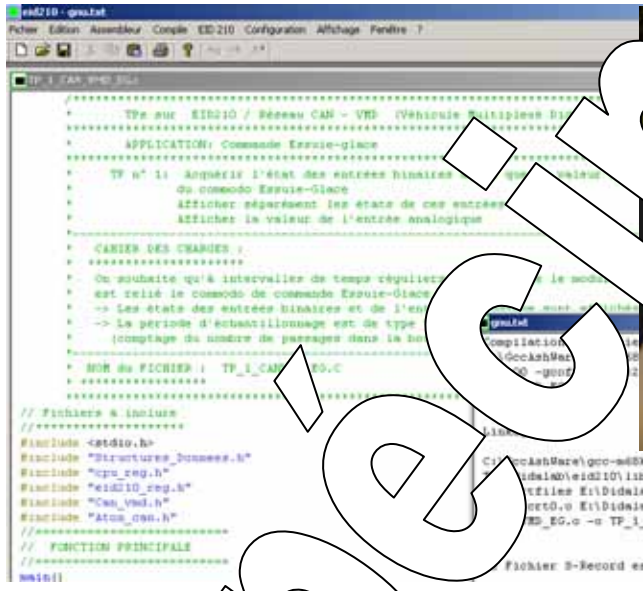
MANUEL
de
TRAVAUX PRATIQUES

Système CAN - V.M.D.

Véhicule **M**ultiplexé **D**idactique

Application: **Commande Essuie-Glace**



Véhicule Multiplexé Didactique (VMD)
Application "Commande Essuie-Glace"
Référence système: CAN 01B
+
Carte processeur Réf: EID210 000

- E... de développement intégré
- Éditeu... eur, linkeur, loadeur) Réf: EID210
- ...pilate... Réf: EID210 100

...s techniques associées
- ...a carte processeur EID210 000 seule Réf: EID210 010
- ...r le système CAN 01B Réf: EID057 010
- Sur "CAN Expander" MCP25050

Manuels de TP associés
-Sur la carte processeur EID210000 seule Réef: EID210040
-Sur la carte simulateur E/S EID210000 seule Réef: EID211040

didalab
ENSEIGNEMENT SUPERIEUR

Z.A. La Clef St Pierre - 5, rue du Groupe Manoukian 78990 ELANCOURT France
Tél. : 33 (0)1 30 66 08 88 - Télécopieur : 33 (0)1 30 66 72 20
e-mail : ge@didalab.fr Web : www.didalab.fr

Version du : 20/07/2010 Référence document : EID 057 040

SOMMAIRE

1 TP N°1 : ACQUERIR L'ETAT DU COMMODO ESSUIE GLACE

1.1 Sujet:

Objectifs :	- Définir les trames de commande qui permettent d'initialiser un nœud CAN - Définir, puis envoyer une trame interrogative à un module d'entrées, accessible à une adresse définie. - Tester si une trame a été reçue. - Extraire d'une trame réponse les informations attendues. - Visualiser sur l'écran les trames reçues ainsi que les trames envoyées. - Visualiser sur l'écran les données attendues.
Cahier des charges :	A intervalles de temps réguliers, on interroge le module sur lequel est connecté le commodo "Essuie-Glace" afin de connaître son état. → L'entrée analogique sera configurée. → Les trames reçues ou envoyées sur le bus CAN sont affichées. → La temporisation est de type "logiciel" (comptage du nombre de passages dans la boucle principale) → Les différentes commandes générées par la position de la manette commodo seront affichées individuellement.

1.2 Eléments de solution

1.2.1 Analyse

Avant de pouvoir lire l'état des entrées du module 8 entrées (dont on donne le schéma structurel en Annexe) sur lequel est relié le commodo essuie-glace, il faut préalablement configurer le "CAN expender MCP25050" Cette configuration a pour but de définir en entrée les 8 bits du port (GP0 à GP7). Pour confirer un bit du port en entrée, il faut écrire un "1" sur le bit correspondant du registre GPDDR (Data Direction Register) (voir tableau "Register 5-1" à la page 27 de la notice du 25050)
Dans ce cas, la trame envoyée par le contrôleur CAN (Circuit SJA1000 sur carte CAN_PC104) sera vue par le récepteur (circuit MCP25050 sur module) comme une IM "Input message", avec la fonction "Write register" (voir documentation technique du MPC25025 pages 22). On pourra ainsi modifier les différents registres du module "Asservissement".

☞ Configuration des liaisons en entrées

Définition de la trame de commande (IM) qui sera envoyée pour [illegible] module "Commodo-EG"

→ Définition de variables structurées sous le modèle "**Trame**"

```
Trame T_IM_Commodo_EG;
```

→ Définition des éléments d'identification de la variable structurée "**T_IM_Commodo_EG**"

```
T_IM_Commodo_EG.trame_info.registre=0x00;  //On [illegible] tous les bits à 0
T_IM_Commodo_EG.trame_info.champ.extend=1; //On [illegible] mode étendu
T_IM_Commodo_EG.trame_info.champ.dlc=0x07  [illegible] octets de données
T_IM_Commodo_EG.ident.extend.identificat   =Ident_T_IM_Commodo_EG;
```

Rem: **Ident_T_IM_Commodo_EG** est défini dans le [illegible] CAN_PC104.h

→ Définition des paramétres associée à la trame de commande

Il faut initialiser le registre GPDDR ("Data Direction Register") en écrivant un 1 si bit d'entrée et un 0 si bit de sortie(doc MCP25050 p27).

```
T_IM_Commodo_EG.data[0]=0x1F; // Adresse du registre GPDDR en écriture
                                                    (doc MCP25050 page 16)
T_IM_Commodo_EG.data[1]=0x7F; // [illegible] bit non concerné (doc MCP25050 page 16)
T_IM_Commodo_EG.data[2]=0x7[illegible] // Valeur à charger dans le registre adressé
                                       (tous les bits du port sont des entrées)
```

Suite à ces définitions, il faudra

- envoyer la trame par la fonction **Ecire_Trame(T_IM_Commodo_EG)**
- puis attendre la réponse de type "Ack" en utilisant la fonction **Lire_Trame(&T_Recue)**

L'entrée "GP0" étant une entrée analogique (AN0), il faudra mettre en œuvre la fonctio de conversion anaolique -> numérique

Activation de la conversion Analogique-> Numérique

Définition des trames de données associées pour:
→ activer et configurer la conversion Analogique vers Numérique
D'après la notice technique du circuit MCP25050 (pages 34 à37) :
Il faut initialiser le registre ADCON0,

```
IM_Commodo_EG.data[1]=0x2A; // Adresse du registre ADCON0 en écriture
(doc MCP25050 p15), décalage = 0E_H + 1C_H = 2A_H
T_IM_Commodo_EG.data[2]=0xF0; // Masque: Seul le bit 7 est concerné
T_IM_Commodo_EG.data[3]=0x80; // Valeur: ADON=1 -> Activation convertisseur
                                et "prescaler rate" = 1:32
```

Suite à ces définitions, il faudra

- envoyer la trame par la fonction **Ecire_Trame(T_IM_Commodo_EG)**
- puis attendre la réponse de type "Ack" en utilisant la fonction **Lire_Trame(&T_Recue)**

Il faut aussi initialiser le registre ADCON1:

```
T_IM_ Commodo_EG.data[0]=0x2B; // Adresse du registre ADCON1 en écriture
 (doc MCP25050 p15) 0F_H + décalage = 0E_H + 1C_H = 2B_H
T_IM_ Commodo_EG.data[1]=0xFF; // Masque:  les 8 bits sont concernés
T_IM_ Commodo_EG.data[2]=0x03; // Valeur: (doc MCP25050 p36)
     b7=ADCS1=0; b6=ADCS0=0 → Fréquence Fosc/2
     b5=VCFG1=0; b4=VCFG0=0 → Plage de tension d'entrée 0/+5V
     PCFG3:PCFG0=1100       → Convertion des entrées analogiques 1 et 0 ( sur GP1 et GP0)
```

Suite à ces définitions, il faudra

- envoyer la trame par la fonction **Ecire_Trame(T_IM_Commodo_EG)**
- puis attendre la réponse de type "Ack" en utilisant la fonction **Lire_Trame(&T_Recue)**

Acquisition de l'état des entrées sur le module "Commodo-EG"

A intervalles de temps réguliers, on interroge le module 8 entrées sur lequel est connecté le commodo Essuie-Glace. Dans la réponse on attend également le résultat de conversion [illegible] analogique.

Définition de la trame interrogative qui sera envoyée

Dans ce cas, la trame envoyée par le contrôleur CAN (Circuit [illegible]1000 sur carte CAN_PC104) sera vue par le récepteur (circuit MCP25050 sur module) comme un "Info[illegible]est Message", avec la fonction "Read A/D Regs" (voir documentation technique du "CAN Expa[illegible]" MCP25[illegible]5 pages 22).

→ Définition de variables structurées sous le modèle "**Trame**"

```
Trame T_IRM_Commodo_EG;
```

// Trame destinée à l'interrogation du module 8 entrées sur [illegible]necté le commodo lumière

Rem: La variable structurée **T_IRM_Commodo_EG** [illegible]rtera [illegible]s utiles seulement, 1octet pour trame_info et 4 octets pour l'identificateur en mode étendu [illegible]endra [illegible]dresse du registre concerné par la lecture.

→ Accès et définition des différents [illegible]émen[illegible] variable structurée "T_IRM_Commodo_EG"

```
T_IRM_Commodo_EG.trame_inf[illegible]gistre[illegible]  //[illegible]initialise tous les bits à 0
T_IRM_Commodo_EG.trame_info.cha[illegible]exte[illegible]  //On travaille en mode étendu
T_IRM_Commodo_EG.trame_info.cha[illegible]=0x0[illegible]  //Type trame ->  Interrogative
T_IRM_Commodo_EG.trame_info.champ[illegible]x08;  //Il y aura 8 octets de données
T_IRM_Commodo_EG.ident.exte[illegible]enti[illegible]ur.ident= Ident_T_IRM8_Commodo_EG;
  //! c'est sur 29 bits [illegible]our défi[illegible]l'adresse du commodo
```

Des labels définissant les différe[illegible]identific[illegible]srs ont été déclarés dans le fichier CAN_VMD.h

Suite à ces définition[illegible] il faudra

- envoyer la [illegible]ame par la fo[illegible]**Trame(T_IRM_Commodo_EG)**
- puis attendre l[illegible]type [illegible]M" en utilisant la fonction **Lire_Trame(&T_Recue)**

Trame reçue en répons[illegible]'i[illegible]rro[illegible]ion

D'après la définition des [illegible]ate[illegible]donnée en Annexe 1, une trame de réponse à une IRM a le même identificateur qu[illegible] int[illegible]é qui en a été à l'origine.

Vu du module [illegible](CA[illegible]0), la réponse à un IRM (Information Request Message) est un OM (Output Message).

