

MISE EN OEUVRE D'UN BUS DE TERRAIN CAN A PARTIR DE MICROCONTROLEURS HC12

Serge BOUTER¹ et Rachid MALTI^{1,2}

{serge.bouter, rachid.malti}@iut.u-bordeaux1.fr

¹IUT Bordeaux 1, 15 rue Naudet, CS 10207, 33175 GRADIGNAN Cedex

²IMS UMR 5218 CNRS, Département LAPS – Université Bordeaux I
351, cours de la Libération – F 33405 Talence cedex – France

RESUME : L'objectif de cette communication est de présenter des solutions matérielles et logicielles pour la mise en place d'une maquette pédagogique destinée à l'étude du bus de terrain CAN. La solution présentée est basée sur l'utilisation d'un starter kit du commerce incluant un microcontrôleur de type HC12. La mise en oeuvre matériel du réseau de terrain requiert aussi un circuit d'interface pour connecter les microcontrôleurs au bus CAN. Cette maquette présente l'avantage d'être facile à développer et d'être « bon marché ». Ce sujet concerne les étudiants de la Licence Systèmes Automatisés et Réseaux Industriels de l'IUT Bordeaux 1 qui le traite dans une séance de TP. Ils ont à développer des fonctions en C destinées à la configuration du bus CAN et aux opérations d'émission/réception. Cette séance de TP fait partie d'un module pratique concluant leur formation théorique

Mots clés : bus de terrain, CAN, microcontrôleur HC12, systèmes embarqués, réseaux locaux.

1 INTRODUCTION

La licence Système Automatisés et Réseaux Industriels consacre une partie de son enseignement aux bus de terrain et notamment à ceux qui ont pour vocation à être intégrés au sein de systèmes embarqués. C'est le cas du réseau de terrain CAN.

CAN (Controller Area Network) est initialement créé par Bosch et Intel pour des applications automobiles (Mercedes-Benz Classe S). CAN spécifie la partie physique du réseau, le contrôle de l'accès au support de transmission et la gestion des erreurs de transmission. Si au départ le bus CAN a été développé pour des systèmes embarqués (automobiles, machine agricoles, aéronautique...), il a été très rapidement adopté par le milieu des automatismes industriels.

Ainsi dans le département, le bus CAN est utilisé pour traiter aussi bien les automatismes que les systèmes embarqués.

2 QUELQUES SPÉCIFICITÉS DU BUS CAN

2.1 Généralités

Ce protocole est caractérisé par la diffusion d'informations d'un nœud (ou station) vers les autres. Cette information est associée à un identificateur « reconnu » par certains nœuds du réseau :

le nœud dont est issue cette information : **le producteur**

les nœuds nécessitant cette information pour leur traitement : **les consommateurs**.

Ainsi une trame (voir fig. 1) émise par un nœud ne contient pas véritablement d'adresses destination

ou/et source. L'identificateur associé à la donnée contenue dans la trame est analysé par tous les nœuds du réseau. Ces derniers récupèrent ou non cette trame.

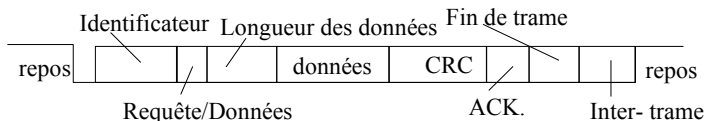


fig 1: trames CAN 2A

2.2 La couche physique

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

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

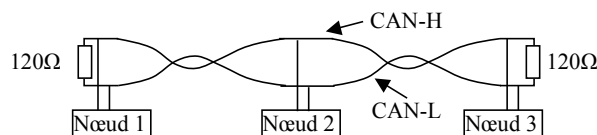


fig 2: topologie du bus CAN

CAN s'appuie sur le codage NRZ avec des niveaux électriques, pour CAN High Speed, suivants: état logique =0 ou état dit dominant:

