

**TP INFORMATIQUE INDUSTRIELLE**  
**MP13 : LE BUS CAN**

## LE BUS CAN

### 1) But

Le bus CAN est de plus en plus utilisé dans les systèmes d'automatismes industriels et a pris un essor considérable dans le domaine de l'automobile. Le but de ces travaux pratiques est d'étudier la forme des trames qui circulent sur le bus CAN et de caractériser celles-ci.

### 2) Objectifs

L'étudiant devra être capable :

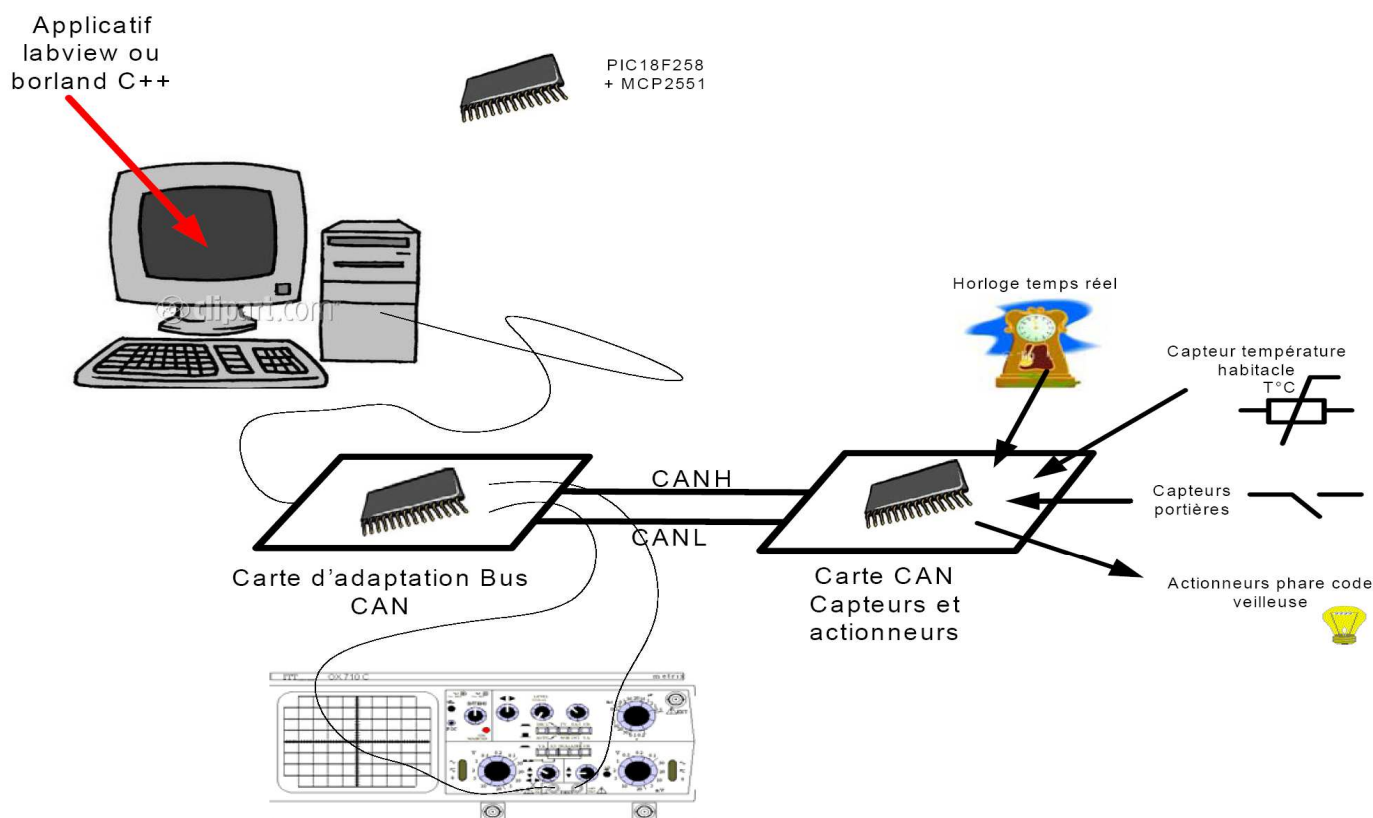
- De décrire la forme d'une trame qui circule sur le BUS CAN
- D'exploiter la documentation technique fournie en commentant certains éléments logiciels