La [illegible]éponse, [illegible]u comportera en données associées 8 octets (doc MCP25050 p22):

- octet de rang [illegible]a[0])→ valeur IOINTFL non utilile dans notre cas
- [illegible]g 1 [illegible]1)→ valeur GPIO → Valeur des entrées logiques
- [illegible]t de [illegible]2 (da[illegible])→ valeur AN0H → 8 bits MSB conversion entrée anologique 0
- octet de [illegible]3 (data[3])→ valeur AN1H → 8 bits MSB conversion entrée anologique 1
- oc[illegible]g 4 (data[4])→ valeur AN10L → 2 fois 2 bits LSB conversion entrées ana. 1 et 0

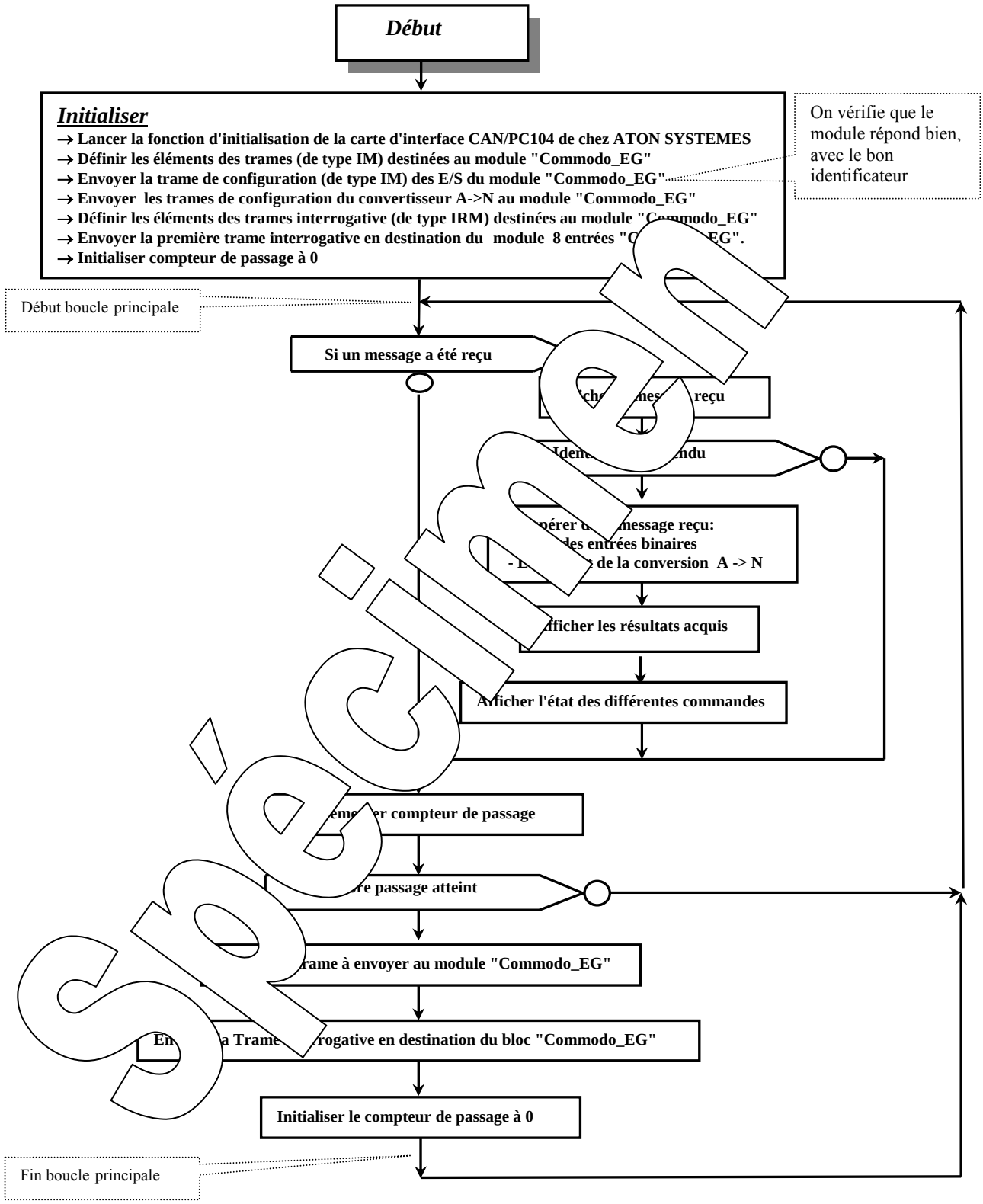
Les 3 autres oc[illegible]ne sont pas utiles dans notre application.

Le résultat de conversion est sur 10bits:

- pour résultat AN0

d9 d8 d7 d6 d5 d4 d3 d2	d1 d0 - - - - - -
AN0H ->data[2]	**AN10L -> data[4]**

1.2.2 Organigramme



1.2.3 Programme en "C"

```
/*****************************************************************************************************
*      TPs sur  EID210 / Réseau CAN - VMD  (Véhicule Multiplexé Didactique)
*****************************************************************************************************
*      APPLICATION: Commande Essuie-glace à distance
*****************************************************************************************************
*            TP n°1:  Acquérir l'état des entrées binaires ainsi que la valeur de l'entrée analogique
*           du commodo Essuie-Glace
*           Afficher séparément les états de ces entrées binaires
*                        Afficher la valeur de l'entrée analogique
*-----------------------------------------------------------------------------------------
*  CAHIER DES CHARGES :
*  ********************
*  On souhaite qu'à intervalles de temps réguliers on interroge le module 8 entrées sur lequel
*  est relié le commodo de commande Essuie-Glace
*  -> Les trames reçues et envoyées sur le bus CAN sont affichées
*  -> Les états des entrées binaires et de l'entrée analogique sont affichés
*  -> La temporisation est de type "logiciel"
*           (comptage du nombre de passages dans la boucle principale)
*-----------------------------------------------------------------------------------------
*  NOM du FICHIER :  TP_1_CAN_VMD_EG.C

* *****************
*****************************************************************************************************/
// Fichiers à inclure
//*********************
#include <stdio.h>
#include "Structures_Donnees.h"
#include "cpu_reg.h"
#include "eid210_reg.h"
#include "Can_vmd.h"
#include "Aton_can.h"
//============================
//  FONCTION PRINCIPALE
//============================
main()
{// Définition de variables locales
int Compteur_Passage;
unsigned short S_Temp,Valeur_ANA;
unsigned int Cptr_TimeOut;
unsigned char AN0H,AN10L;  // Pour récupé          at de la co          /A->N
Trame Trame_Recue;
Trame T_IRM_Commodo_EG;  // Trame po          rroger M          E sur Commodo Essuie Glace
                                                  // IRM          formation Request Message -> Trame interrogative
Trame T_IM_Commodo_E          // Trame pour          E sur Commodo Essuie Glace
                                                  formation Message -> Trame de commande

// Initialisations
//*****************
clsscr();
/* Initialisation DU SJA1000 de la          04 */
Init_Aton_CAN();
// Pour initialiser les lia          s du          mmodo Essuie-Glace"
T_IM_Commodo_EG.          =0x00;
T_IM_Commodo_          e_info.          xtend=1; // On travaille en mode étendu
T_IM_Commodo_          info.ch          =0x03; // Il y aura 3 données de 8 bits (3 octets envoyés)
T_          do_EG.          ateur.ident=Ident_T_IM_Commodo_EG;
T          odo_EG.data[          // première donnée -> "Adresse" du registre concernée
                                                  //(GPDDR donne la direction des I/O) adresse = 1Fh  page 16
          =0x7          ème donnée -> "Masque"
                                                  //-> Tous les bits concernés sauf GP0 (voir doc page 16)
T_IM_          do_EG.da          0x7F; // tro          ème donnée -> "Valeur" -> Tous les bits en entrée
// Envoi trame po          défi          direction des entrées sorties
Ecrire_Trame(T          do_EG);  // Envoyer trame sur réseau CAN
          Cptr          0;
          do{Cptr_TimeOut++;}while((Lire_Trame(&Trame_Recue)==0)&&(Cptr_TimeOut<2000));
          if(Cptr_TimeOut==2000)
                    { gotoxy(2,12);
                     printf(" Pas de reponse a la trame de commande en initialisation \n");
                     printf(" Couper et/ou  remettre    alimentation 12 V  \n");
                     do{}while(Lire_Trame(&Trame_Recue)==0); // On attend les trames  "On Bus"
        for(Cptr_TimeOut=0;Cptr_TimeOut<100000;Cptr_TimeOut++);
        Ecrire_Trame(T_IM_Commodo_EG);  // Renvoyer trame IM sur réseau CAN
                    Cptr_TimeOut=0;
                    do{Cptr_TimeOut++;}while((Lire_Trame(&Trame_Recue)==0)&&(Cptr_TimeOut<2000));
                    if(Cptr_TimeOut==2000)
                              { gotoxy(2,12);
```

```
                                printf(" Pas de reponse a la trame de commande en initialisation \n");
                                printf("  Modifier le programme et recommencer  \n");
                        do{}while(1); // On reste bloqué, on ne continue pas
                                }
                        }
clsscr();
// Pour afficher le titre du TP
gotoxy(1,4);
printf("  ***************************************************************************\n");
printf("   TPs sur Reseau CAN   Application: Commande Essui-Glace a distance  \n");
printf("  ---------------------------------------------------------------------\n");
printf("   TP  n°1                                                  \n");
printf("    ACQUERIR les entrees binaires du COMMODO EG                  \n");
printf("    ACQUERIR la valeur de l'entree analogique du COMMODO EG        \n");
printf("    AFFICHER les valeurs acquises                                      \n");
printf("    ********************************************************************* \n");
// Pour activer les conversions Ana -> Num
T_IM_Commodo_EG.data[0]=0x2A;         // Adresse du registre ADCON0
T_IM_Commodo_EG.data[1]=0xF0;         // Masque -> bits 7..4 concernés
T_IM_Commodo_EG.data[2]=0x80;         // Valeur ->  ADON=1 et "prescaler rate"=1:32
Ecrire_Trame(T_IM_Commodo_EG);
do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre réponse
// Pour définir le mode de conversion
T_IM_Commodo_EG.data[0]=0x2B;         // Adresse du registre ADCON1
T_IM_Commodo_EG.data[1]=0xFF;         // Masque -> tous les bits sont concer
T_IM_Commodo_EG.data[2]=0x0C;         // Valeur ->  voir doc MCP25050 pa
Ecrire_Trame(T_IM_Commodo_EG);
do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre réponse
// Pour trame interrogative envoyée au commodo EG -> 'IRM'   (Informati  Reques
// Définir données d'identification
T_IRM_Commodo_EG.trame_info.registre=0x00;
T_IRM_Commodo_EG.trame_info.champ.extend=1;
T_IRM_Commodo_EG.trame_info.champ.dlc=0x08;  // On dem                r de 8 
T_IRM_Commodo_EG.trame_info.champ.rtr=1;
T_IRM_Commodo_EG.ident.extend.identificateur_ident=Ide          M8_C          EG;
                                                                //          nitions dans fichier CAN_VMD.h
Ecrire_Trame(T_IRM_Commodo_EG); // On en      ane      emière
// Initialiser les variables diverses
Compteur_Passage=0;
// Boucle principale
//*******************
  do
          {  // On teste si une trame a été re
            if (1==Lire_Trame(&Trame_Re      // La fonc      nvoie 1 dans ce cas
                    {goto      4,18);
                    print      rame recue en re      : c'est une 'OM' (Output Messsage) \n");
                    Affich     Tra      Recue
                    if (Trame_      d.identificateur.ident==Ident_T_IRM8_Commodo_EG)
                                    n a r      etat      commodo donc on affiche les nouveaux états
                                    G.v      =~Trame_Recue.data[1];  // Les entrées binaires
                                    ame_      .data[2];  // On récupère les MSB de la conversion A->N  voie AN0
                                    ue.data[4]; // On récupère les LSB AN1 et AN0
                                    s et reconstituer le résultat de conversion sur 10 bits
                                    ANA=(unsigned short)(AN0H);     //Transfert avec transtypage
                                    NA=Valeur_ANA<<2;               // Décaler de 2 bits vers poids forts
                                    unsigned short)(AN10L&0x0C); // Pour ne récupérer que les 2 bits AD1 et AD0
                                    A=Valeur_ANA|(S_Temp>>2);  // On reconstitue les 10 du resultat de conversion
                                    iche les résultats
                                    4,22);
                                    des differentes entrees imposees par le commodo:\n");
                                    t entree binaires (en Hexa) =%2.2x\n",Commodo_EG.valeur);
                                    printf("  aleur de l'entee analogique =%4d\n",Valeur_ANA);
                                    }

            Compt      age++;
            if (Compteur_Passage==50000)
                {Compteur_Passage=0;// C'est la fin de temporisation
                        gotoxy(4,14);
                        printf("Trame de demande etat commodo: c'est une  'IRM'   Input Request Mesage\n");
                        Affiche_Trame(T_IRM_Commodo_EG);  // La trame interrogative est affichée à l'écran
                        Ecrire_Trame(T_IRM_Commodo_EG);
                }
            }while(1);    // FIN de la boucle principale
}
// FIN fonction principale
```

2 TP N°2: MARCHE/ARRET DU MOTEUR ESSUIE GLACE

2.1 Sujet:

Objectifs :	- Définir les différentes trames de commande à faire transiter par le réseau CAN en fonction d'une action souhaitée. - Commander un actionneur électrique (moteur à courant continu) en variation de vitesse, par l'intermédaire d'un pré-actionneur commandable par réseau CAN.
Cahier des charges :	Le bouton poussoir BP1, connecté sur l'entrée GP4 du "CAN Expander" sur module 8 entrées sur lequel peut être [illegible] le commodo Essuie-Glace, est utilisé pour commander le moteur: BP1 appuyé (GP4 = 0) → le [illegible] est en [illegible]be BP1 n'est pas appuyé (GP4 = 1) → [illegible]oteur est à l'arrêt La vitesse du moteur est déf[illegible] d'ent[illegible]logique n°0 du module "Commodo Essuie Glace"

2.2 Elements de solution

2.2.1 Analyse

Principe:

Le module "Asservissement" (dont on donne le schéma structurel en Annexe) permet la commande d'un moteur à courant continu 12V/ 1A, dans les deux sens de rotation, en mode "PWM" (modulation de largeur d'impulsions).