$$V_{CAN_L} = 1V \quad \text{et} \quad V_{CAN_H} = 4V$$

état logique =1 ou état dit récessif:
 $V_{CAN_L} = 2,5V$ et $V_{CAN_H} = 2,5V$

2.3 Méthode d'accès au support de transmission

Le support de transmission est une ressource unique à partager par tous les noeuds. Les tentatives d'accès au support sont effectuées dès que ce dernier est libre de tout signal. Dans le cas d'émissions simultanées de trames (voir fig. 1), on parle de collision; il est alors nécessaire d'éviter une destruction de ces trames et d'assurer un arbitrage. L'arbitrage s'appuie sur l'évaluation des identificateurs commençant une trame.

Chaque nœud est constitué de deux fonctions: la génération du signal et l'écoute du signal sur le bus.

Chacun des noeuds débute son émission par l'identificateur qui est composée de bits dominants et de bits récessifs.

Le tableau ci-après donne l'état du bus lorsque deux noeuds émettent simultanément.

Bit émis par le noeud A	Bit émis par le noeud B	État du bus
récessif	récessif	récessif
récessif	dominant	dominant
dominant	récessif	dominant
dominant	dominant	dominant

Chaque émetteur « écoute » le bus. Si un noeud émet un bit récessif et si, dans le même intervalle de temps, il détecte un bit dominant, il bascule en réception et perd l'arbitrage. Ce noeud tentera un accès au bus de suite après la libération de ce dernier.

La « priorité » d'un identificateur dépend de sa valeur. Plus la valeur de l'identificateur est petite, plus sa priorité sera grande.

2.4 Transmission de l'information :durée nominale du bit ou Nominal Bit Time

Pour transmettre une trame CAN, le codage NRZ est utilisé. Ce dernier, en terme de transmission n'est pas très performant, notamment en ce qui concerne la synchronisation entre l'émetteur et les récepteurs. CAN a défini certains mécanismes pour fiabiliser la transmission. Une des méthodes utilisées est de prendre en compte les caractéristiques temporelles des composants du bus de terrain.

La « durée nominale du bit » est composée d'un ensemble d'intervalles de temps identiques appelés "quanta de temps". Le quantum de temps est réalisé à partir des oscillateurs internes de chaque nœud et la précision des oscillateurs influe sur la qualité de la « durée nominale du bit ». Aussi, pour tolérer des

fluctuations au niveau du bit; CAN a divisé ce dernier en plusieurs segments de temps:

- le segment synchronisation,
- le segment temps de propagation,
- le segment phase 1,
- le segment phase 2.

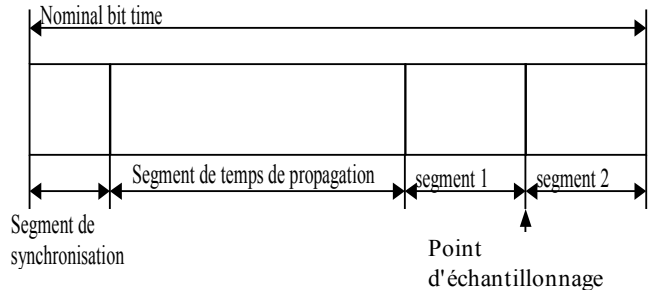


fig 3: composition du bit nominal

La « durée nominale du bit » doit comporter de 8 à 25 quanta de temps.

2.4.1 Segment de synchronisation

Durée = 1 quantum de temps.

Il s'agit de synchroniser les différents nœuds disposés sur le bus. « A la fin du bit en cours d'émission », les nœuds doivent « supposer » l'arrivée d'un nouveau. Cette appréciation se révèle arbitraire dans le cas où il n'y a pas de changement de valeur.

2.4.2 Segment de temps de propagation

Durée = la durée de ce segment dépend de la longueur du réseau. De 1 à 8 quanta de temps.