Pour cette séance on utilisera 2 maquettes didactiques qui mettent en œuvre le BUS CAN. Ces 2 maquettes sont décrites dans un dossier technique ( voir document annexe1). Celles-ci disposent de point de mesure nous permettant de relever les trames qui circulent sur le BUS CAN.

### 3) Pré-requis

- Cours sur le BUS CAN.
- Lecture du dossier technique des maquettes didactiques

### 4) Schéma de câblage





Cliquez sur le bouton récupérer l'heure et relevez les chronogrammes de TXD et RXD de la carte hôte( l'oscilloscope déclenchera en mode simple coup ).

Vérifiez la présence des 2 trames ( la requête et la réponse de la carte cible)

Interprétez la trame réponse obtenue en identifiant :

- le champ identificateur
- le champ requête d'arbitrage en le justifiant (voir dossier technique)
- le champ de commande en le justifiant (voir dossier technique)
- le champ de donnée en le justifiant
- le champ de CRC

Remarque : déclenchement de l'oscilloscope mode simple coup voie RXD front descendant (base de temps 100µS, délai -184µS).

Quelle est le format des informations heure, minutes, secondes obtenues sur le champ donnée de la trame.

Expliquez sommairement le programme partiel en C de gestion du circuit temps réel (voir annexe 2)

Justifiez l'utilisation de la fonction rtc\_reg2bin document annexe2 et expliquez succinctement son fonctionnement en prenant un exemple, pourquoi avoir choisi un tel format ? .

### **5.2.3) Etude de la conversion analogique numérique**

Cliquez sur le bouton acquérir et relevez les chronogrammes de TXD et RXD de la carte hôte( l'oscilloscope déclenchera en mode simple coup ).

Vérifiez la présence des 2 trames ( la requête et la réponse de la carte cible)

Interprétez la trame réponse obtenue en identifiant :

- le champ identificateur
- le champ requête d'arbitrage en le justifiant (voir dossier technique)
- le champ de commande en le justifiant (voir dossier technique)
- le champ de donnée en le justifiant
- le champ de CRC

Remarque : déclenchement de l'oscilloscope mode simple coup voie RXD front descendant (base de temps 100µS, délai , -400µS).

### **5.2.4) Déconnexion du BUS**

Déconnectez un des fils du bus CAN, agir sur le logiciel pc2can, que remarquez vous à l'écran et pourquoi ?

Relevez les chronogrammes de TXD et RXD de la carte hôte, justifiez les chronogrammes obtenus.

### **5.2.5) Préparation**

Lire le dossier technique (document annexe).

Répondre aux questions préliminaires.

## Annexe 1 : Descriptif des maquettes pédagogiques

### 1 ) présentation

Ces 2 cartes permettent de resituer l'utilisation du bus CAN au sein d'un véhicule (même si cela reste très simplifié par rapport à son implantation dans un véhicule). De façon très simplifiée un programme sur PC (équivalent au calculateur de bord sur le véhicule) permet d'agir sur :

