détectée lorsqu'un bit qui devrait être à une certaine valeur est à une valeur différente (un délimiteur de CRC ou une fin de trame qui ne contiennent pas les bonnes valeurs de bits, par exemple).

■ Gestion et confinement des erreurs

Le confinement des erreurs est un mécanisme permettant de faire la différence entre des erreurs temporaires et les erreurs permanentes. Les erreurs temporaires peuvent être causées par des bruits transitoires, tandis que des erreurs permanentes sont en général dues à de mauvaises connexions ou à des composants défaillants. Cette distinction d'erreurs permet de retirer un nœud défaillant du bus qui sinon aurait pu perturber les autres nœuds.

Pour gérer de manière efficace les erreurs, CAN prévoit trois états de fonctionnement pour un nœud :

- *Etat d'erreur actif* : un nœud se trouvant dans l'*état d'erreur actif* peut prendre part normalement dans la communication sur le bus. Il transmet un *drapeau d'erreur actif* (constitué de 6 bits dominants consécutifs) s'il détecte une condition d'erreur. Un nœud qui est dans l'*état d'erreur actif* est considéré comme peu ou pas du tout perturbé par les erreurs.
- *Etat d'erreur passif* : un nœud se trouvant dans l'*état d'erreur passif* peut prendre part dans la communication, mais s'il détecte une condition d'erreur sur le bus, il transmet un *drapeau d'erreur passif* (constitué de 6 bits récessifs consécutifs). Cet état indique un nœud à problèmes. Un nœud qui est dans l'*état d'erreur passif* est considéré comme perturbé par les erreurs.

- *Etat d'erreur déconnecté* : un nœud se trouvant dans l'*état d'erreur déconnecté* n'est pas autorisé à avoir une quelconque influence sur le bus ; le nœud est considéré comme étant tellement perturbé par les erreurs qu'il doit cesser d'intervenir sur le bus. Ce nœud peut réintégrer le bus quand le niveau supérieur (sa couche application) lui demande de repasser en mode de fonctionnement normal et après avoir observé, sans erreur sur le bus, 128 occurrences de 11 bits consécutifs récessifs lui indiquant que le bus a retrouvé son état de fonctionnement normal.

Pour gérer les erreurs, deux compteurs d'erreurs sont implantés dans chaque nœud : un compteur en transmission, *CptTx* (ce compteur est utilisé pour noter ce qui se passe pendant que le nœud émet une trame) et un compteur en réception *CptRx* (ce compteur est utilisé pour noter ce qui se passe pendant que le nœud est récepteur).

Un nœud est dit *Emetteur* quand il transmet une trame (de données ou de requête) et il est dit *Récepteur* quand il reçoit la trame émise par un autre nœud. Un nœud Emetteur reste Emetteur jusqu'à ce que le bus soit libre ou qu'il ait perdu l'arbitrage au profit d'un autre nœud. Un nœud Récepteur reste Récepteur jusqu'à ce que le bus devienne libre.

Les règles de modification des compteurs d'erreurs sont les suivantes :

1. Si un récepteur détecte une erreur, son compteur *CptRx* est incrémenté de 1, sauf si l'erreur détectée est une *erreur de bit* pendant l'envoi d'un *drapeau d'erreur actif* ou un *drapeau de surcharge*.
2. Si un récepteur détecte un bit dominant comme premier bit après émission d'un drapeau d'erreur, son compteur *CptRx* est incrémenté de 8.
3. Si un émetteur envoie un drapeau d'erreur, son compteur *CptTx* est incrémenté de 8, sauf dans les deux cas suivants où le compteur *CptTx* n'est pas modifié : a) si l'émetteur est dans l'*état d'erreur passif* et il détecte une *erreur d'acquittement* parce qu'il n'a pas détecté un bit ACK dominant et ne détecte pas de bit dominant pendant l'envoi de son *drapeau d'erreur passif* et b) si l'émetteur envoie un drapeau d'erreur parce qu'il a détecté une *erreur de stuffing* pendant la phase d'arbitrage.
4. Si un émetteur détecte une *erreur de bit* alors qu'il est en train d'émettre un *drapeau d'erreur actif* ou un *drapeau de surcharge*, son compteur *CptTx* est incrémenté de 8.
5. Si un récepteur détecte une *erreur de bit* pendant l'envoi d'un *drapeau d'erreur actif* ou un *drapeau de surcharge*, son compteur *CptRx* est incrémenté de 8.
6. Chaque nœud tolère jusqu'à sept bits consécutifs dominants après avoir émis un *drapeau d'erreur actif*, un *drapeau d'erreur passif* ou un *drapeau de surcharge*. Après détection du 14ème bit consécutif dominant (suivant un *drapeau d'erreur actif* ou un *drapeau de surcharge)* ou après détection du 8ème bit dominant consécutif (suivant un *drapeau d'erreur passif*) et après chaque séquence additionnelle de huit bits consécutifs dominants, chaque émetteur incrémente son compteur *CptTx* de 8 et chaque récepteur incrémente son compteur *CptRx* de 8.
7. Après une transmission de trame réussie (ACK reçu et pas d'erreur jusqu'à la fin de la trame), le compteur *CptTx* est décrémenté de 1, sauf s'il est déjà nul.
8. Après une réception de trame réussie (réception correcte et émission de bit ACK), le compteur *CptRx* est décrémenté de 1, si sa valeur était comprise entre 1 et 127. Si le compteur était nul, il reste à zéro. Si sa valeur était supérieure à 127, il prend (le choix est aléatoire) une valeur entre 119 et 127.