Cette partie prend en compte les retards dus à la propagation du signal sur un réseau donné (configuration géométrique, interfaces d'entrées/sorties des nœuds). Pour un fonctionnement correct, ce segment représente un temps aller/retour égal à la somme des temps de propagation sur le support et des retards apportés par les interfaces d'entrées et de sorties des nœuds.

2.4.3 Segments phase 1 et phase 2

Durée = de 1 à 8 quanta de temps.

Ces deux segments encadrent le point d'échantillonnage. Ils sont utilisés pour compenser les variations de phase des signaux. Ces phénomènes sont dus essentiellement aux horloges des nœuds du réseau.

2.4.4 Point d'échantillonnage

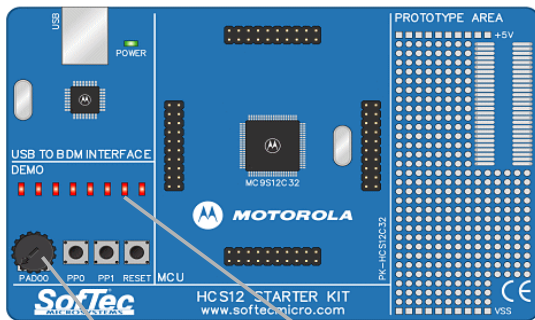
C'est l'instant où l'on échantillonne le signal présent sur le bus et par conséquent l'instant où l'on peut le lire, l'interpréter et définir la valeur du bit.

3 MAQUETTE PÉDAGOGIQUE

3.1 Starter Kit MC9S12C32 de Softec

Commercialisé par la société Softec pour la somme de 75 € ce starter kit composé de la plaquette hardware de la figure 3 et du logiciel de développement Metrowerks de Motorola permet de programmer le microcontrôleur HC12[2] ainsi que certains périphériques de base comme les ports d'entrée/sortie, les convertisseurs Analogique Numérique, Numérique Analogique, ... Il dispose également de plusieurs broches d'extension permettant de relier les différents modules du HC12[2] à des périphériques externes. Une miniplaquette de développement de prototype est disponible à cet effet.

Ce kit se connecte sur le port USB et ne nécessite pas d'alimentation supplémentaire extérieure.



Entrée analogique Sortie Tout-ou-Rien
fig 3: Kit SofTec MC9S12C32

3.2 Développement du circuit d'interface

Le contrôleur CAN[3] intégré au 68HC9S12C32 n'émet et ne reçoit que des signaux logiques du type HighSpeed CMOS. Aussi un circuit d'interfaçage (voir fig 4) est nécessaire entre le bus CAN et le contrôleur CAN.

Le circuit tel qu'il a été conçu permet d'éventuellement connecter la résistance de terminaison.

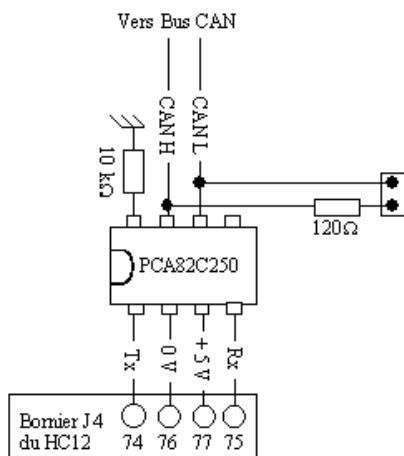


fig 4: circuit d'interfaçage

3.3 Organisation de la séance de travaux pratiques

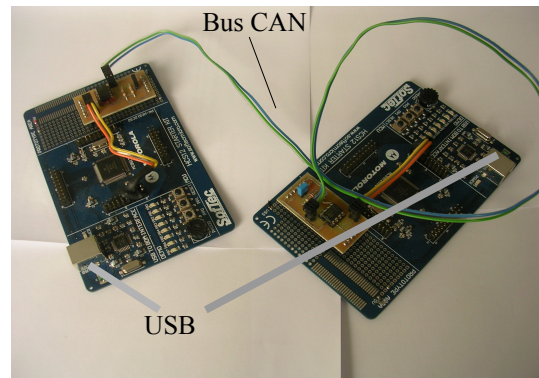


fig 5: carte HC12 connectées par un bus CAN

Cette séance de travaux pratiques dure quatre heures et est incluse dans un module pratique concluant leur formation théorique. En effet cette partie de l'enseignement a une structure classique: cours, travaux dirigés, travaux pratiques. Ainsi les étudiants possèdent tous les documents nécessaires pour préparer et traiter ce sujet.