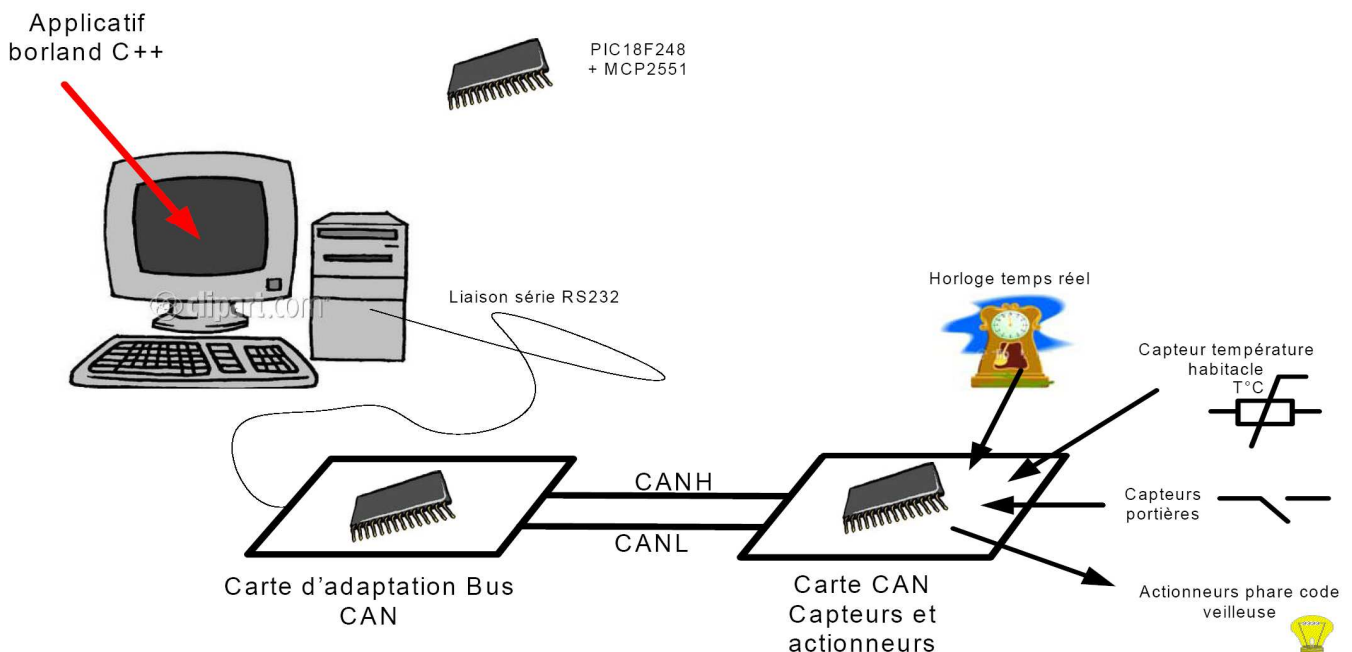
- Les phares
- Les veilleuses
- Les codes
- L'horloge du véhicule

Et de recevoir comme informations :

- L'horloge temps réel du véhicule
- L'état des portières
- La température habitacle voiture

Tous ces actionneurs et capteurs sont situés sur une carte appelée cible. Une carte hôte permet de faire l'adaptation du PC au bus CAN

### 2) Schéma fonctionnel des 2 cartes



### 3) L'applicatif PC

Ce programme développé sous BORLAND C++ permet d'agir grâce à des boutons sur les actionneurs et d'afficher les informations issues des différents capteurs.

On peut agir au niveau des actionneurs :