REMARQUE : Les compteurs en transmission et en réception croissent et décroissent selon les occurrences d'erreurs. La quantité d'incrémentation des compteurs est plus importante que celle de leur décrémentation, car une erreur n'a pas les mêmes conséquences sur l'état d'un nœud qu'une trame bien émise ou bien reçue. Ainsi, sur une longue période, les compteurs peuvent avoir des valeurs élevées, même s'il y a eu plus de trames correctes que d'erreurs ; dans ce cas les compteurs reflètent bien l'état du réseau. Le rapport augmentation/diminution des compteurs a été fixé à 8 dans CAN.

Comme le montre la figure 8, les changements d'état d'un nœud se fait selon les règles suivantes :

1. A l'initialisation (ou réinitialisation), le nœud se trouve dans l'*état d'erreur actif* et ses deux compteurs sont nuls.
2. Un nœud passe à l'*état d'erreur passif* quand l'un de ses deux compteurs *CptTx* ou *CptRx* dépasse la valeur 127.
3. Un nœud passe à l'*état d'erreur déconnecté*, quand son compteur *CptTx* dépasse la valeur 255.
4. Un nœud passe à l'*état d'erreur passif*, quand ses deux compteurs *CptTx* et *CptRx* passent sous la valeur 128.
5. Un nœud qui est dans l'*état d'erreur déconnecté* passe à l'*état d'erreur actif* quand il aura vu passer 128 occurrences de 11 bits consécutifs récessifs. Dans ce cas il remet à zéro ses deux compteurs.

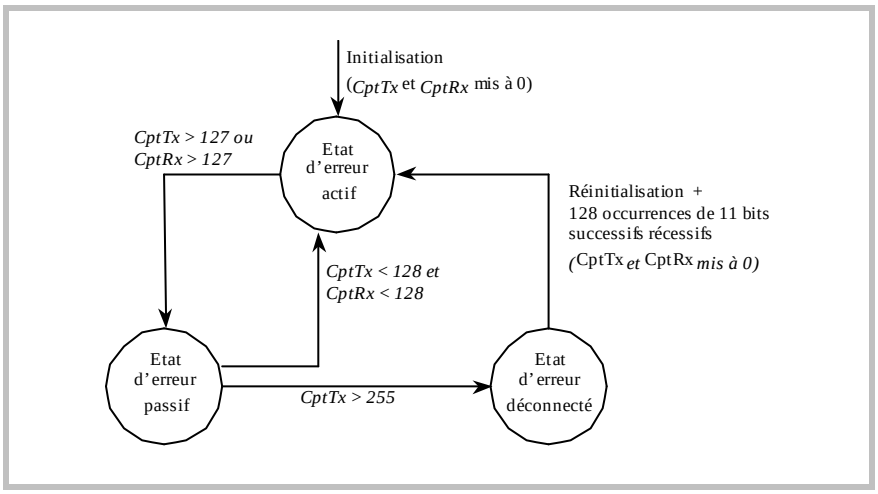


Figure 8 - Diagramme d'états de nœud en fonction de l'apparition d'erreurs.

REMARQUES

1. Le temps de recouvrement (temps entre le moment où une erreur est détectée et le moment où redémarre la transmission d'une nouvelle trame) est au maximum égal au temps de transmission de 29 bits pour CAN 2.0A et 31 bits pour CAN 2.0B, dans le cas où il n'y a pas d'autres erreurs détectées.
2. Le confinement d'erreur de CAN permet - contrairement à la plupart des autres réseaux de terrain - de localiser et d'isoler plus facilement les éléments défaillants.

■ Trames d'erreurs

Lorsqu'un nœud détecte une erreur (et s'il est dans un état lui permettant d'agir sur le bus), il interrompt immédiatement la trame en cours et envoie un drapeau d'erreur (un drapeau actif ou passif selon son état) pour signaler la situation aux autres nœuds.

Une trame d'erreur est constituée de deux parties (*figure 9*). La première est formée par la superposition des différents drapeaux d'erreurs émis par les nœuds du bus qui peuvent en émettre. La seconde partie est un délimiteur.