Quant à l'organisation matérielle, le poste de travail comporte:

- les deux cartes HC12 (voir fig 5)
- 2 PCs équipés de liaison USB
- le logiciel de développement MétroWerks

Bien qu'il soit possible de réaliser cette séance de travaux pratiques avec un seul PC, l'utilisation de deux micro-ordinateurs facilite le travail des étudiants. En effet l'ouverture des deux projets HC12 peut entraîner rapidement des confusions et ralentir le travail des étudiants.

4 DÉVELOPPEMENT DES FONCTIONS DE CONTRÔLE DU BUS CAN

4.1 Objectifs

La mise en oeuvre du bus de terrain CAN reliant les deux microcontrôleurs passe par une série d'exercices. Il s'agit d'obtenir les fonctions permettant aux microcontrôleurs d'échanger des données. L'ensemble du programme est écrit en langage C. Les exercices proposés sont les suivants:

- la configuration des modules CAN pour répondre aux caractéristiques du bus.
- l'émission des trames CAN
- la réception de trames CAN
- le filtrage des trames à partir des identificateurs.

4.2 Initialisation du module CAN du microcontrôleur HC12

La configuration du contrôleur CAN porte essentiellement sur la durée du bit nominale et le

format et les filtres permettant le traitement des identificateurs des trames reçues.

La détermination de la « durée du bit nominale » requiert une caractérisation temporelle du réseau mis en place. Il s'agit de réaliser un inventaire des temps de propagation travers les composants les plus distants du bus de terrain. Cette partie est traitée en cours et en travaux dirigés. Il existe plusieurs documents qui traitent de la détermination du nombre de quanta de temps devant composer la « durée nominale du bit » [1][4]. Le bus mis en place étant de dimension modeste (moins de 20cm), il n'est pas nécessaire d'établir un inventaire « poussé » des retards et des temps de propagation sur les composants de l'installation. Les valeurs choisies pour le quantum de temps et les différents segments composant la « durée nominale du bit » sont conformes au standard CAN 2A. L'initialisation du module se déroule dans un mode de fonctionnement particulier du module CAN, qui est obtenu par une série d'écriture et de test sur des registres: CANCTL1 et CANCTL0

Activer le module MSCAN

CANCTL1_CANE ← 1

Entrer dans le mode SLEEP (préconisé par le constructeur):

CANCTL0_SLPRQ ← 1
 Tant que CANCTL1_SLPK ≠ 1
 //Attendre
 FinTant

Entrer dans le mode SOFT_RESET:

CANCTL0_INTRQ ← 1
 Tant que CANCTL1_INITAK1 ≠ 1
 //Attendre
 FinTant

Configurer la durée nominale d'un bit :

a) Choix de l'horloge: registre CANCTL1
 CANCTL1_CLKSCR ← 1 : horloge du bus interne,
 ou CANCTL1_CLKSCR ← 0 : oscillateur interne du microcontrôleur.

Dans le sujet, le choix s'est porté sur l'oscillateur interne du microcontrôleur qui offre une fréquence 8MHz.

b) configuration du quantum de temps à partir du pré-diviseur: registre CANTBR0

registre CANBTR0 ← valeur de prédivision

Le bus mis en place étant de dimension modeste, le pré-diviseur est positionné à 2. Ainsi le quantum de temps est de 0,25 µs.

c) Nombre de quanta de temps par bit: registre CANBTR1

SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11	TSEG10
------	--------	--------	--------	--------	--------	--------	--------

Le triplet (TSEG22, TSEG21, TSEG20) définit le nombre de quanta composant le segment de phase 2. Le quadruplet (TSEG13, TSEG12, TSEG11, TSEG10) définit le nombre de quanta composant les segments de phase 1 et de propagation.