Un circuit de puissance (réf: L6202"), pilotable par signaux logiques est implanté sur le module.

D'après le schéma électronique du module "Asservissement" donné en Annexe, ce pilotage se fait par les sorties GP2, GP3 et GP4 de l'interface CAN (circuit MCP25050):

→ GP2 ou "PWM1" sortie logique à rapport cyclique variable, qui permet de faire varier la vitesse

→ GP3 ou "PWM2" sortie logique à rapport cyclique variable qui permet de faire varier la vitesse du moteur dans le sens négatif,

→ GP4 ou "ValidIP" sortie logique qui, lorqu'elle est positionnée à 1, valide le circuit d' Interface de Puissance "L6202".

Dans notre application, on fera tourner le moteur en positif. Seule la sortie PWM1 sera activée. Par contre la sortie GP4 ou "ValidIP" devra être positionnée à 1.

Définitions des trames pour commander le moteur

Une sucession de trames de commande (type IM) devra être envoyée pour donner un ordre bien précis au module asservissement, afin de l'initialiser dans le but d'obtenir la fonction désirée

Définition des éléments d'identification d'une trame de type IM permettant de commander le module "Asservissement":

Dans ce cas, la trame envoyée par le contrôleur CAN (circuit SJA1000 sur carte CAN_PC104) sera vue par le récepteur (circuit MCP25050 sur module) comme une IM "write message", avec la fonction "Write register" (voir documentation technique du MCP2025 page 30). On pourra ainsi modifier les différents registres du module "Asservissement".

L'identificateur défini dans le chapitre 1, pour "IM" envoyé à la carte "Asservissement"est :

0x008800[illegible] (label: **Ident_T_IM_Asservissement**)

→ Définition de variables structurées sous le modèle "**Trame**":

```
Trame T_IM_Asservissement;
```

→ Définition des éléments d'identification de la variable structurée "**T_IM_Asservissement**"

```
T_IM_Asservissement.[illegible]=0x00;  //On initialise tous les bits à 0
T_IM_Asservissement.[illegible]champ.extend=1; //On travaille en mode étendu
T_IM_Asservissement.[illegible]champ.dlc=0x03; //Il y aura 3 octets de données
T_IM_Asservissement.[illegible].identificateur.ident=Ident_T_IM_Asservissement;
```

A chacune des trames de commande "IM" il y aura donc à définir les trois octets associés.

Trame n°1 → *pour définir les entrées et sorties du module "Asservissement"*

Il faut initialiser le registre DDR ("Data Direction Register") en écrivant un 1 si bit d'entrée et un 0 si bit de sortie du MCP2505[illegible] après schéma structurel de la carte "Asservissement" donné en Annexe:

GP7=fcg -> entrée; GP6=fcg -> entrée; GP5=fcd -> entrée; GP4=ValidIP -> sortie;

GP3=PWM2 -> sortie; GP2=PWM2 -> sortie; GP1=AN1 -> entrée; GP0=AN0 -> entrée;

```
T_IM_Asservissement.data[0]=0x1F; // Adresse du registre GPDDR en écriture
                                                    (doc MCP25050 page 16)
T_IM_Asservissement.data[1]=0x7F; // Masque: bit 7 non concerné (doc MCP25050 page 16)
T_IM_Asservissement.data[2]=0xE3; // Valeur: à charger dans le registre adressé
```

Suite à ces définitions, il faudra

- envoyer la trame par la fonction **Ecire_Trame(T_IM_Asservissement)**
- puis attendre la réponse de type "Ack" en utilisant la fonction **Lire_Trame(&T_Recue)**

Trame n°2 → *pour initialiser la sortie GP2 en sortie PWM1 (commande du moteur dans le sens positif)*

D'après la notice technique du circuit MCP25050 (pages 30 à 32), la génération du signal PWM1 se fait à partie du "Timer1" et la fréquence de ce signal est choisie grâce au registre "T1CON" d'adresse 05_H (page 15 Doc MCP25050).

bit 7 =1 TMR1ON Validation du "Timer 1"
bits 5:4 seront mis à 0 pour avoir un coefficient de division de fréquence égal à 1 ("TMR1 prescaler value" = 1)

```
T_IM_Asservissement.data[0]=0x21; // Adresse du registre T1CON en écriture
 (doc MCP25050 p15) 05H + décalage = 05H + 1CH = 21H
T_IM_Asservissement.data[1]=0xB3; // Masque sur le registre (doc MCP25050 p32)
T_IM_Asservissement.data[2]=0x80; // Valeur à charger dans le registre adressé
```

Suite à ces définitions, il faudra

- envoyer la trame par la fonction **Ecire_Trame(T_IM_Asservissement)**
- puis attendre la réponse de type "Ack" en utilisant la fonction **Lire_Trame(&T_Recue)**

Trame n°3 → *pour définir la fréquence de la sortie PWM1:*

Cette fréquence dépend de la valeur chargée dans le registre "PR1"

```
T_IM_Asservissement.data[0]=0x23; // adresse du registre PR1 en écriture
 (doc MCP25050 p15) 07H + décalage = 07H + 1CH = 23H
T_IM_Asservissement.data[1]=0xFF; // Masque sur le regi       MCP25050 p32)
T_IM_Asservissement.data[2]=0xFF; // On chargera 255      le      tre
```

La fréquence du quartz implanté sur la carte "asservissement" étant [illegible] 16M[illegible] fréquence du signal PWM1 sera donc égale à : $F_{PWM} = 16.10^6/(4.256) = 15{,}6$ KH[illegible]
Ce qui est une fréquence correcte pour piloter un moteur en [illegible]quen[illegible]siblement inaudible).
Suite à ces définitions, il faudra

- envoyer la trame par la fonction **Ecire_Trame(T_IM_[illegible]men**
- puis attendre la réponse de type "Ack" en [illegible]isant [illegible]re_Trame(&T_Recue)**

Trame n°4 → *pour définir le rapport cyclique de [illegible]M1 (module de la commande donc de la vitesse du moteur)*

```
T_IM_Asservissement.data[0]=0x25; //a[illegible] registre PWM1DCH en écriture
 (doc MCP25050 p15) 09H + dé[illegible]lage[illegible] + 1C[illegible]
T_IM_Asservissement.data[1]=0xFF; //[illegible] sur [illegible]registre (doc MCP25050 p33)
T_IM_Asservissement.data[2]=0x7[illegible] // C[illegible]le =127 (0xFF=255 pour la commande Maxi)
```

Suite à ces définitions, il faudra

- envoyer la trame par la fonction **E[illegible]ne(T_IM_Asservissement)**
- puis attendre la réponse [illegible]e "Ack [illegible]ilisant la fonction **Lire_Trame(&T_Recue)**

Pour démarrer (ou arrêter) le mo[illegible]on agir[illegible]rs sur le signal 'ValidIP' soit la sortie GP4
→ *pour valider le [illegible]uit de puiss[illegible]r le moteur)*

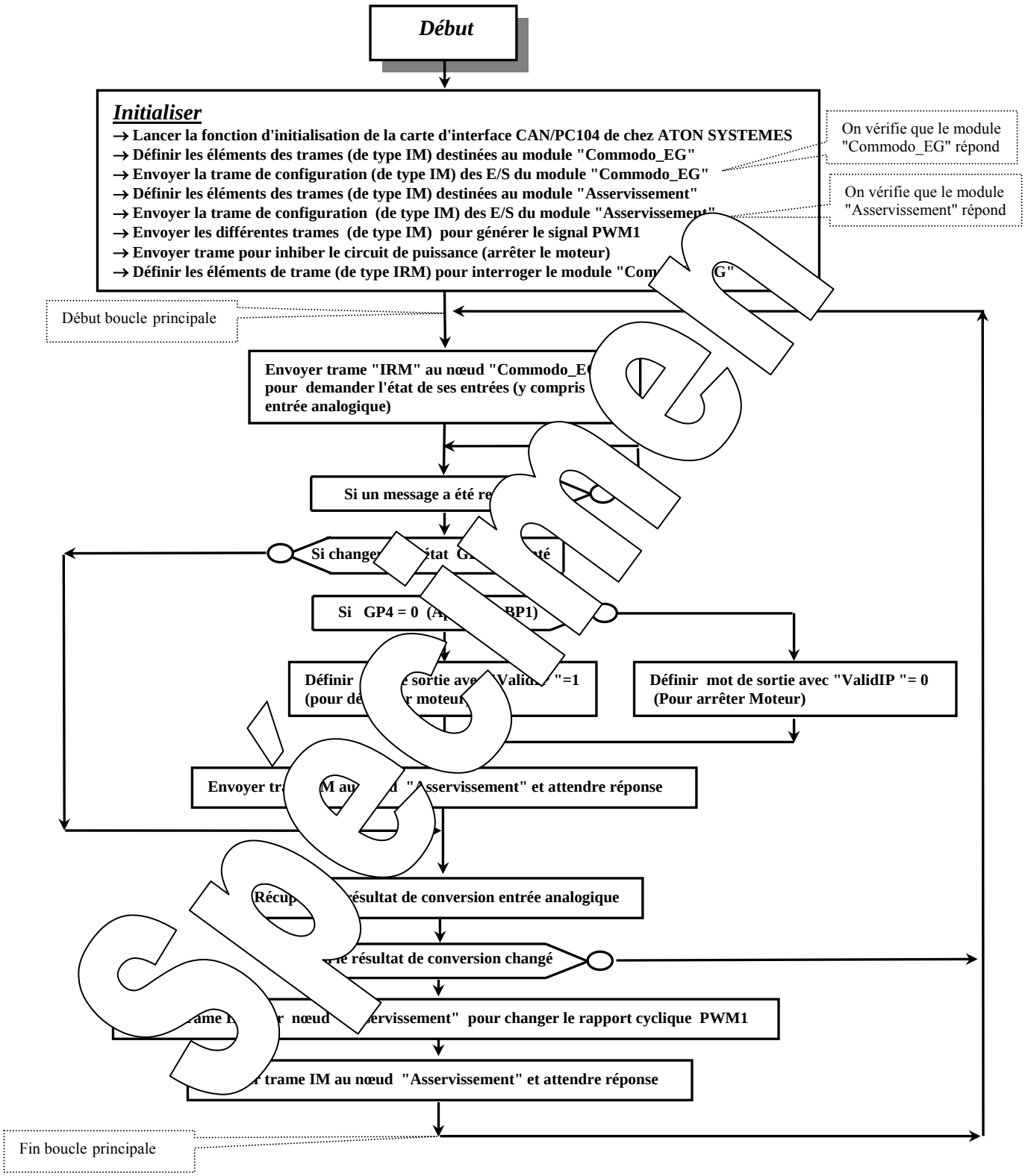
```
T_IM_Asservisse[illegible]nt[illegible]=0[illegible] //adresse du registre GPLAT en écriture
T_IM_Asservisseme[illegible] 0x10; //masque sur le registre (doc MCP25050 p27)
T_IM_Asservisseme[illegible]ata[illegible]=0[illegible]0; //On positionne le bit GP4 du registre à 1
```