Un nœud qui détecte une erreur la signale aux autres en envoyant un drapeau d'erreur. Celui-ci viole la règle du "bit stuffing" (car il envoie 6 bits consécutifs identiques) et par conséquent, tous les autres nœuds détectent aussi une erreur et commencent à envoyer un drapeau d'erreur. La séquence de bits dominants qui existe alors sur le bus est le résultat de la superposition de plusieurs drapeaux d'erreurs ; sa longueur varie entre 6 et 12 bits.

Le délimiteur d'erreur est composé de 8 bits récessifs. En fait, après avoir transmis son drapeau d'erreur, chaque nœud envoie des bits récessifs et observe le bus jusqu'à ce qu'il détecte un bit récessif, après quoi il envoie encore 7 bits récessifs supplémentaires.

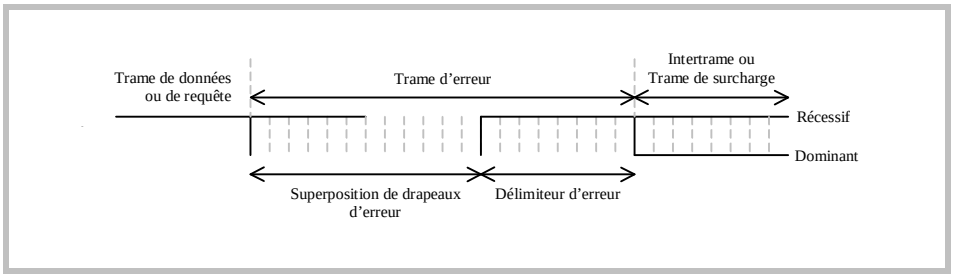


Figure 9 - Format de trame d'erreur.

2.2.7 Trames de surcharge

Les conditions internes d'un nœud peuvent le conduire à demander un certain temps ("une pause") pour accepter la prochaine trame de données ou de requête en provenance des autres nœuds. Il peut faire cette demande en envoyant une *trame de surcharge*. L'émission automatique de trames de surcharge par les nœuds saturés assure en quelque sorte le contrôle de flux.

Une trame de surcharge ne peut se produire qu'à la fin d'une trame normale ou d'erreur ou d'une autre trame de surcharge. Elle remplace l'intertrame. La norme CAN autorise deux (au maximum) trames de surcharge consécutives, pour éviter de bloquer indéfiniment le bus.

Une trame de surcharge ne contient que deux champs : un champ contenant des drapeaux de surcharge et un délimiteur de champ. Un drapeau de surcharge est constitué de 6 bits dominants. Le délimiteur de surcharge est constitué de 8 bits récessifs.

3. Implantation

3.1 Architectures autour de CAN

Les trois constituants de base d'un nœud CAN sont le transceiver, le contrôleur CAN et le microcontrôleur. Le *transceiver* (Transmetter-receiver) est une interface de raccordement au bus. Cette interface est aussi appelée MAU (Medium Access Unit). Le contrôleur CAN est l'entité qui gère le protocole CAN. Le microcontrôleur est l'entité sur laquelle est implantée l'application de l'utilisateur.

Etant donnée la popularité de CAN (si l'on juge par le nombre d'applications et de domaines d'applications qui l'ont choisi), beaucoup de fondeurs de silicium (ou fabricants de circuits intégrés) proposent de larges gammes de circuits pour monter des applications utilisant le protocole CAN. On distingue essentiellement trois types d'architectures :

- Les architectures fondées sur l'utilisation de contrôleurs CAN autonomes (*figure 10*). Les contrôleurs CAN autonomes sont conçus pour s'interfacer avec une large gamme de processeurs, ce qui permet de réutiliser le logiciel développé.
- Les architectures où le contrôleur CAN et le microcontrôleur sont intégrés dans un même boîtier ou une même carte électronique (*figure 11*). Les contrôleurs intégrés sont surtout attractifs par leur prix et leur performance ; la fonction de contrôle de CAN et l'application partagent les mêmes registres et la même mémoire, ce qui permet d'accélérer les transferts de données entre les deux fonctions.
- Les architectures où tout le nœud est intégré dans une seule carte (*figure 12*). La tendance de tout intégrer dans un même circuit (le transceiver, le contrôleur et le microcontrôleur) devient de plus en plus une réalité. Avec une telle technologie, l'ensemble d'un nœud CAN est implanté sur un seul circuit, ce qui réduit à la fois le coût et l'encombrement qui son deux facteurs essentiels pour les applications ciblées par CAN (en particulier l'automobile).