Exemple de valeurs proposées aux étudiants
 segment de propagation = 1 quantum de temps
 segment phase 1 = 3 quanta de temps
 segment phase 2 = 4 quanta de temps

Le bit SAMP fixe le nombre de points d'échantillonnage (1: 3 points et 0: 1 point).

Configurer le mode de fonctionnement des filtres

Les trames reçues par un noeud sont placées dans un buffer dit d'arrière plan (il en existe plusieurs). Les identificateurs de ces trames sont analysés au moyen de filtres. Il s'agit de tester si la trame reçue concerne le noeud et si c'est le cas alors la trame passe dans un buffer de premier plan et peut être exploité par le programme en cours d'exécution. Les filtres peuvent être organisés de plusieurs façons :

-2 filtres de 32 bits (utilisés pour de trames CAN 2B,
 4 filtres de 16 bits, . . .
 8 filtres de 8 bits

c'est le registre CANIDAC qui permet de configurer le type de filtre souhaité.

Définir les filtres et les masques d'«acceptation »

Le mécanisme de filtrage repose sur 8 registres CANIDARx, les filtres d'acceptation et 8 registres CANIDMRx, les masques d'acceptation. Cette partie est développée dans le chapitre suivant.

Retourner dans le mode normal de fonctionnement

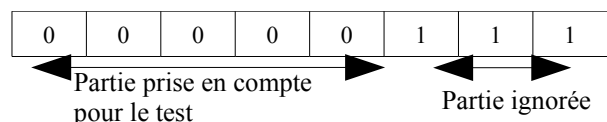
CANCTL0_INTRQ ← 0
 Tant que CANCTL1_INITAK1 ≠ 0
 //Attendre
 FinTant

4.3 Les filtres et les masques d'acceptation

Dans un premier, il s'agit de définir le mode de fonctionnement des filtres: tester les identificateurs sur 8 bits, 16 bits ou 32 bits? Le mode de fonctionnement choisi, il est possible d'affiner les parties de l'identificateur que l'on souhaite tester et ceci est obtenu au moyen des masques d'acceptation.

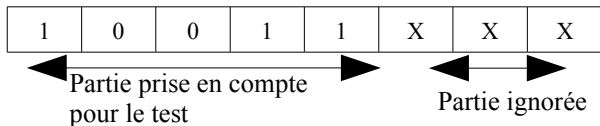
Exemple:

Supposons que le test porte sur les 5 bits les plus significatifs des identificateurs CAN 2A (11 bits). Les filtres et les masques d'acceptation sont positionnés sur un format 8 bits. Ainsi seuls les 8 bits les plus significatifs de l'identificateur sont testés. Les masques d'acceptation sont alors configurés de la façon suivante:



La valeur binaire 0 indique les bits qui doivent être pris en compte dans le test de l'identificateur. La valeur binaire 1 indique ceux qui sont ignorés.

Ensuite, les filtres d'acceptation sont configurés pour que les 5 bits les plus significatifs des identificateurs des trames acceptées aient la valeur (1,0,0,1,1).



Ainsi seules les trames ayant comme identificateur la combinaison (1,0,0,1,1,X,X,X,X,X) sont acceptées et passe le buffer de premier plan et peuvent être récupérées.