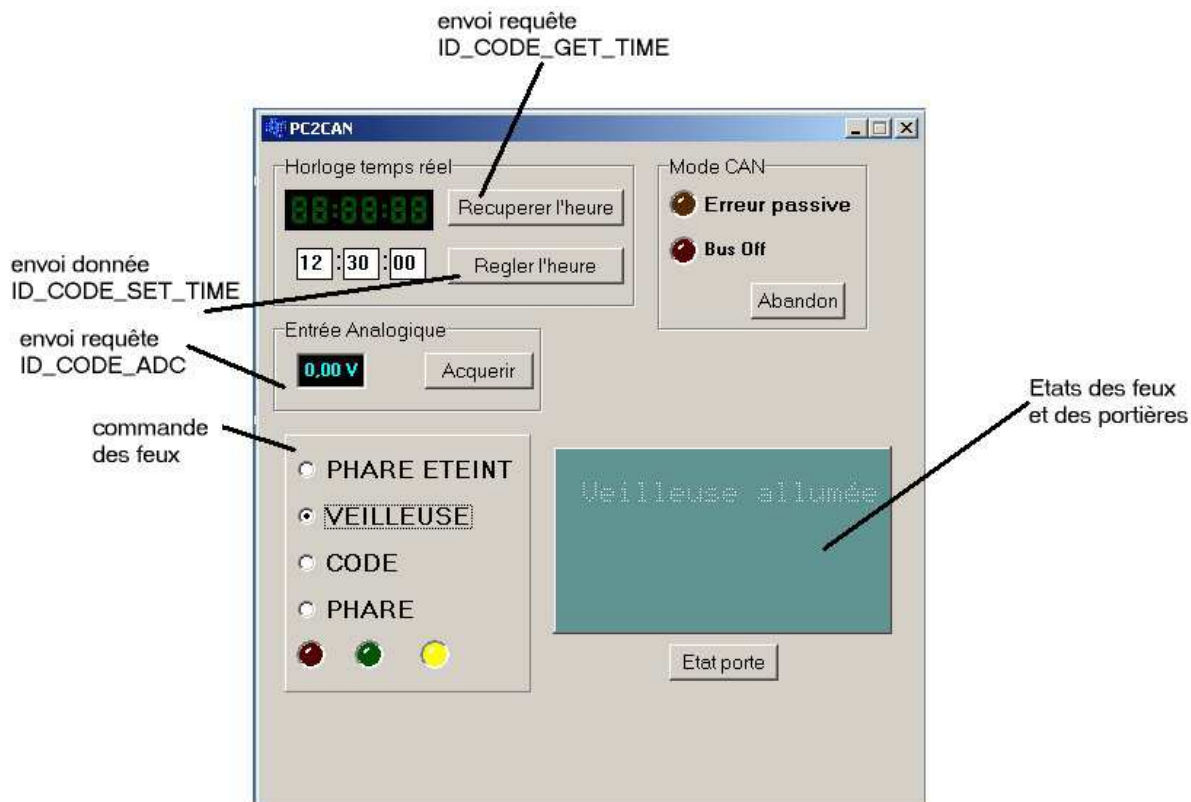
- Sur les feux du véhicule grâce aux 4 onglets à cocher. l'afficheur LCD et les 3 LEDs nous indiquent alors quels sont les feux qui sont allumés.

- On peut programmer l'heure de l'habitacle, en remplissant les différents champs et en cliquant sur le bouton régler l'heure.

On peut recueillir les informations suivantes

- Horloge de l'habitacle en cliquant sur le bouton récupérer l'heure
- Etat des portières en cliquant sur le bouton état porte (l'afficheur LCD nous indique les portières ouvertes).

La partie Mode CAN nous indique l'état du BUS CAN erreur passive ou bus off



## 2) Carte hôte

Elle est constituée d'un PIC18F258 (microcontrôleur intégrant un module CAN) , d'un driver CAN MCP2551 et d'un circuit MAX233 ou MAX232 de conversion de niveaux électriques servant à l'interface RS232.