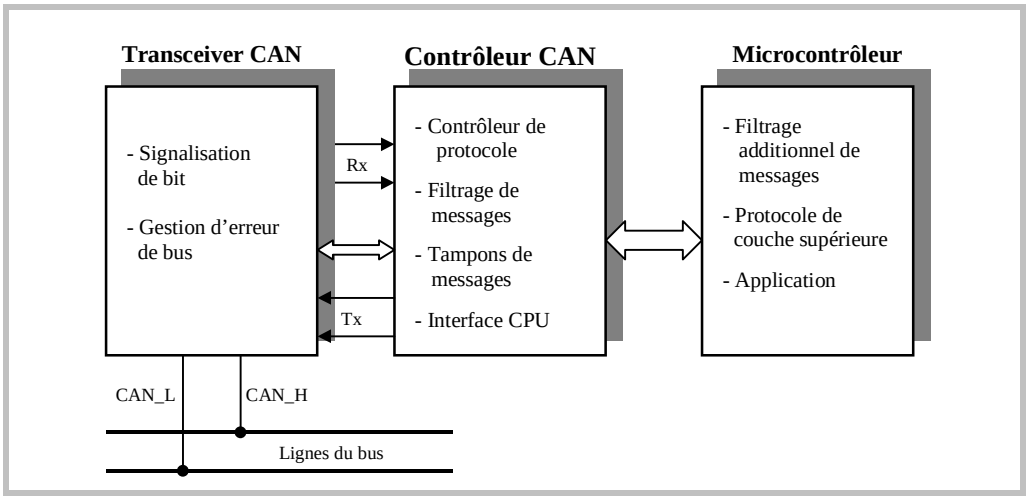


Figure 10 - Architecture à base de contrôleur CAN autonome.

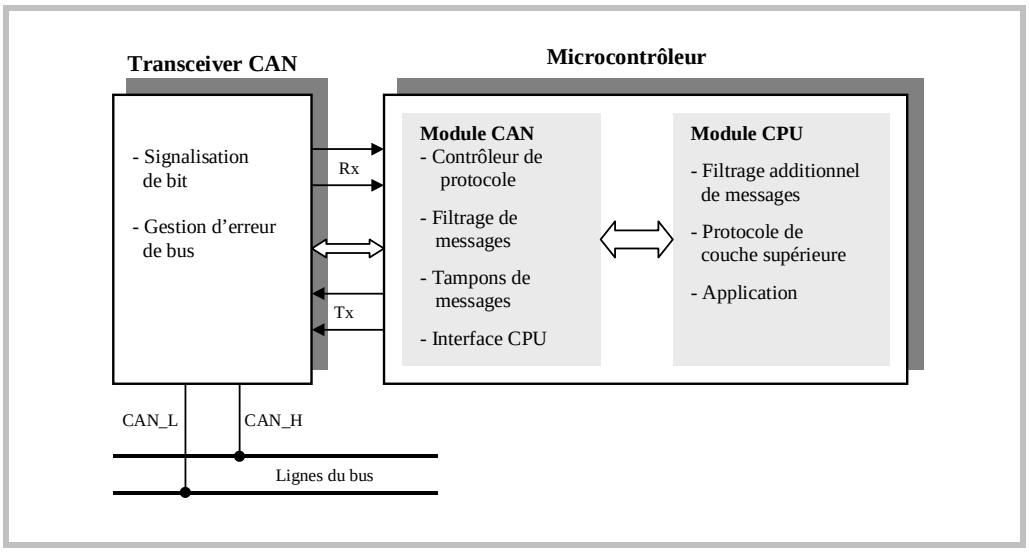


Figure 11 - Architecture à base de contrôleur CAN intégré.