→ *pour inhiber le circ[illegible]ssa[illegible](arrêter le moteur)*

```
T_IM_Asservissement.[illegible]01[illegible]E; //adresse du registre GPLAT en écriture
T_IM_Asservi[illegible] da[illegible]x10; //masque sur le registre (doc MCP25050 p27)
T_IM_Asservi[illegible]ta[2]=0x00; //On positionne le bit GP4 du registre à 0
```

Dé[illegible]ns d[illegible]me[illegible]ur configurer et interroger le module commodo E.G. ([illegible]P n°1)

2.2.2 Organigramme



2.2.3 Programme en "C"

```
/*****************************************************************************************************
*      TPs sur  EID210 / Réseau CAN - VMD  (Véhicule Multiplexé Didactique)
*****************************************************************************************************
*      APPLICATION: Commande Essuie-glace à distance
*****************************************************************************************************
*            TP n°2:  Marche Arrêt du moteur + vitesse réglable par commodo Essuie-Glace
*-----------------------------------------------------------------------------------------------
*   CAHIER DES CHARGES :

*  ********************

*   On souhaite que l'etat du moteur (marche ou arret) suive l'etat de BP1 (GP4)
*    La vitesse du moteur sera fonction de l'entrée analogique 0 du commodo EG
*-----------------------------------------------------------------------------------------------
*  NOM du FICHIER :  TP_2_CAN_VMD_EG.C
* ******************
*****************************************************************************************************/
// Fichiers à inclure
//********************
#include <stdio.h>
#include "Structures_Donnees.h"
#include "cpu_reg.h"
#include "eid210_reg.h"
#include "Can_vmd.h"
#include "Aton_can.h"
//============================
//  FONCTION PRINCIPALE
//============================
main()
{// Définition de variables locales
int Compteur_Passage;
char Commodo_EG_Mem;
unsigned int Cptr_TimeOut;
unsigned char AN0H; // Pour mémoriser les 8 biits de poi        résult      nversion A->N
unsigned char Val_Vitesse,Val_Vitesse_Mem;
Trame Trame_Recue;
Trame T_IRM_Commodo_EG;                 //      ur interroger      e 8E sur Commodo Essuie Glace
                                        // IRM      nformation Request Message -> Trame interrogative
Trame T_IM_Commodo_EG;                  //      pour co      ler le Module 8E sur Commodo Essuie Glace
                                        // IM -      rmation Message -> Trame de commande
Trame T_IM_Asservissemen;               // Tra      our co      ule "Asservissement"
                                        //      nformation Message -> Trame de commande

// Initialisations
//*****************

/* Initialisation DU SJA1000 de la        04 */
Init_Aton_CAN();
// Pour initialiser les lia        s  no      mmodo_EG"
T_IM_Commodo_E      c_Im      =0x00;
T_IM_Commodo      info.c      tend=1; // On travaille en mode étendu
T_IM      do_E      fo.cha      =0x03; // Il y aura 3 données de 8 bits (3 octets envoyés)
T      do_EG.id      ateur.ident=Ident_T_IM_Commodo_EG;
      Commodo_EG.data[0]      // première donnée -> "Adresse" du registre concernée
                                //(GPDDR donne la direction des I/O) adresse = 1Fh  page 16
T      =0x7F;      ne donnée -> "Masque"
                                //-> Tous les bits concernés sauf GP0 (voir doc page 16)
T_IM_Commodo_EG.da      0x7F; // troisième donnée -> "Valeur" -> Tous les bits en entrée
// Envoi trame p      a direction des entrées sorties
Ecrire_Trame(T      do_EG);  // Envoyer trame sur réseau CAN
        Cptr_T      t=0;
        do{Cptr_TimeOut++;}while((Lire_Trame(&Trame_Recue)==0)&&(Cptr_TimeOut<2000));
        if(Cptr_TimeOut==2000)
                { clsscr();
                 gotoxy(2,12);
                 printf(" Pas de reponse a la trame de commande en initialisation \n");
                 printf(" Couper et/ou  remettre    alimentation 12 V  \n");
                 do{}while(Lire_Trame(&Trame_Recue)==0); // On attend les trames  "On Bus"
      for(Cptr_TimeOut=0;Cptr_TimeOut<200000;Cptr_TimeOut++);
      Ecrire_Trame(T_IM_Commodo_EG);  // Renvoyer trame IM sur réseau CAN
                 Cptr_TimeOut=0;
                 do{Cptr_TimeOut++;}while((Lire_Trame(&Trame_Recue)==0)&&(Cptr_TimeOut<2000));
                 if(Cptr_TimeOut==2000)
```

```
                                { gotoxy(2,12);
                                 printf(" Pas de reponse a la trame de commande en initialisation \n");
                                 printf("  Modifier le programme et recommencer  \n");
                        do{}while(1); // On reste bloqué, on ne continue pas
                                 }
                        }
// Pour activer les conversions Ana -> Num
T_IM_Commodo_EG.data[0]=0x2A;          // Adresse du registre ADCON0
T_IM_Commodo_EG.data[1]=0xF0;          // Masque -> bits 7..4 concernés
T_IM_Commodo_EG.data[2]=0x80;          // Valeur ->  ADON=1 et "prescaler rate"=1:32
Ecrire_Trame(T_IM_Commodo_EG);
do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre réponse
// Pour définir le mode de conversion
T_IM_Commodo_EG.data[0]=0x2B;          // Adresse du registre ADCON1
T_IM_Commodo_EG.data[1]=0xFF;          // Masque -> tous les bits sont concernés
T_IM_Commodo_EG.data[2]=0x0C;          // Valeur ->  voir doc MCP25050 page 36
Ecrire_Trame(T_IM_Commodo_EG);
do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre réponse
// Pour initialiser le module "Asservissement"
// Trame de type "IM" (trame de commande): Données d'identification pour le noeud "Asser
T_IM_Asservissement.trame_info.registre=0x00;
T_IM_Asservissement.trame_info.champ.extend=1;
T_IM_Asservissement.trame_info.champ.dlc=0x03;
T_IM_Asservissement.trame_info.champ.rtr=0;
T_IM_Asservissement.ident.extend.identificateur.ident=Ident_T_IM_Asservisse
// Pour définir des Entrées/Sorties
T_IM_Asservissement.data[0]=0x1F;          // Adresse du registre GPDDR  (dire
T_IM_Asservissement.data[1]=0xEF;          // Masque -> Bit 7 non concerné
T_IM_Asservissement.data[2]=0xE3;          // Valeur ->  1 si Entrée et 0
            // GP7=fs Entrée; GP6=fcg Entree; GP5=fcd Entrée;  GP4=
            // GP3=PWM2 Sortie; GP2=PWM1 Sortie; GP1=AN1 Entr
Ecrire_Trame(T_IM_Asservissement);
do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre trame
// Pour mettre à 0 les sorties
T_IM_Asservissement.data[0]=0x1E;          // Ad
T_IM_Asservissement.data[1]=0x1C;          //
T_IM_Asservissement.data[2]=0x00;          // Va
Ecrire_Trame(T_IM_Asservissement);
do{}while(Lire_Trame(&Trame_Recue)==0); // Attendr
// Pour définir sortie GP2 en PWM1
T_IM_Asservissement.data[0]=0x21;          //
T_IM_Asservissement.data[1]=0xB3;          //
T_IM_Asservissement.data[2]=0x80;          //
Ecrire_Trame(T_IM_Asservissement);
do{}while(Lire_Trame(&T
// Pour définir fréquence sig
T_IM_Asservissement.data[0
T_IM_Asservissement.data[1]=0
T_IM_Asservissement.data[2]=0
Ecrire_Trame(T_IM_Asservisse
do{}while(Lire_Trame(&Trame_R

// Pour trame interroga
// Définir données d'id
T_IRM_Commodo
T_IRM_Commodo_
T_IRM_Commodo_EG.
T_IRM_Commodo_EG.tra
T_IRM_Commodo_EG.ident.ex
                                                                    // Voir définitions dans fichier CAN_VMD.h
Ecrire_Trame(T_IRM_Commodo_EG); // Envoi une première trame interrogative pour initialiser la mémoire
do{}while(Lire_Trame(&Trame_Recue)==0);
Commodo_EG.valeur=Trame_Recue.data[1]&0x10;
Commodo_EG.valeur=Commodo_EG.valeur;  // On met en mémoire l'état actuel pour comparaison ultéérieure

Val_Vitesse=Trame_Recue.data[2]; // On récupére les MSB de la conversion A->N  voie AN0
// Pour initialiser rapport cyclique PWM1 -> c'est le module de la commande
T_IM_Asservissement.data[0]=0x25;          // Adresse du registre PWM1DC
T_IM_Asservissement.data[1]=0xFF;          // Masque -> tous les bits sont consernés
T_IM_Asservissement.data[2]=Val_Vitesse;          // Valeur ->PWM1DC   commande égale à 50% (car valeur maxi=255)
Ecrire_Trame(T_IM_Asservissement);  // Le moteur ne peut tourner car ValidIP=0
do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre réponse
Val_Vitesse_Mem=Val_Vitesse;  // On met en mémoire la valeur pour comparaison ultéérieure

// Les initialisations se sont bien passée
// On peut afficher le titre du TP
clsscr();
```

```
gotoxy(1,2);
printf("    ************************************************************ \n");
printf("     TPs sur Reseau CAN    Application: Commande Essui-glace      \n");
printf("     ------------------------------------------------------- \n");
printf("     TP n° 4        Marche/Arret du moteur                    \n");
printf("     Le moteur est en etat MARCHE si on appuie sur BP1 (GP4)      \n");
printf("     Modifier l'entrée analogique pour modifier la vitesse        \n");
printf("    ************************************************************ \n");

// Pour afficher l'état initial
            gotoxy(2,10);
            printf("  Etat entree GP4 (BP1) = %d \n",Commodo_EG.bit.GP4);
            if(Commodo_EG.bit.GP4)printf("   Moteur a l'arret    \n");
            else {printf("   Moteur en marche    \n");
                        T_IM_Asservissement.data[0]=0x1E;         // Adresse du registre GPLAT  (Registre I/O)
                        T_IM_Asservissement.data[1]=0x10;         // Masque -> seule la sortie ValidIP sera consernée
                        T_IM_Asservissement.data[2]=0x10; // ValidIP =1 donc moteur en marche
                        Ecrire_Trame(T_IM_Asservissement);  // Le moteur ne peut tourner car V
                        do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre réponse
                }
            gotoxy(2,14);
            printf("  Valeur entree analogique = %d \n",Val_Vitesse);  // On affiche sa valeur

// Boucle principale
//*******************
  do
            {  Ecrire_Trame(T_IRM_Commodo_EG); // On envoi trame interroga
       // On attend la réponse
              do{}while(Lire_Trame(&Trame_Recue)==0);  // La fonction
                        if (Trame_Recue.ident.extend.identificateur.iden
                                    { // On a reçu l'état du commodo donc
                                    // Pour l'entrée Marche/Arret
                                    Commodo_EG.valeur=Trame_R
                                    if(Commodo_EG.valeur!=Com
                                                {// S'il y a eu cha
                                          gotoxy(2,
                                          printf("  E
                                                if(Comm
                                                {// BP1 est
                                                 T_IM_Asserv
                                                 T_IM_Asservissement.data[1]=0x10;       // Masque -> seule la sortie ValidIP sera consernée
                                                 T_IM_Asservissement.data[2]=0x00;       // ValidIP =0 donc moteur à l'arret
                                                 printf("   Moteur a l'arret    \n");
                                                 Ecrire_Trame(T_IM_Asservissement);
                                                 do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre réponse
                                                }
                                                else
                                                {// Si ce n'est pas le cas contraire BP1 est appuyé
                                                 T_IM_Asservissement.data[0]=0x1E;       // Adresse du registre GPLAT  (Registre I/O)
                                                 T_IM_Asservissement.data[1]=0x10;       // Masque -> seule la sortie ValidIP sera consernée
                                                 T_IM_Asservissement.data[2]=0x10; // ValidIP =1 donc moteur en marche
                                                 printf("   Moteur en marche    \n");
                                                 Ecrire_Trame(T_IM_Asservissement);
                                                 do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre réponse
                                                }
                                          // Si changement
                                          Commodo_EG_Mem=Commodo_EG.valeur;  // On met en mémoire l'état actuel
                                          // Entrée analogique
                                          Val_Vitesse=Trame_Recue.data[2]; // On récupére les MSB de la conversion A->N  voie AN0
                                          if(Val_Vitesse!=Val_Vitesse_Mem)
                                           {
                                            gotoxy(2,14);
                                            printf("  Valeur entree analogique = %d \n",Val_Vitesse);  // On affiche sa valeur
                                           // Pour modifier le rapport cyclique PWM1 -> c'est le module de la commande
                                           T_IM_Asservissement.data[0]=0x25;       // Adresse du registre PWM1DC
                                           T_IM_Asservissement.data[1]=0xFF;       // Masque -> tous les bits sont consernés
                                           T_IM_Asservissement.data[2]=Val_Vitesse;            // Valeur ->PWM1DC
                                           Ecrire_Trame(T_IM_Asservissement);  // Le moteur ne peut tourner car ValidIP=0
                                           do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre réponse
                                           }
                                          Val_Vitesse_Mem=Val_Vitesse;
                                          } // Fin si identificateur correct
            }while(1);// FIN de la boucle principale
}// FIN fonction principale
```