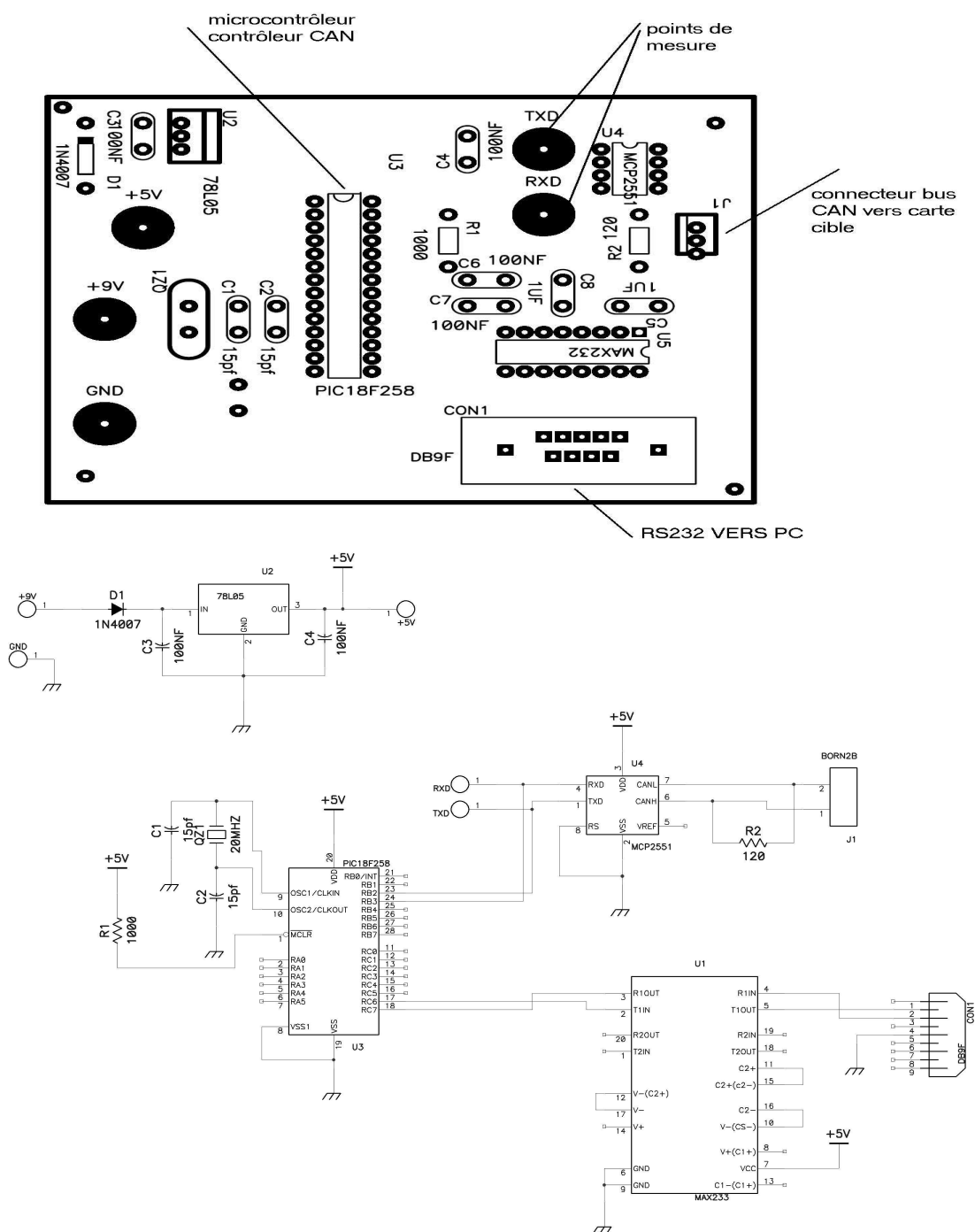
La carte hôte reçoit du PC des demandes d'émission de donnée ou de requête et transmet ces demandes vers la carte cible via le bus CAN avec l'identifiant approprié. Les demandes sont aux nombres de 9 :

Envoyé par le PC vers la carte Hôte

- CODE\_GET\_TIME : requête demande de l'heure
- CODE\_PHARE : donnée phare allumé
- CODE\_VEILLEUSE : donnée veilleuse allumé
- CODE\_CODE : donnée code allumé

- CODE\_PORTE : requête demande état des portières
- CODE\_ETEINT : donnée tous feux éteints
- CODE\_SET\_TIME : donnée programmation de l'heure
- CODE\_ADC : requête demande de la valeur de tension
- CODE\_ABORT : demande d'abandon à la carte hôte des transmissions sur le bus CAN.

## 2.1) Schéma structurel + implantation



CARTE CAN HOTE

24/05/2005  
hote.sch

PETITPA

## 2.2) Les identifiants

Les identifiants envoyés sur le bus CAN sont les suivants :

Type de trame	Identifiant	Requête ou donnée	Longueur données octets
Acquisition heure ID_CAN_GET_TIME	0X0A	requête	3
Prépositionnement heure ID_CAN_SET_TIME	0x0B	donnée	3
Acquisition CAN ID_CAN_ADC	0x0C	requête	1
Commande veilleuse ID_CODE_VEILLEUSE	0x0D	donnée	0
Commande code ID_CODE_CODE	0x0E	donnée	0
Commande phare ID_CODE_PHARE	0x0F	donnée	0
Etat porte ID_CODE_PORTE	0x10	requête	1
Commande feux éteint ID_CODE_ETEINT	0x11	donnée	0

## 2.3) principe

Le programme du PIC est construit autour d'une boucle principale d'attente active de réception de données sur le port série USART .