Masque d'acceptation

0	0	0	0	0	1	1	1
---	---	---	---	---	---	---	---

Filtre d'acceptation

1	0	0	1	1	X	X	X
---	---	---	---	---	---	---	---

Identificateur d'une trame acceptée

1	0	0	1	1	X	X	X	1	0	0
---	---	---	---	---	---	---	---	---	---	---

Identificateur d'une trame non acceptée

1	1	0	0	1	X	X	X	1	0	0
---	---	---	---	---	---	---	---	---	---	---

fig 6: mécanisme de filtrage des trames

4.4 Programme Emission

Il s'agit de constituer la trame CAN. Pour cela le module possède trois buffers TX0, TX1 et TX2. Chacun de ces buffers est constitué de registres représentant les éléments de la trame CAN. Ces buffers sont le point de départ de la transmission. Le contenu de ces buffer est ensuite « sérialisé ». La transmission terminée, ils sont considérés comme « vide ». L'intérêt de trois buffers est de pouvoir procéder à l'émission trois trames de façon consécutive sans attendre que l'un des buffer soit « vidé ». Ces trois buffers n'apparaissent pas directement sur la cartographie mémoire du microcontrôleur. Les adresses disponibles ne référence qu'un buffer. Il est nécessaire de sélectionner le buffer au moyen des bits TX0, TX1, TX2 du registre CANTBSEL.

Identificateur: les registres CANTXIDRx

Au nombre de quatre, ces registres stockent des identificateurs CAN 2A(11 bits) et CAN 2B(29 bits).

IDR0	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3
IDR1	ID2	ID1	ID0	RTR	IDE			
IDR2								
IDR3								

1
2

Remarques:

1) Le bit RTR distingue les trames de données des trames de requête. Le sujet ne traite que les trames de données et le bit RTR est positionné à 0.

2) Le bit IDE distingue les identificateurs CAN2A et CAN2B. Le bit IDE est positionné à 0 pour ne traiter que les trames CAN2A.

Les zones grisées sont destinées pour CAN2B et ne sont pas utilisées dans le sujet.

Données: les registres CANTXDSRx

Il y a huit registres qui permettent d'ajuster les champs « données » d'un à huit octets.

Longueur des données: CANTXDLR

Ce registre renseigne le champ « longueur des données » de la trame et indique au module CAN du microcontrôleur le nombre de registres CANTXDSRx à transmettre.

Procédure d'émission

Pour écrire dans un des buffers de transmission, il est nécessaire de sélectionner le buffer, puis de vérifier l'état de ce dernier. L'état de la transmission au niveau d'un buffer est donné le bit TXEx du registre CANTFLG. L'algorithme ci-dessous montre les opérations à réaliser pour transmettre une trame sur le bus à partir du buffer TX0.

```

/ *** Sélection du buffer de transmission TX0 ***/
CANTBSEL_TX0 ←1;
Tant que CANTFLG_TXE0 = 0
    // Attendre
FinTant
/ *** CANTFLG_TXE0 = 1 →buffer vide ***/
/ *** identificateur ***/
CANTXIDR0 ←valeur1
CANTXIDR1 ←valeur2 // RTR =0 et IDE = 0
/ *** données ***/
CANTXDSR0 ←donnée1
CANTXDSR1 ←donnée2
..
CANTXDLR ←longueur des données en octets
/ *** Reset du flag TXE0 et Transmission ***/
CANTFLG_TXE0 ←1
...

```

4.5 Programme Réception

Toutes les trames qui transitent sur le bus sont stockées dans des buffers dits « d'arrière-plan ». Seules les trames dont les identificateurs ont passé le « barrage » des filtres d'acceptation sont recopiées dans le buffer de « premier plan » (ou de réception) dont le contenu peut être récupéré par le programme. Le buffer de réception est composé de registres représentant les éléments d'une trame CAN. Son organisation est identique aux buffers de transmission.