Page laissé vierge

3 TP N°3: FAIRE BATTRE LE BALAI D'ESSUIE GLACE

3.1 Sujet:

Objectifs :	- Définir les différentes trames interrogatives ou de commande à faire transiter par le réseau CAN en fonction d'une action souhaitée. - Commander un actionneur électrique (moteur à courant continu), dans les deux sens de rotation par l'intermédaire d'un pré-actionneur commandable par réseau CAN. - Acquérir l'état de capteurs TOR (fins de course) acessibles par réseau CAN et en déduire les actions à mener pour satisf[illegible] cahier des charges imposé.
Cahier des charges :	Après avoir fait tourner le moteur da[illegible]ns pos[illegible] (déplacement à droite) jusqu'à atteindre l'action sur l[illegible] cou[illegible]it, l'essuie glace réalise le cycle continu suivant: → déplacement à gauche j[illegible]tio[illegible]r le fin de course gauche → déplacement à droite jus[illegible]io[illegible]le fin de course droite → etc

3.2 Eléments de solution

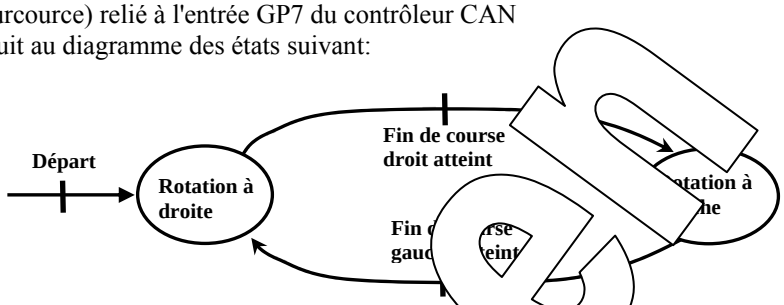
3.2.1 Analyse

Principe:

Le module d'interface CAN mis en œuvre dans ce TP est le module repéré "Asservissement".
Ce module permet la commande d'un moteur à courant continu 24V/ 1A, dans les deux sens de rotation, en mode "PWM" (modulation de largeur d'impulsion). Cette possibilité a été expérimentée dans le TP5.
Il permet d'acquérir 3 entrées TOR sur lesquelles on pourra connecter les capteurs de fin de courses:

→ fcd (fin de course droit) relié à l'entrée GP5 du contrôleur CAN
→ fcg (fin de course gauche) relié à l'entrée GP6 du contrôleur CAN
→ fs (fin de surcource) relié à l'entrée GP7 du contrôleur CAN

Le cycle demandé conduit au diagramme des états suivant:



Configurations du module "Asservissement" pour l'utilisation envisagée:

Comme dans le TP n°2, il faut envoyer un certain nombre de "Trames" afin de configurer le module "Asservissement"
Trame n°1 → *pour définir les entrées et sorties du module "Asservissement"*
Il faut initialiser le registre GPDDR ("Data Direction Register") -> Idem TP n°2

Trame n°2 → *pour initialiser la sortie CCP2 en sortie PWM1 (commande du moteur dans le sens positif)*
-> Idem TP n°2

Trame n°3 → *pour définir la fréquence de la sortie PWM1:* -> Idem TP n°2
Trame n°4 → *pour définir le rapport cyclique de la sortie PWM1 (module de la commande donc de la vitesse du moteur)* -> Idem TP n°2

Trame n°5 → *pour initialiser la sortie CCP2 en sortie PWM2 (commande du moteur dans le sens négatif)*
D'après la notice technique du MCP25050 (pages 30 à 32), la génération du signal PWM2 se fait à partie du "Timer2" et la fréquence du signal est choisie grâce au registre "T2CON" d'adresse 06_H (page 15 Doc MCP25050).

bit 7 =1 : Validation du "Timer 2"
bits 5:4 : sélectionner à 0 pour avoir un coefficient de division de fréquence égal à 1
"TMR2 to PR2 match value" = 1)

```
T_IM_Asservissement.data[0]=0x22; // Adresse du registre T1CON en écriture
 (doc MCP25050 p15) 06H + décalage = 06H + 1CH = 22H
T_IM_Asservissement.data[1]=0xB3; // Masque sur le registre (doc MCP25050 p32)
T_IM_Asservissement.data[2]=0x80; // Valeur à charger dans le registre adressé
```

Suite à ces définitions, il faudra envoyer la trame puis attendre la réponse de type "Ack".

Trame n°6 → *pour définir la fréquence de la sortie PWM2:*
Cette fréquence dépend de la valeur chargée dans le registre "PR2"

```
T_IM_Asservissement.data[0]=0x24; // adresse du registre PR2 en écriture
 (doc MCP25050 p15) 08H + décalage = 08H + 1CH = 24H
T_IM_Asservissement.data[1]=0xFF; // Masque sur le registre (doc MCP25050 p32)
T_IM_Asservissement.data[2]=0xFF; // On chargera 255 dans le registre
```

La fréquence du quartz implanté sur la carte "asservissement" étant égale à 16Mhz, la fréquence du signal PWM2 sera donc égale à : $F_{PWM} = 16.10^6/(4.256) = 15{,}6$ KHz
Ce qui est une fréquence correcte pour piloter un moteur en PWM (fréquence sensiblement inaudible).
Suite à ces définitions, il faudra envoyer la trame puis attendre la réponse de type "Ack".

Trame n°7 → *pour définir le rapport cyclique de la sortie PWM2 (module de la commande donc de la vitesse du moteur)*

```
T_IM_Asservissement.data[0]=0x25; //adresse du registre PWM2DCH en écriture
 (doc MCP25050 p15) 0A_H + décalage = 0A_H + 1C_H = 26_H
T_IM_Asservissement.data[1]=0xFF; //masque sur le registre (doc MCP25050 p33)
T_IM_Asservissement.data[2]=0x7F; // Commande =127 (0xFF=255 pour la commande Maxi)
```

Suite à ces définitions, il faudra envoyer la trame puis attendre la réponse de type "Ack".

Acquisition de l'état des fins de course:

Définition de la trame interrogative permettant de connaître l'état des fins de course:

Dans ce cas, la trame envoyée par le contrôleur CAN (Circuit SJA1000 sur carte CAN_PC104) sera vue par le récepteur (circuit MCP25050 sur module) comme une IRM "Information Request Message", avec la fonction "Read register" (voir documentation technique du MPC25025 pages 22).

D'après le tableau donné page 22 de la notice du MCP25050, l'identificateu contiendra l'adresse du registre lu. Cette adresse est placée sur les bits ID15 à ID8 de l'identificateur en r ui seront réceptionnés et placés dans le registre RXBEID8). Le registre concerné est GPPIN d'adr " (vo mentation technique du MPC25025 pages 37).
D'autre part, les trois bits de poids faibles de l'identificateur en mo devr positionnés à 1.
L'identificateur défini dans le chapitre 1 devra donc être complé

00 84 xx xx — D'après tableau
00 84 1E 07 — Adresse registre — C ctio — ls oit nt pris en compte)

→ Définition de variables structurées sous le m **Trame**

```
Trame T_IRM_Acquisition_FC;      // Trame  interrog on du module asservissement pour acquérir Fins de Courses.
```

Remarque: La variable structurée **M_Acq on_F** mportera 5 octets utiles seulement, 1octet pour trame_info et 4 octets pour l'identificat mo du (qui comprendra l'adresse du registre concerné par la lecture.

→ Accès et définition des différen ents d iable structurée **"Lecture_FC"**

```
T_IRM_Acquisition_FC.tram  fo.registre 0x00;  //On initialise tous les bits à 0
T_IRM_Acquisition_FC.tram  nfo.ch extend=1; //On travaille en mode étendu
T_IRM_Acquisition_FC.tram  fo.cha dlc=0x01; //Il y aura 1 octet de données demandé
T_IRM_Acquisit  _FC.ident  ntificateur.ident=0x00841E07;
```

Suite à ces définitions, il fa
- envoyer la tra r la ion **Ecire_Trame(T_IM_Asservissement)**
- puis attendre la ns ty OM" en utilisant la fonction **Lire_Trame(&T_Recue)**

D'après la définitio ident ée en Annexe, une trame de réponse à une IRM a le même identificateur que la trame interrogat été à ême.
Vu du module (P25 réponse à un "IRM" (Information Request Message) est un "OM" (Output Message).
La différence a me in ative origine est que cette trame réponse comporte le paramètre "value" (au rang 0 de la ta" de l se). Ce paramètre est l'image des entrées. On récupère donc l'état des différents ars.

Déf var structs images de l'état des fins de course

La trame reçue en ré e à cette trame interrogative comportera en data[0], l'état des fins de course. On recopie cette donnée reçue ariable image.