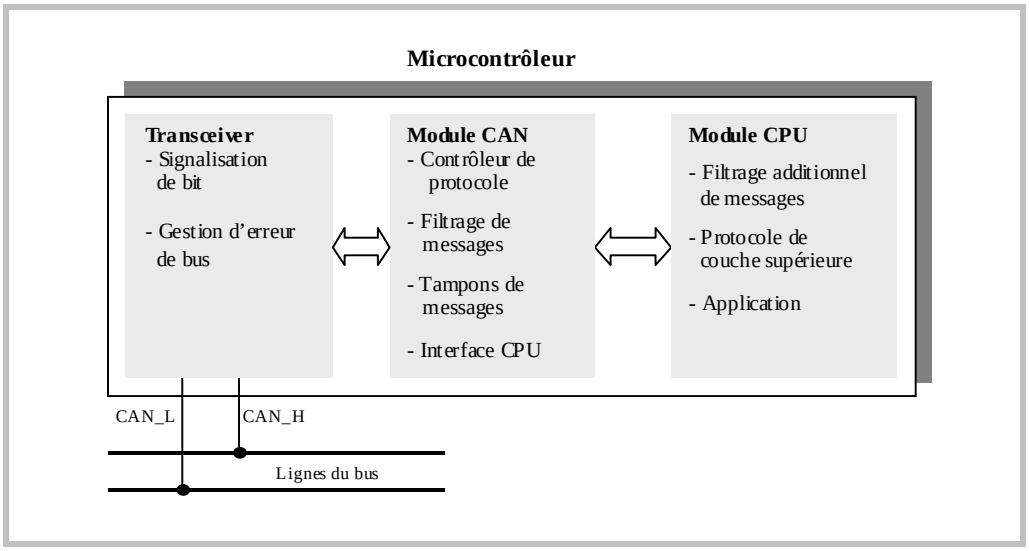
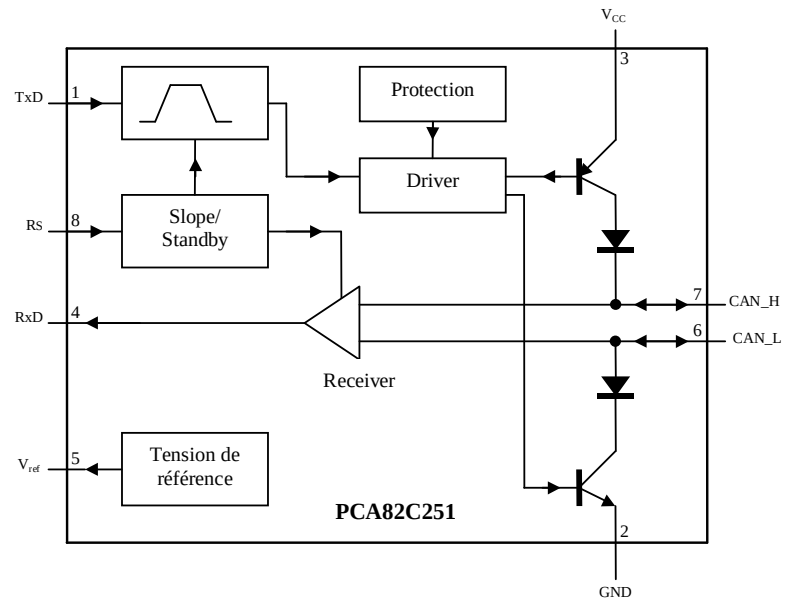


Figure 12 - Architecture à base d'un nœud CAN monolithique.

3.2. Exemples de composants CAN

■ Transceiver PCA82C251 de Philips

Il constitue une interface entre le contrôleur de protocole CAN et la ligne physique (le bus). Il est compatible à la norme ISO 11898 et est conçu pour supporter des débits jusqu'à 1 Mb/s. Il offre une très bonne immunité contre les interférences électromagnétiques dans les systèmes 24 V. Il peut fonctionner sous des températures de – 40 à +125 °C. L'architecture du transceiver est donnée par la figure 13.



TxD : Entrée de données à transmettre — RxD : Sortie de données reçues
GND : Masse — V_{CC} : Tension d'alimentation
Rs : Résistance — V_{ref} : Sortie de tension de référence
CAN_L : Niveau bas de tension CAN — CAN_H : Niveau haut de tension CAN

Figure 13 - Diagramme du Transceiver PCA82C251 de Philips.
(Doc. Philips Semiconducteurs – janvier 2000)