Les données reçues représentant les directives envoyées par l'application tournant sur le PC à travers son port de communication série RS232.

Les données reçues sur le bus CAN sont gérées par interruption.

La carte hôte reçoit de la cible via le bus CAN les réponses aux requêtes suivantes

**CODE\_GET\_TIME** : requête demande de l'heure

Le code CODE\_GET\_TIME est suivi de 3 octets représentant les heures, minutes et secondes.

**CODE\_PORTE** : requête demande état des portières

Le code CODE\_PORTE est suivi de 1 octet représentant l'état des portières.

**CODE\_ADC** : requête demande de la valeur de tension

Le code CODE\_ADC est suivi de 1 octet représentant la valeur température habitacle.



**CODE\_ABORT** : demande d'abandon à la carte hôte des transmissions sur le bus CAN.

Elle transmet l'information issue de ces réponses vers le PC ainsi que l'état du BUS suivant :

- **CODE\_CAN\_PASV\_ERROR** : indique que la carte cible a son module CAN dans l'état d'erreur passive.
- **CODE\_CAN\_ACTV\_ERROR** : indique l'état d'erreur active.
- **CODE\_CAN\_BUSOFF** : indique que le nœud de la carte hôte est déconnecté *bus off*.

### 3) Carte cible

Comme la carte hôte, elle est dotée d'un microcontrôleur PIC18F258 (microcontrôleur intégrant un module CAN) et d'un driver MCP2551.

A cela s'ajoute :

- l'horloge temps réel DS1337 et son oscillateur DS32kHz,
- 3 LEDS pour simuler l'état des codes, veilleuses, phares.