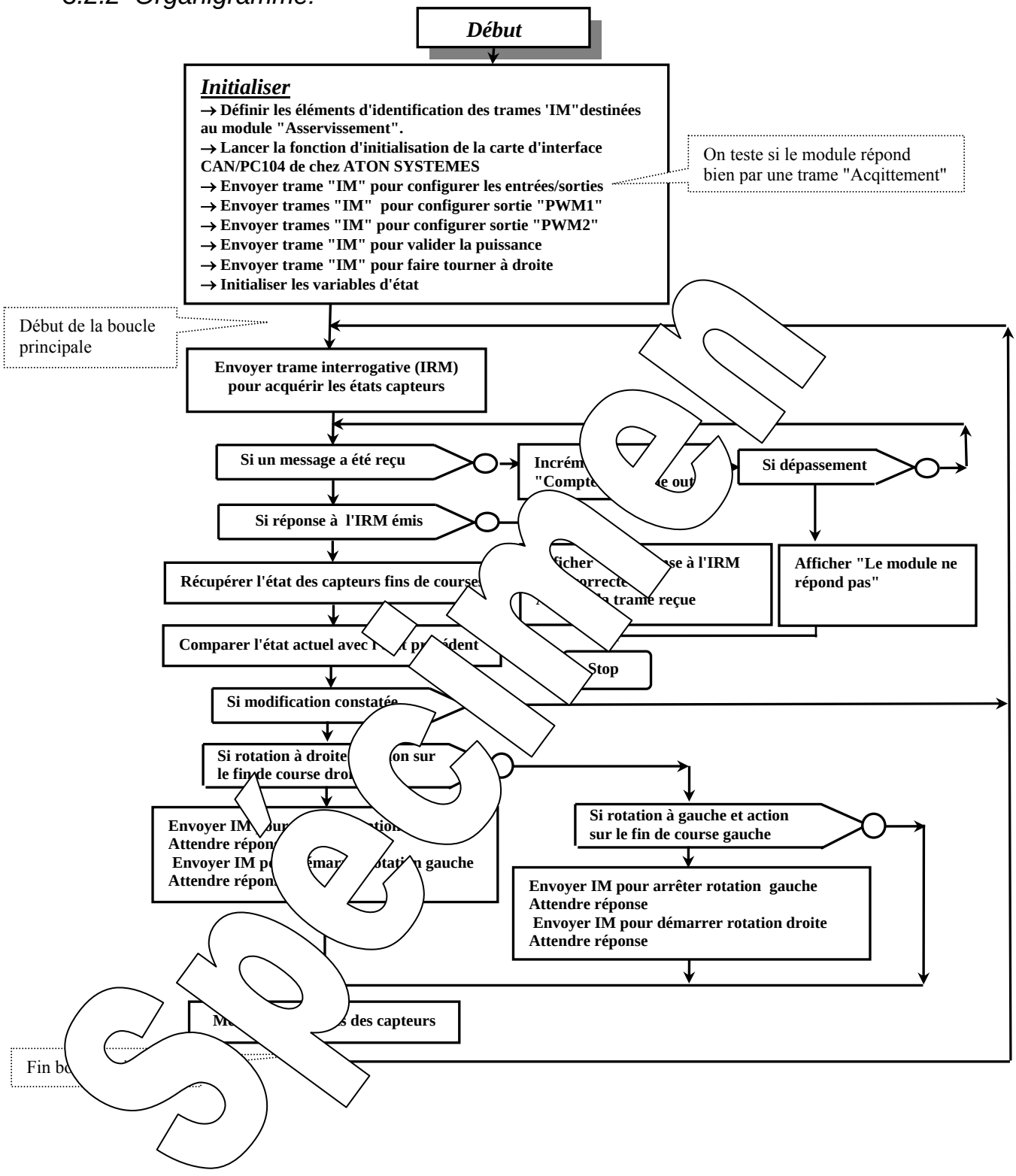
```
union byte_bits FC;
#define Etats_FC FC.valeur // Pour l'ensemble des états fins de courses
#define fs FC.bit.b7  // Pour fin de surcourse
#define fcg FC.bit.b6 // Pour fin de course gauche
#define fcd FC.bit.b5 // Pour fin de course droit
```

Afin de pouvoir détecter les changements d'état des capteurs, on mémorise l'état dans une deuxième variable structurée -> Etat_mémorisé

```
union byte_bits FC_Mem; // Pour fins de courses mémorisés
#define Etats_FC_Mem FC_Mem.valeur  // Pour l'ensemble des états mémorisés
```

Si l'état d'une variable acquise est différente de sa valeur mémorisée, c'est qu'il y a eu un changement d'état. On en déduit alors les actions à mener.

3.2.2 Organigramme:



3.2.3 Programme en "C"

```
/*****************************************************************************
*      TPs sur  EID210 / Réseau CAN - VMD  (Véhicule Multiplexé Didactique)
*****************************************************************************
*      APPLICATION: Commande Essuie-glace à distance
*****************************************************************************
*   TP n°3:   Battement essuie glace avant
*---------------------------------------------------------------------------
*   CAHIER DES CHARGES :
*  *********************
*  On souhaite que l'essuie glace réalise le cycle suivant:
*              - rotation droite jusqu'à atteindre le fin de course droit
*              - arrêt sur fin de course droit et démarrage en rotation gauche
*              - rotation gauche jusqu'à atteindre le fin de course gauche
*              - arrêt sur fin de course gauche et démarrage en rotation droite
*              etc ...
*  La structure de programme sera de type "boucle de scrutation".
*  On affichera à l'écran, dans quel état on est.
*---------------------------------------------------------------------------
*  NOM du FICHIER :  TP_3_CAN_VMD_EG.C
* *****************
*****************************************************************************/

// Déclaration des fichiers d'inclusion
#include <stdio.h>
#include"Structures_Donnees.h"
#include"cpu_reg.h"
#include "eid210_reg.h"
#include "CAN_vmd.h"
#include "Aton_can.h"

// Déclaration des variables
// Pour les Indicateurs divers (variables binaires)
union byte_bits Indicateurs,FC,FC_Mem; // Stru  s de bi
#define I_Sens_Rotation Indicateurs.bit.b0
#define I_Attente_Reponse_IRM Indicateurs.bit.b1
#define I_Message_Pb_Affiche Indicateurs.bit.b2
// Pour les fins de course
#define Etat_FC FC.valeur // Pour l'ensemble des f  de cour
#define fs FC.bit.b7   // Pour fin de surcourse
#define fcg FC.bit.b6 // Pour fin de course ga
#define fcd FC.bit.b5 // Pour fin de course d
#define Etat_FC_Mem FC_  em.valeur // Po  émorisati  fins de courses
// Déclarations des diverses  nes de commun
Trame Trame_Recue;// Pou  tram  t d  ontroleur
#define Ident_Trame_Recue Tram  xten  cateur.ident
// Trmes de type "IM" (Input Me  -> tra  commande)
Trame T_IM_Asservissement;  r la  han  u moteur
Trame T_IRM_Acquisition_FC;  isiti  l'état des fins de courses EG
// Déclaration constantes
#define Module_Vitess

//=============  ===
// FONCTION PR  E
//=  =====
m

/ IAL
//
// D  variabl  es à la fo  n principale
int Cptr_Affichage=0,Cp  eOut;

Init_Aton_CA
clsscr();
// Trame de type "IM" (trame de commande): Données d'identification
T_IM_Asservissement.trame_info.registre=0x00;
T_IM_Asservissement.trame_info.champ.extend=1;
T_IM_Asservissement.trame_info.champ.dlc=0x03;
T_IM_Asservissement.trame_info.champ.rtr=0;
T_IM_Asservissement.ident.extend.identificateur.ident=Ident_T_IM_Asservissement;
// Pour définir des Entrées/Sorties
T_IM_Asservissement.data[0]=0x1F;        // Adresse du registre GPDDR  (direction de E/S)
                                                    // doc MCP25050 Page 16
T_IM_Asservissement.data[1]=0xEF;        // Masque -> Bit 7 non concerné
T_IM_Asservissement.data[2]=0xE3;        // Valeur ->  1 si Entrée et 0 si Sortie
```