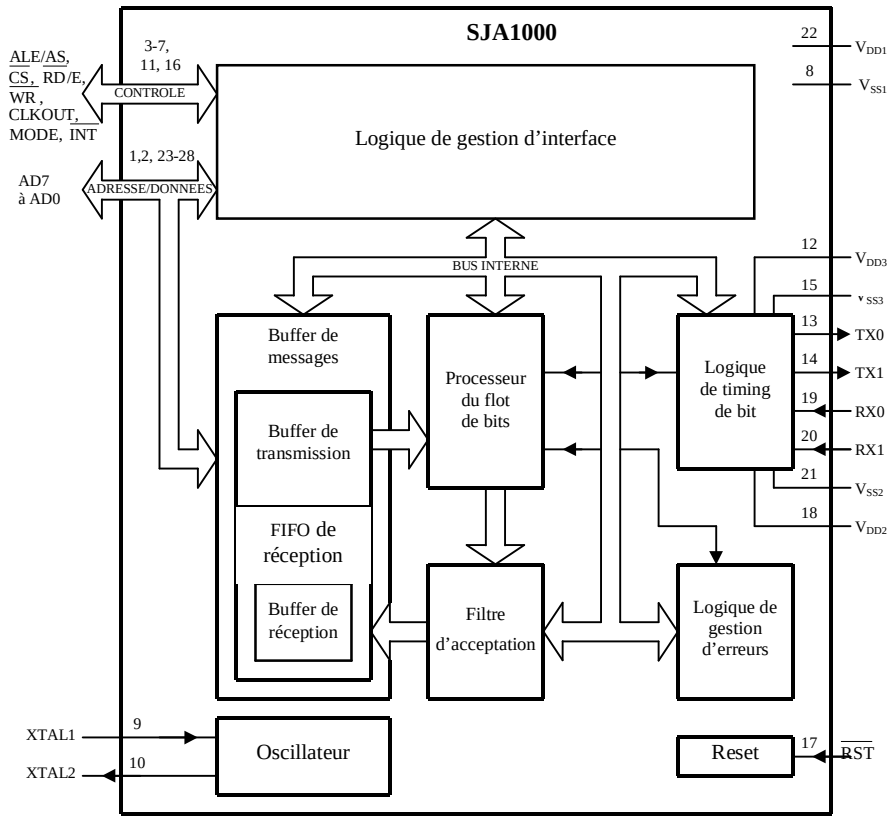
■ Contrôleur CAN SJA1000 de Philips

Le SJA1000, développé par Philips, est un contrôleur CAN autonome qui ne nécessite qu'un microcontrôleur externe. Ses principales caractéristiques sont les suivantes :

- Supporte le CAN 2.0A et 2.0B ;
- Débit jusqu'à 1Mbit/s ;
- Buffer de réception de 64 octets ;
- Compteurs d'erreur avec accès lecture/écriture ;
- Interruptions pour chaque type d'erreur sur le bus ;
- Capture de la dernière erreur détectée ;
- Détails sur le bit d'identificateur ayant causé une perte d'arbitrage ;
- Mise à l'état d'écoute du bus (pas de transmission d'aucun signal) ;
- Adapté à des microprocesseurs variés ;
- Température de fonctionnement comprise entre -40 à +125 °C.

De manière sommaire les différents blocs du SJA1000 présentés par la figure 14 assurent les fonctions suivantes :

- *Logique de gestion d'interface* : elle contrôle l'adressage des registres et fournit les interruptions et des informations d'état au microcontrôleur.
- *Buffer de transmission :* il permet de stocker le message à transmettre.
- *Buffer de réception* : il permet de stocker le dernier message reçu et accepté.
- *FIFO de réception* : c'est une file qui permet de stoker plusieurs messages reçus en attendant leur traitement.
- *Filtre d'acceptation* : il compare les identificateurs reçus avec ceux que le nœud souhaite traiter (car il est soit producteur soit consommateur de ces identificateurs) pour décider de l'acceptation ou du rejet des messages.
- *Processeur du flot de bits* : c'est un séquenceur qui contrôle le flot des bits entre le buffer de transmission et le bus CAN. Il assure aussi les fonctions de détection et gestion d'erreurs, d'arbitrage et de "bit stuffing".
- *Logique de timing de bit* : Elle supervise la ligne du bus et se synchronise sur le flot de bits à partir du premier bit dominant.
- *Logique de gestion d'erreurs* : Elle est responsable du confinement d'erreurs.



AD0 à AD7 : bus d'adresses/données
ALE/AS : Signal d'entrée ALE (mode Intel) ou signal AS (mode Motorola)
$\overline{CS}$: Entrée de sélection du circuit
$\overline{RD}$ /E : Signal $\overline{RD}$ (mode Intel) ou signal E (mode Motorola)
$\overline{WR}$: Signal $\overline{WR}$ (mode Intel) ou signal $\overline{RD}$ / $\overline{WR}$ (mode Motorola)
CLKOUT : signal de sortie d'horloge
XTAL1 : entrée de l'amplificateur de l'oscillateur
XTAL2 : sortie de l'amplificateur de l'oscillateur
Mode : sélection du mode Intel ou Motorola
TX0, TX1 : sortie du driver de CAN vers la ligne de transmission
RX0, RX1 : Entrée en provenance de la ligne de transmission
$\overline{INT}$: sortie d'interruption
$\overline{RST}$: entrée de réinitialisation
V_{SS1}, V_{SS2}, V_{SS3} : masse
V_{DD1}, V_{DD2}, V_{DD3} : Alimentation (5V)

Figure 14 - Diagramme du contrôleur CAN SJA1000 de Philips.
(Doc. Philips Semiconducteurs – janvier 2000)

Calcul du pire temps de transfert de message périodique pour le réseau CAN

On considère n messages périodiques ayant comme périodes P_1, …, P_n et comme temps de transmission $TTransmission_1$, …, $TTransmission_n$. On associe des identificateurs (donc des priorités) aux messages en fonction de leur période. On suppose qu'il existe un mécanisme (basé sur une fonction aléatoire ou sur une fonction d'importance/criticité des messages) pour attribuer des priorités différentes à deux messages quand ils ont la même période.