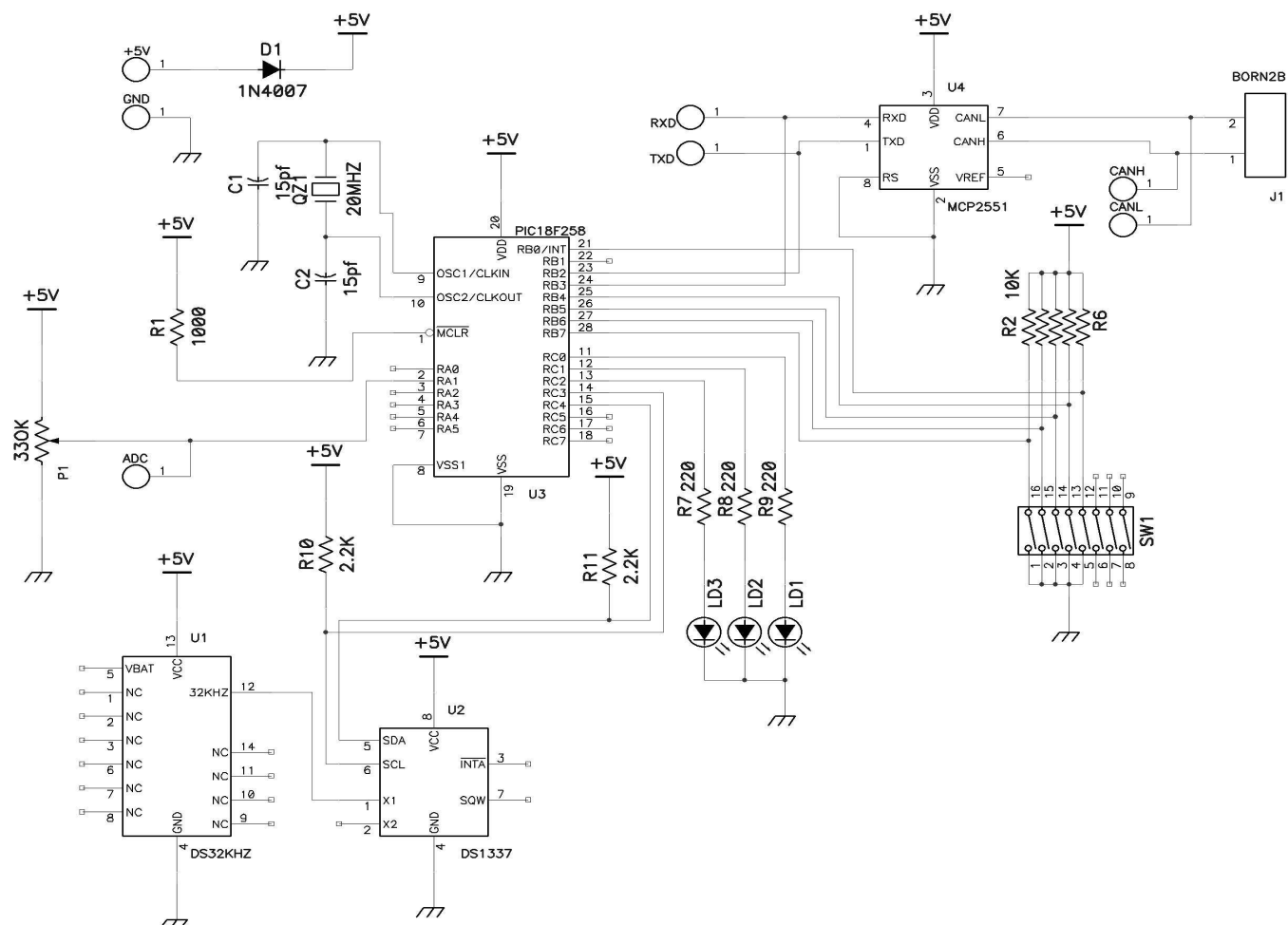
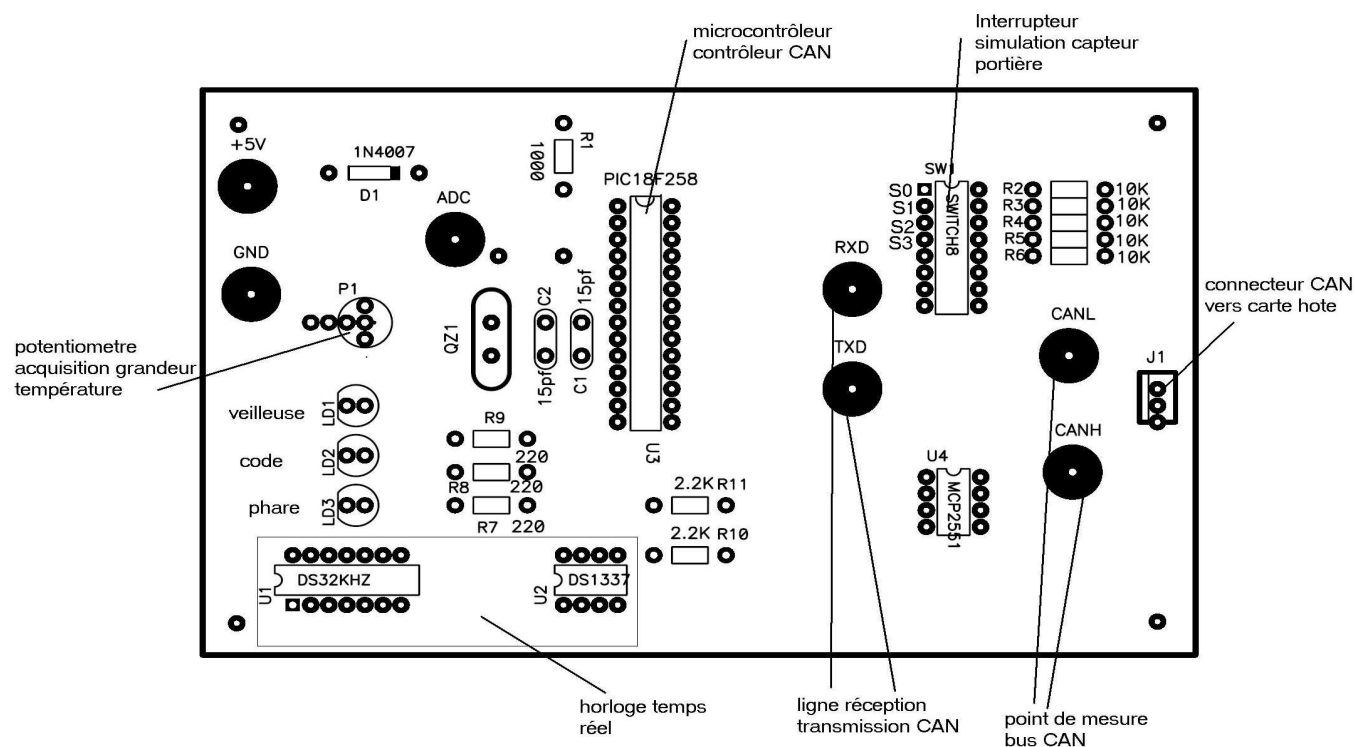
Nom des LEDs	FEUX
LD1	veilleuse
LD2	code
LD3	phare