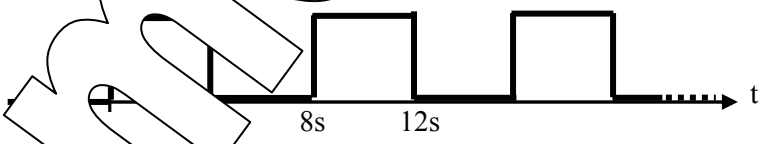
```
            // GP7=fs Entrée; GP6=fcg Entree; GP5=fcd Entrée;  GP4=ValidIP Sortie;
            // GP3=PWM2 Sortie; GP2=PWM1 Sortie; GP1=AN1 Entrée; GP0=AN0 Entrée;
I_Message_Pb_Affiche=0;
do {Ecrire_Trame(T_IM_Asservissement); // C'est la première trame envoyée
             Cptr_TimeOut=0;
             do{Cptr_TimeOut++;}while((Lire_Trame(&Trame_Recue)==0)&&(Cptr_TimeOut<2000));
             if(Cptr_TimeOut==2000)
                        { gotoxy(2,12);
                          printf(" Pas de reponse a la trame de commande en initialisation \n");
                          printf(" Couper et/ou  remettre    alimentation 12 V  \n");
                          do{}while(Lire_Trame(&Trame_Recue)==0); // On attend les trames  "On Bus"
          for(Cptr_TimeOut=0;Cptr_TimeOut<100000;Cptr_TimeOut++);
          Ecrire_Trame(T_IM_Commodo_EG);  // Renvoyer trame IM sur réseau CAN
                          Cptr_TimeOut=0;
                          do{Cptr_TimeOut++;}while((Lire_Trame(&Trame_Recue)==0)&&(Cptr_TimeOut<2000));
                          if(Cptr_TimeOut==2000)
                                    { gotoxy(2,12);
                                      printf(" Pas de reponse a la trame de commande en initialisatio
                                      printf("  Modifier le programme et recommencer  \n");
                                   do{}while(1); // On reste bloqué, on ne continue pas
                                    }
                         }
// Pour mettre à 0 les sorties
T_IM_Asservissement.data[0]=0x1E;         // Adresse du registre GPLAT  (Registr
T_IM_Asservissement.data[1]=0x1C;         // Masque -> sorties GP4,3,2 sont cor
T_IM_Asservissement.data[2]=0x00;         // Valeur ->  les 3 sorties à 0
Ecrire_Trame(T_IM_Asservissement);
do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre réponse
// Pour définir sortie GP2 en PWM1
T_IM_Asservissement.data[0]=0x21;         // Adresse du registre T1C
T_IM_Asservissement.data[1]=0xB3;         // Masque -> seuls bit 7;5;4
T_IM_Asservissement.data[2]=0x80;         // Valeur ->  TMR1ON      caler
Ecrire_Trame(T_IM_Asservissement);
do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre répons
// Pour définir fréquence signal sortie PWM1
T_IM_Asservissement.data[0]=0x23;         // A      du regi
T_IM_Asservissement.data[1]=0xFF;         // M      -> ous les      onserne
T_IM_Asservissement.data[2]=0xFF;         // Valeur      255
Ecrire_Trame(T_IM_Asservissement);
do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre re
// Pour définir sortie GP3 en PWM2
T_IM_Asservissement.data[0]=0x22;         //      registre T2
T_IM_Asservissement.data[1]=0xB3;         //      ue -> seuls      7;5;4;;0 consernés
T_IM_Asservissement.data[2]=0x80;         //      r ->  TM      =1; Prescaler2=1
Ecrire_Trame(T_IM_Asser      ement);
do{}while(Lire_Trame(&T      e_Recue)==0); //
// Pour définir fréquence sign      Sort
T_IM_Asservissement.data[0]=0      sse du registre PR2
T_IM_Asservissement.data[1]=0      asqu      -> tous les bits sont consernés
T_IM_Asservissement.data[2]=0      Val      PR2=255
Ecrire_Trame(T_IM_Asservissem
do{}while(Lire_Trame(&      Rec      dre réponse
// Pour initialiser rappo      VM1
T_IM_Asservisseme      t      // Adresse du registre PWM1DC
T_IM_Asservisse      [1]=0x      // Masque -> tous les bits sont consernés
T_IM_      vissem      =Modu      esse;         // Valeur ->  PWM1DC
Ec      T_IM_A
d      re_Trame(&T      =0); // Attendre réponse
      initiali      ycliq      à 0
T      l]=0x2      // Adresse du registre PWM2DC
T      =0xFF;      // Masque -> tous les bits sont consernés
T_IM_      ssement.da      0;          // Valeur ->  PWM2DC=0
Ecrire_Trame(T_      ement);
do{}while(Lire      ne_Recue)==0); // Attendre réponse
// Pour Valider le      puissance
T_IM_Asservissement.data[0]=0x1E;         // Adresse du registre GPLAT  (Registre I/O)
T_IM_Asservissement.data[1]=0x10;         // Masque -> sortie GP4 (ValidIP) est conserné
T_IM_Asservissement.data[2]=0x10;         // Valeur ->  ValidIP=1
Ecrire_Trame(T_IM_Asservissement);
do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre réponse
// Masque pour les commandes IM futures
T_IM_Asservissement.data[1]=0xFF;         // Masque -> tous les bits sont consernés
// Pour acquérir l'état des fin de course
// Trame de type "IRM" (trame interrogative): Données d'identification
T_IRM_Acquisition_FC.trame_info.registre=0x00;
T_IRM_Acquisition_FC.trame_info.champ.extend=1;
T_IRM_Acquisition_FC.trame_info.champ.dlc=1;
```

```
T_IRM_Acquisition_FC.trame_info.champ.rtr=1;
T_IRM_Acquisition_FC.ident.extend.identificateur.ident=Ident_T_IRM1_Asservissement;
 Ecrire_Trame(T_IRM_Acquisition_FC);    // Envoi trame pour acquérir états fins de course
do{}while(Lire_Trame(&Trame_Recue)==0);            //On attend la réponse
            Etat_FC=~Trame_Recue.data[0];          // On récupére l'état des fin de course
            Etat_FC_Mem=Etat_FC;                              // On le mémorise
// Initialisation des variables diverses
            I_Sens_Rotation=0;

// Pour afficher titre
gotoxy(1,6);
printf(" ********************************************************************** \n");
printf("  TPs sur Reseau CAN  Application: Commande Essui-glace a distance  \n");
printf(" ---------------------------------------------------------------- \n");
printf("   TP  n°3    Battement du BALAI D'ESSUI GLACE              \n");
printf(" ********************************************************************** \n");

// BOUCLE PRINCIPALE
//*******************
while(1)
            {
            // Pour "Acquérir l'état des fins de course"
            Ecrire_Trame(T_IRM_Acquisition_FC); // Envoi trame pour acquérir états fins de co
            Cptr_TimeOut=0;
            do{Cptr_TimeOut++;}while((Lire_Trame(&Trame_Recue)==0)&&(C                    000));
             if(Cptr_TimeOut==10000)
                        {clsscr(),gotoxy(2,10);
                         printf(" Pas de reponse a la trame interrogative pour fins de                );
                         printf("  Il faut recharger et relancer le programme \n");
                         do{}while(1);} // Stop
             else { if(Trame_Recue.ident.extend.identificateur.ident==Id                    Asservissement)
                                    // Si identificateur attendu
                                     {Etat_FC=~Trame_Recue.data[                    // On re                tat des fin de course
                                     if(Etat_FC!=Etat_FC_Mem)                    "il y a eu                e modification
                                                                                 tat des fin de courses -> on traite
                                     {Etat_FC_Me                 t_FC;           morisa                uvel état
                 // Traiter le changement de l'état
                        if(I_Sens_Rotation)   // Si on to                négatif
                                    {if(fcg)                                    arrive                course gauche
                                                {T_IM_Asserv                    ata[2]=                    // On arrête la rotation négative
                                                T_I                 vissem                  0]=0x26;        // Adresse du registre PWM2DC
                                                E                       ne(T_IM_A                       ement);
                                                Lire_Trame                  Trame_Recue)==0){};             // On attend la réponse
                                                Asservis                    data[2]=Module_Vitesse; // On commande la rotation positive
                                                T                   sservisse                   data[0]=0x25;       // Adresse du registre PWM1DC
                                                Ecr                 (T                  Asservissement);
                                                while(L                 Trame_Recue)==0){};             // On attend la réponse
                                                I_Rotation=!I_Sens_Rotation;                            // On change d'état

                                    else                    to                  en r                f
                                                                    // Si on est arrivé en fin de course droit
                                                                    servissement.data[2]=0;             // On arrête la rotation positive
                                                                    sservissement.data[0]=0x25;         // Adresse du registre PWM1DC
                                                                    Ecrire_Trame(T_IM_Asservissement);
                                                                    while(Lire_Trame(&Trame_Recue)==0){};           // On attend la réponse
                                                                    T_IM_Asservissement.data[2]=Module_Vitesse; // On commande la rotation négative
                                                                    T_IM_Asservissement.data[0]=0x26;       // Adresse du registre PWM2DC
                                                                    Ecrire_Trame(T_IM_Asservissement);
                                                                    while(Lire_Trame(&Trame_Recue)==0){};           // On attend la réponse
                                                                    I_Sens_Rotation=!I_Sens_Rotation;               // On change d'état

                                    }}}                 "Traiter ch                gement d'état"
            } //  Fi                boucl                ipale
} // Fin fonctio
```

Page laissée vierge

4 TP N°4: SELECTION DE COMMANDE MOTEUR ESSUIE GLACE

4.1 Sujet:

Objectifs :	- Définir les différentes trames interrogatives ou de commande à faire transiter par le réseau CAN en fonction d'une action souhaitée. - Adapter le commande d'un actionneur électrique (moteur à courant continu) par réseau CAN, en fonction de l'état de signaux d'entrées (binaires et analogique) accessibles par réseau CAN, afin de satisfaire un cahier des charges imposé.
Cahier des charges :	L'état de fonctionnement du moteur "E... Glace" dépend de l'état du commodo Essuie Glace avec le cahie... ...es suivant: - Si Pos1 (GP4) actionné, le mote... ...ne a... ...itesse fixe. - Si Pos2 (GP5) actionné, le moteu... ...à gra... ...itesse fixe. - Si Pos3 (GP6) actionné, le moteur tou... ...itesse ajustable imposée par l'entrée analogique AN0 du ... ...mm... ...G., - Si Pos4 (GP7) actionné, ...oteu... ...an fonctionnement marche/arrêt cyclique avec, dans l'état "marche", ...se a... ...able imposée par l'entrée analogique AN0.  - Si ni P... ...GP4), ...(GP5), ni Pos3 (GP5) est actionné, le moteur reste à l'arrêt

4.2 Eléments de solution

4.2.1 Acquisition des entrées "Commodo Essuie Glace"

La technique d'acquisition des entrées binaires et de l'entrée analogique, par réseau CAN, du module "Commodo E.G" est décrite dans le TP n°1.

4.2.2 Commande du moteur Essuie Glace

La technique de commande du moteur Essuie Glace est décrite dans le TP n°2.

4.2.3 Déclaration et utilisation de variables binaires

Si on souhaite déclarer des variables binaires, il est possible d'utiliser la structure de données de type "union" de label "byte_bits" déclarée dans le fichier "Structures_Données.h"

```
/*      Union pour accéder à un octet (BYTE) soit en direct, soi          dualisant les 8 bits
//*************************************************************          ********************/
        union byte_bits
        {       struct
                 {      unsigned char b7:1;
                        unsigned char b6:1;
                        unsigned char b5:1;
                        unsigned char b4:1;
                        unsigned char b3:1;
                        unsigned char b2:1;
                        unsigned char b1:1;
                        unsigned char b0:1;
                }bit;
                BYTE valeur;
        };
```

Utilisation:

☞ Dans la partie déclarative du pr me

```
/* Déclaration des variables*/
// Pour les variables binaires
union byte_bits Indicateurs; // Structu        bits
#define User_Bit1 Indicateurs.bit_b0  //          ation variable binaire
#define User_Bit2 Indicateurs.bi      //
```

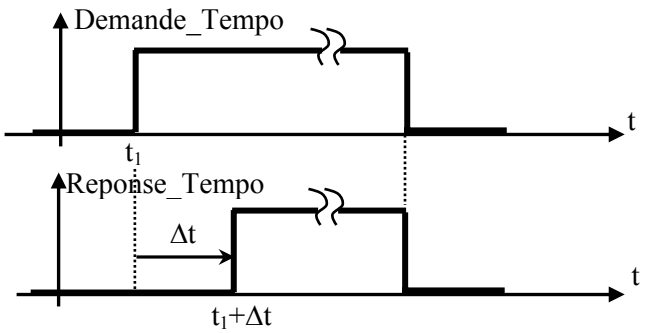
☞ Utilisation des variables binai ans les tions

```
// Pour initialiser  ous les 8      de l'   mble "Indicateurs"
Indicateurs.valeur=  00;  // tou        nt initialisés à 0
// Pour affecter une vai      v      inaire
User_Bit1=1;
User_Bit0=0;
// Pour tester la vale      'u  var    le binaire
if(User_Bit1==1)        {       ions   faire si valeur = 1
                        …

else    { // Ac          si valeur = 0
          …

if      t2==0)          {       ions à faire si valeur = 0
```

4.2.4 Utilisation d'une base de temps

Le micro contrôleur implanté sur la carte processeur EID210 intégre un bloc fonctionnel (noté "Time Processor Unit (TPU)" permettant la réalisation de fonctions temporelles.
L'une des nombreuses fonctions du TPU est la génération d'une interruption périodique.

Exemple d'application:
Soit la génération d'une temporisation égale à 1 S, comme le montre les signaux ci-desous.



Si on envisage une base de temps élémentaire égale à 1mS, la solution comportera les parties de programmes suivantes:

☞ Dans la partie déclarative du programme

```
/* Déclaration des variables*/
// Pour les variables binaires
union byte_bits Indicateurs; // Structures de bits
#define Demande_Tempo Indicateurs.bit.b0  //  déclaration variables
#define Reponse_Tempo Indicateurs.bit.b1  //  pour temporisation
#define Tempo_En_Cours Indicateurs.bit.b2
// Pour le compteur de passage dans la fonction d'interruption
int CPTR_mS;   /* déclaration des variables et compteurs de millisecondes */
```