Le temps de transfert (c'est-à-dire le temps d'attente dans la file de l'émetteur plus le temps de transmission) du message M_i dans le pire des cas est donné par la formule [1] :

$$TempsTransfert_i = \Delta attente_i + TTransmission_i \quad [1]$$

$$\Delta attente_i = B_i + \sum_{k \in PP(i)} \left\lceil \frac{\Delta attente_i + \tau bit}{P_k} \right\rceil * TTransmission_k \quad [2]$$

$$B_i \leq \max_{j \in MP(i)} \left(TTransmission_j \right) - \tau bit$$

Pour calculer le point fixe, on applique une procédure de récurrence :

Initialisation : $\Delta attente_i^0 = 0$

Arrêt quand : $\Delta attente_i^{n+1} = \Delta attente_i^n$

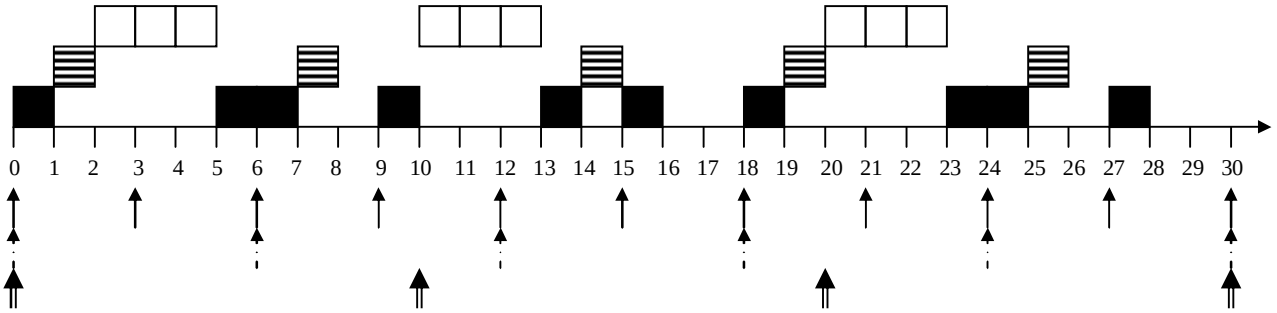
Condition de convergence : $\sum_{k=1}^{n} \frac{TTransmission_k}{P_k} \leq 1$

$TempsTransfert_i$: délai de transfert du message M_i
$\Delta attente_i$: délai d'attente du message M_i avant d'être transmis. Le message M_i attend la fin de transmission des messages plus prioritaires et éventuellement d'un message moins prioritaire mais qui a commencé sa transmission avant celle de M_i.
$TTransmission_i$: temps de transmission du message M_i (ce temps dépend de la taille du message et du débit)
B_i : Temps maximum de transmission d'un message moins prioritaire que le message M_i.
P_k : période du message M_k
$PP(i)$: ensemble des indices des messages plus prioritaires que le message M_i
$MP(i)$: ensemble des indices des messages moins prioritaires que le message M_i
τbit : délai de transmission d'un bit sur le bus CAN.

L'ensemble des n messages respecte les contraintes de transfert si et seulement si :

$$\forall i_{i=1,\ldots,n} \quad TempsTransfert_i < P_i \quad [3]$$

Exemple 1 :
Considérons trois messages M_1, M_2 et M_3 avec les périodes $P_1 = 3$, $P_2 = 6$ et $P_3 = 10$ et avec des temps de transmission $C_1 = 1$, $C_2 = 1$ et $C_3 = 3$.
On suppose ici que le temps de transmission d'un bit (τbit) est l'unité. Nous choisissons des messages très courts pour pouvoir représenter schématiquement leur transmission.



Sur le graphique, on a :
Délai d'attente de $M_1 = 2$; Délai d'attente de $M_2 = 2$; Délai d'attente de $M_3 = 2$.
On notera que la condition [3] est respectée dans cet exemple.

Application de la formule [2] :

$$B_1 = \max_{k \in MP(1)} \left(TTransmission_k\right) - \tau bit = 3 - 1 = 2$$

Donc $\Delta attente_1 = B_1 = 2$

$$B_2 = \max_{k \in MP(2)} \left(TTransmission_k\right) - \tau bit = 3 - 1 = 2$$

$$\Delta attente_2^0 = 0 \qquad \Delta attente_2^1 = B_2 + \left\lceil \frac{0+1}{P_1} \right\rceil * TTransmission_1 = 2 + 1*1 = 3$$

$$\Delta attente_2^2 = B_2 + \left\lceil \frac{3+1}{P_1} \right\rceil * TTransmission_1 = 2 + 2*1 = 4$$

$$\Delta attente_2^3 = B_2 + \left\lceil \frac{4+1}{P_1} \right\rceil * TTransmission_1 = 2 + 2*1 = 4$$