- 4 interrupteurs S0 à S3 pour simuler l'état des portières

Interrupteur	Etat des portières
ON	ouverte
OFF	fermé

Nom des interrupteurs	Type de portières
S0	Porte arrière gauche
S1	Porte arrière droite
S2	Porte avant gauche
S3	Porte avant droite

### 3.1) schéma structurel et implantation



### **3.2) Principe**

A la réception d'une trame de requête avec l'identifiant ID\_CAN\_GET\_TIME, le microcontrôleur interroge l'horloge temps réel et envoie une trame de données contenant 3 octets correspondant aux heures, minutes et secondes.

A la réception d'une trame de requête avec l'identifiant ID\_CODE\_PORTE, le microcontrôleur envoie une trame de données contenant 1 octet correspondant à l'état des portières.

A la réception d'une trame de données avec l'identifiant ID\_CODE\_PHARE, ID\_CODE\_CODE, ID\_CODE\_VEILLEUSE, ID\_CODE\_ETEINT le microcontrôleur agit sur les feux correspondants .

Si l'identifiant correspondant à la valeur de la tension appliquée sur l'entrée AN1 est reçu, on lance la conversion analogique-numérique puis on envoie une trame de données contenant la valeur de la tension échantillonnée sous la forme d'un octet (0 représentant 0V, 255 la valeur de VDD).

La remise à l'heure de l'horloge temps réel s'effectue après la réception d'une trame de données d'identifiant ID\_CAN\_SET\_TIME contenant les 3 octets représentant les heures, minutes et secondes.

**Annexe 2 : extrait partiel du programme en C de la carte cible**

Dans la fonction main(), la gestion des requêtes est la suivante:

```
switch(id)
```

```
{
    // Demande de l'heure
    case ID_CAN_GET_TIME:
        MessageData[0]=RTC_get_hour();
        MessageData[1]=RTC_get_min();
        MessageData[2]=RTC_get_sec();
        len=3;
        break;

    // Valeur de AN1
    case ID_CAN_ADC:
        MessageData[0]=get_adc_conv();
        len=1;
        break;

    default:
        return;
}
```

// La fonction RTC\_get\_reg permet de récupérer l'information dans un des registres du DS1307

// L'adresse du registre est le paramètre d'entrée de la fonction

// La valeur du registre est le paramètre de sortie de la fonction

```
char RTC_get_hour(void)
```

```
{
    char tmp;
    if ((tmp = RTC_get_reg(RTC_HOUREG)) == -1)
        return -1;
    return rtc_reg2bin(tmp);
}
```

```
static unsigned char rtc_reg2bin(unsigned char x)
```

```
{
    return (x & 0x0F) + ((x & 0xF0)>>4)*10;
}
```