Organisation du buffer de réception

- Identificateur: 4 registres CANRXIDRx
- Données: 8 registres CANRXDSRx
- longueur des données: CANRXDLR

Procédure de Réception

Pour lire le buffer de réception, il est nécessaire de vérifier l'état de ce dernier. L'état de la réception au niveau d'un buffer est donné le bit RXF du registre CANRFLG. L'algorithme ci-dessous montre les opérations à réaliser pour récupérer la trame reçue.

```
Tant que CANRFLG_RXF = 0
    // Attendre la réception d'une trame acceptée
FinTant
/**/ CANRFLG_RXF = 1 //buffer plein ***/
/**/ identificateur ***/
identificateur ←calcul(CANRXIDR0,CANRXIDR1)
/**/ données ***/
donnée(0) ←CANRXDSR0
donnée(1) ←CANRXDSR1
...
longueur des données en octets ←CANRXDLR
/**/ Reset du flag RXF ***/
CANRFLG_RXF ←1
...
```

A ce stade du projet, les étudiants ont écrit deux applications: émission et réception. Sur l'application « réception », il est nécessaire de neutraliser les filtres d'acceptation. Pour cela, tous les bits des masques d'acceptation doit être positionnés à 1. Ainsi le test sur chacun des bits des identificateurs est ignoré. Le filtrage des trames est traité dans le paragraphe suivant.

4.6 Les masques et les filtres d'acceptation

Dans le sujet proposé aux étudiants, il est demandé d'obtenir un filtrage complet d'un identificateur CAN2A (11 bits). Par conséquent un filtre sur 16 bits est nécessaire.

mode de fonctionnement des filtres: registre CAN

La sélection du mode de fonctionnement des filtre est obtenu en configurant les bits IDAM0 et IDAM1 du registres CANIDAC. La configuration choisie est celle sur-lignée dans le table ci-dessous.

IDAM1	IDAM0	Mode
0	0	2 filtres 32 bits
0	1	4 filtres 16 bits
1	0	8 filtres 8 bits
1	1	Pas de filtre

Ainsi le filtrage des identificateurs s'opère comme le montre la figure 7.

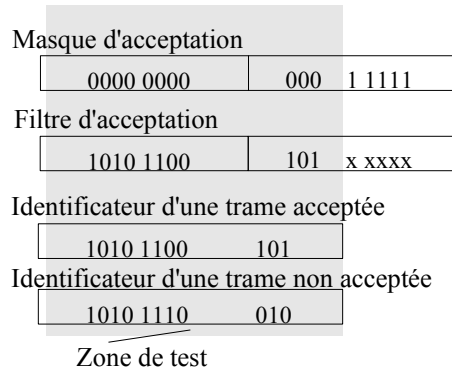


fig 7: filtrage sur des identificateurs CAN2A

masque d'acceptation

```
//filtre n°1
CANIDMR0 ←0x0;
CANIDMR1 ←0x1f;
//filtre n°2
CANIDMR2 ←0x0;
CANIDMR3 ←0x1f;
//filtre n°3
CANIDMR4 ←0x0;
CANIDMR5 ←0x1f;
//filtre n°4
CANIDMR6 ←0x0;
CANIDMR7 ←0x1f;
```

filtre d'acceptation

```
//filtre n°1, valeur d'identificateur acceptée
CANIDAR0 ← 0xAC;
CANIDAR1 ← 0;
//filtre n°2, valeur d'identificateur acceptée
CANIDAR0 ← ...;
CANIDAR1 ← ...;
```

...
L'écriture de filtres sur des identificateurs CAN2A impose le bit IDE à 0.

5 CONCLUSION

Ce sujet permet de montrer clairement aux étudiants la mise en oeuvre matérielle d'un bus CAN (microcontrôleur, interface...). Il permet aux étudiants de réaliser la mise en oeuvre logicielle de ce bus de terrain. D'autres points peuvent être traités tels que les interruptions en réception ou en transmission. Quant à la maquette, elle peut facilement être intégrée dans des projets incluant des réseaux de terrain.

6 RÉFÉRENCES BIBLIOGRAPHIQUES

- [1] Le bus CAN, Dominique Paret, Dunod, 1999
- [2] MSCAN Block Guide V02.15, Motorola, 1998
- [3] AN3034 Using MSCAN on the HCS12 Family Motorola, 2004
- [4] AN 1798 CAN Bit Timing Requirements, Motorola