Donc $\Delta attente_2 = 4$

$B_3 = 0$ *(car M_3 est le moins prioritaire)*

$$\Delta attente_3^0 = 0$$

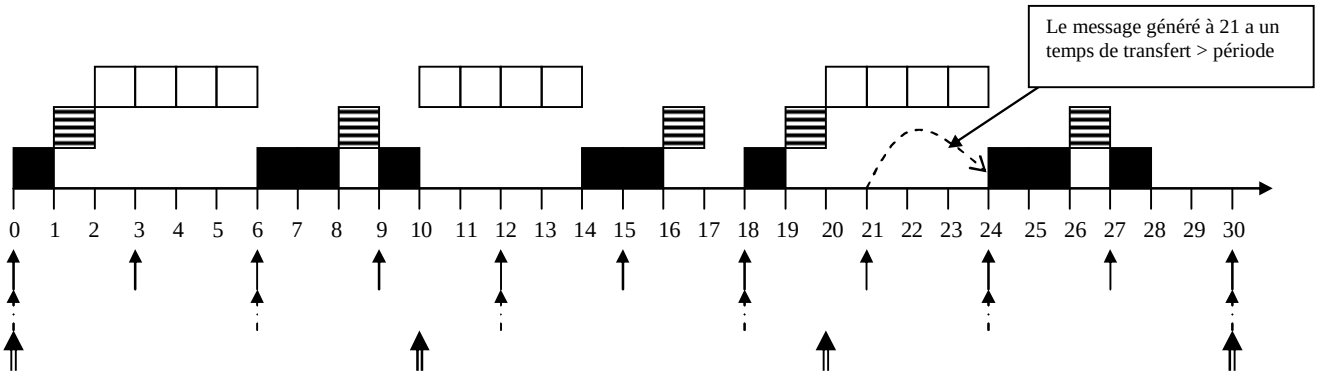
$$\Delta attente_3^1 = 0 + \left\lceil \frac{0+1}{P_1} \right\rceil * TTransmission_1 + \left\lceil \frac{0+1}{P_2} \right\rceil * TTransmission_2 = 1*1 + 1*1 = 2$$

$$\Delta attente_3^2 = 0 + \left\lceil \frac{2+1}{P_1} \right\rceil * TTransmission_1 + \left\lceil \frac{2+1}{P_2} \right\rceil * TTransmission_2 = 1*1 + 1*1 = 2$$

Donc $\Delta attente_3 = 2$

Remarque : le pire temps de transfert pour le message M_2 n'apparaît pas sur la figure précédente. Il apparaît si le message M_3 n'est pas tout à fait cyclique (par exemple, M_3 apparaît à t=0 et t=11).

Exemple 2
Considérons trois messages M_1, M_2 et M_3 avec les périodes $P_1 = 3$, $P_2 = 6$ et $P_3 = 10$ et avec des temps de transmission $C_1 = 1$, $C_2 = 1$ et $C_3 = 4$.



Sur le graphique, on a :
Délai d'attente de $M_1 = 3$; Délai d'attente de $M_2 = 4$; Délai d'attente de $M_3 = 2$;
On notera que la condition [3] n'est pas respectée dans cet exemple, car le temps de transfert de M_1 est égal à 4 (donc supérieur à la période P_1).

Application de la formule [2] :

$$B_1 = \max_{k \in MP(1)} \left(TTransmission_k\right) - \tau bit = 4 - 1 = 3$$

Donc $\Delta attente_1 = B_1 = 3$

$$B_2 = \max_{k \in MP(2)} \left(TTransmission_k\right) - \tau bit = 4 - 1 = 3$$

$$\Delta attente_2^0 = 0$$

$$\Delta attente_2^1 = B_2 + \left\lceil \frac{0+1}{P_1} \right\rceil * TTransmission_1 = 3 + 1 * 1 = 4$$

$$\Delta attente_2^2 = B_2 + \left\lceil \frac{4+1}{P_1} \right\rceil * TTransmission_1 = 3 + 2 * 1 = 5$$

$$\Delta attente_2^3 = B_2 + \left\lceil \frac{5+1}{P_1} \right\rceil * TTransmission_1 = 3 + 2 * 1 = 5$$

Donc $\Delta attente_2 = 5$

$B_3 = 0$ *(car M_3 est le moins prioritaire)*

$$\Delta attente_3^0 = 0$$

$$\Delta attente_3^1 = 0 + \left\lceil \frac{0+1}{P_1} \right\rceil * TTransmission_1 + \left\lceil \frac{0+1}{P_2} \right\rceil * TTransmission_2 = 1 * 1 + 1 * 1 = 2$$

$$\Delta attente_3^2 = 0 + \left\lceil \frac{2+1}{P_1} \right\rceil * TTransmission_1 + \left\lceil \frac{2+1}{P_2} \right\rceil * TTransmission_2 = 1 * 1 + 1 * 1 = 2$$

Donc $\Delta attente_3 = 2$