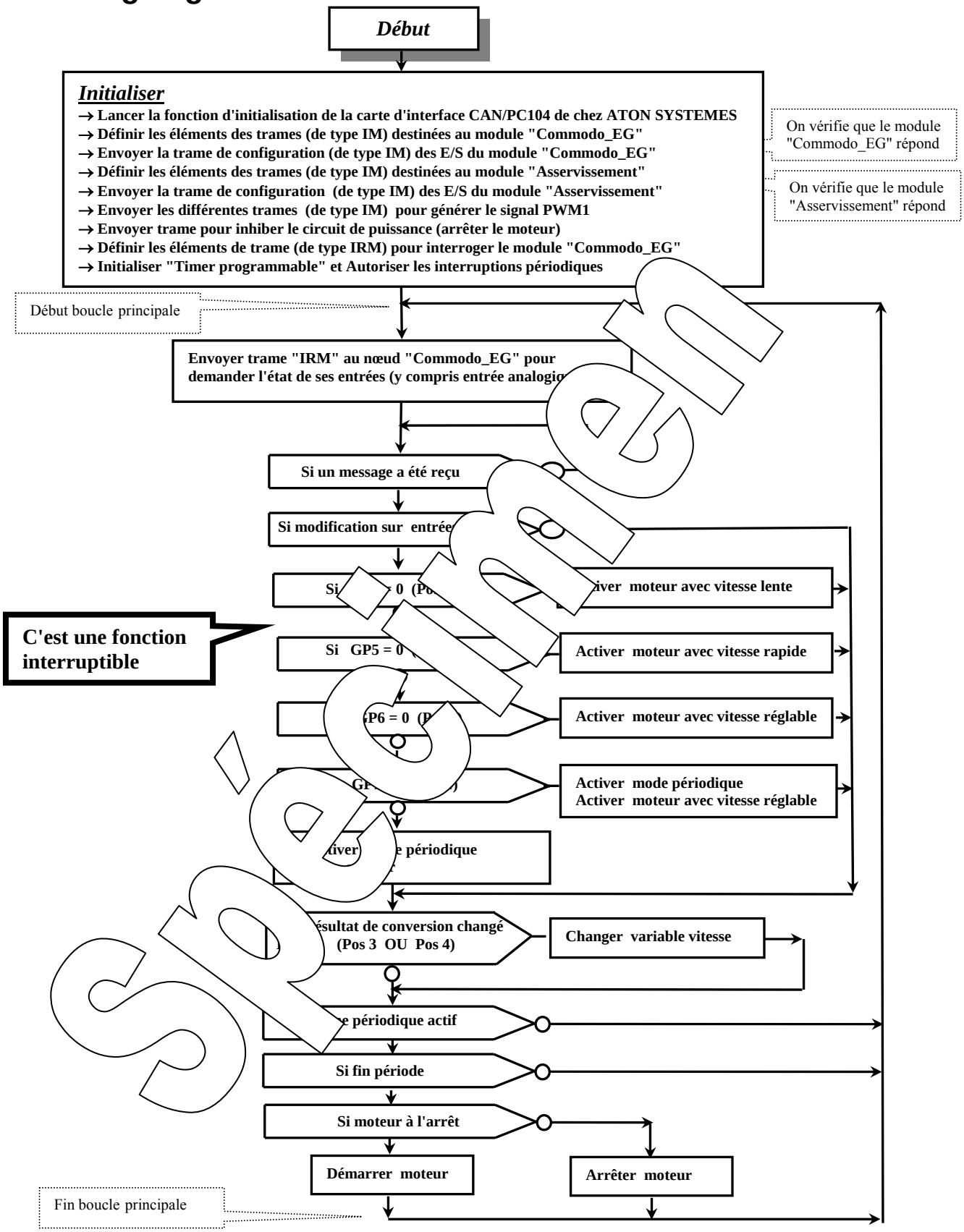
☞ Définition de la fonction d'interruption

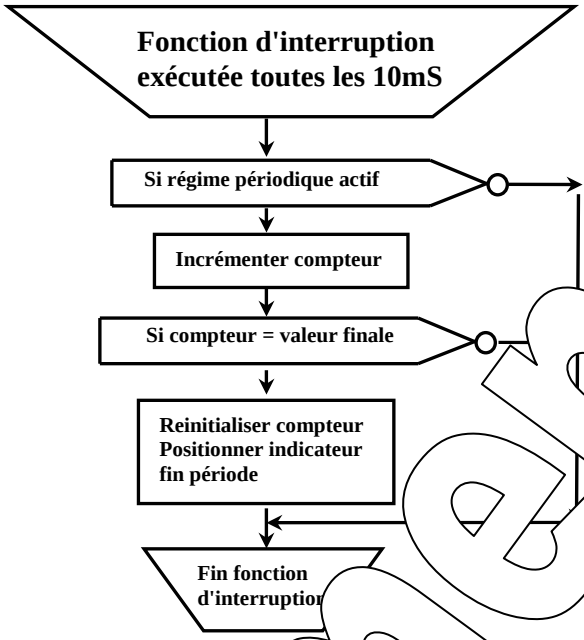
```
/* FONCTION d'interruption pour temporisation  (base de temps:                10 mS)
================================================================                ==*/
void irq_bt()
{
        if (Tempo_En_Cours==1)
              { CPTR_mS++;
                if(CPTR_mS ==100) Reponse_Tempo=1,Tem                0,CP      S=0;
              }
} // Fin de la définition de la fonction d'interruptio      ur      te      risation
```

☞ Définition partielle de la fonction principale

```
/*==============================*/
/*      Fonction principale     */
/*==============================*/
main()
{
/*   Initialisations *
*********************/
// Pour base de temps
SetVect(96,&irq_bt);          // mise en          de l      ovecteur
PITR = 0x0048;                // U          inter          toutes les 10 millisecondes
PICR = 0x0760;                //          60H
// Pour la temporisation
CPTR_mS=0;                    //  Initiali          on compteurs millisecondes
Demande_Tempo=0;
Reponse_Tempo=0;
Tempo_En_Cours=0;
/* Boucle principale *
*********************
       do
       {  // A l'endro          on          aite démarrer la tempo
             Demande_T
             T                Co
          // …                  si          tempo
          if(                se_          1)
                             ponse_          =0;
                             ctio          faire à la fin de la temporisation

                          réin          sation
                          de_Te
             …
             if          ande_Tempo==0)Reponse_Tempo=0,Tempo_En_Cours=0,CPTR_mS=0;

       }wh                /*  Fin de la boucle principale */
} /* Fin de l          nction principale */
```

4.3 Organigramme





4.4 Programme

```
/*  ************************************************************************************************************
        *      TPs sur  EID210 / Réseau CAN - VMD (Véhicule Multiplexé Didactique)
        ************************************************************************************************************
        *      APPLICATION: Commande Essuie-glace
        ************************************************************************************************************
        *            TP n°4:  Séquence commande moteur E.G.
        *-----------------------------------------------------------------------------------------------------------
        *   CAHIER DES CHARGES :

        *  ***********************

        *   On souhaite que l'é                    ende de la position du commodo
        *            - Si Positio                  ur en marche à faible vitesse
        *            - Si Positio                  oteur en marche à grande vitesse
        *            - Si Position                 Moteur en marche à vitesse ajustable par l'entrée analogique
        *            - Si Position 4               marche périodique avec dans l'état marche une vitesse ajustable
        *-----------------------------------------------------------------------------------------------------------
        *  NOM du                                  CAN_VMD_EG.C

        * ******                                   ****

        ************                               ************************************************************************/
/                                                 ers à in
/                                                 ****
#in
#inclu                                            tures_Do                    "
#include "cpu_re                                  .h"
#include "eid21                                   "
#include "Can_V                                   "
#include "Aton_can.h"

// Variables globales
//--------------------
union byte_bits Indicateurs;
#define I_Affiche Indicateurs.bit.b0
#define I_Fin_Tempo Indicateurs.bit.b1
#define I_Active_Tempo Indicateurs.bit.b2
#define I_Etat_Moteur Indicateurs.bit.b3
unsigned int Compteur_Passage_Irq;

//  Fonction d'interruption "Base de Temps"
```

```
//=========================================
void irq_bt()
// Fonction exécutée toute les 10 mS
{if(I_Active_Tempo) // Si mode intermittent actif
        {Compteur_Passage_Irq++;
        if(Compteur_Passage_Irq==400)  // 4  Secondes se sont écoulées
                {Compteur_Passage_Irq=0;
                I_Fin_Tempo=1;
                }
        }
} // Fin de la fonction d'interruption

//============================
//  FONCTION PRINCIPALE
//============================
main()
{// Définition de variables locales
int Compteur_Passage;
unsigned char Commodo_EG_Mem;
unsigned int Cptr_TimeOut;
unsigned char AN0H; // Pour mémoriser les 8 biits de poids forts du résultat de conversion
unsigned char Val_Vitesse,Val_Vitesse_Mem;

Trame Trame_Recue;
Trame T_IRM_Commodo_EG;          // Trame pour interroger Module 8E s                uie C
                                 // IRM -> Information Re        Mess        Trame interrogative
Trame T_IM_Commodo_EG;           // Trame pour commander le Module               odo        uie Glace
                                 // IM -> Information Messa          e de        mande
Trame T_IM_Asservissement;  // Trame pour commander le module "A        issem
                                 // IM -> Inform          e ->          mmande
// Initialisations
//*****************

/* Initialisation DU SJA1000 de la carte PC104 */
Init_Aton_CAN();
// Pour initialiser les liaisons en entrées  noeud "        odo_E
T_IM_Commodo_EG.trame_info.registre=0x00;
T_IM_Commodo_EG.trame_info.champ.extend=1; //        ille en        du
T_IM_Commodo_EG.trame_info.champ.dlc=0x03; // Il        onnées        s (3 octets envoyés)
T_IM_Commodo_EG.ident.extend.identificateur.ident=Iden        Commo        _EG;
T_IM_Commodo_EG.data[0]=0x1F;       // p        nnée -        e" du registre concernée
                                                //(        R donne la direction des I/O) adresse = 1Fh  page 16
T_IM_Commodo_EG.data[1]=0x7F; // deux        onnée -> "M        que"
                                                //->        s bits concernés sauf GP0 (voir doc page 16)
T_IM_Commodo_EG.data[2]=0x7F; // troisie        ée -> "        " -> Tous les bits en entrée
// Envoi trame pour définir        direction des en
Ecrire_Trame(T_IM_Commo        _E        er t        u CAN
        Cptr_TimeOut=0;
        do{Cptr_TimeOut++        le((I        ram        &Trame_Recue)==0)&&(Cptr_TimeOut<2000));
        if(Cptr_TimeOut==2
                { clsscr();
                  g        (2,12);
                  a rep        ame de commande en initialisation \n");
                  "Vou  remettre   alimentation 12 V  \n");
                  while(        me(&Trame_Recue)==0); // On attend les trames  "On Bus"
                  (Cptr_T        Cptr_        t<200000;Cptr_TimeOut++);
                  Trame(        G);  // Renvoyer trame IM sur réseau CAN
                  Cptr        ;
                  (Cptr        ++;}while((Lire_Trame(&Trame_Recue)==0)&&(Cptr_TimeOut<2000));
                  r_Tim        00)
                    { goto        );
                    printf(" Pas de reponse a la trame de commande en initialisation \n");
                    printf("  Modifier le programme et recommencer  \n");
                  hile(1); // On reste bloqué, on ne continue pas
                    }
                }
// Pour activer les conversions Ana -> Num
T_IM_Commodo_EG.data[0]=0x2A;        // Adresse du registre ADCON0
T_IM_Commodo_EG.data[1]=0xF0;        // Masque -> bits 7..4 concernés
T_IM_Commodo_EG.data[2]=0x80;        // Valeur ->  ADON=1 et "prescaler rate"=1:32
Ecrire_Trame(T_IM_Commodo_EG);
do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre réponse
```

```
// Pour définir le mode de conversion
T_IM_Commodo_EG.data[0]=0x2B;          // Adresse du registre ADCON1
T_IM_Commodo_EG.data[1]=0xFF;          // Masque -> tous les bits sont concernés
T_IM_Commodo_EG.data[2]=0x0C;          // Valeur ->  voir doc MCP25050 page 36
Ecrire_Trame(T_IM_Commodo_EG);
do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre réponse
// Pour initialiser le module "Asservissement"
// Trame de type "IM" (trame de commande): Données d'identification pour le noeud "Asservissement"
T_IM_Asservissement.trame_info.registre=0x00;
T_IM_Asservissement.trame_info.champ.extend=1;
T_IM_Asservissement.trame_info.champ.dlc=0x03;
T_IM_Asservissement.trame_info.champ.rtr=0;
T_IM_Asservissement.ident.extend.identificateur.ident=Ident_T_IM_Asservissement;
// Pour définir des Entrées/Sorties
T_IM_Asservissement.data[0]=0x1F;          // Adresse du registre GPDDR  (direction de E/S)
T_IM_Asservissement.data[1]=0xEF;          // Masque -> Bit 7 non concerné
T_IM_Asservissement.data[2]=0xE3;          // Valeur ->  1 si Entrée et 0 si Sortie
            // GP7=fs Entrée; GP6=fcg Entree; GP5=fcd Entrée;  GP4=ValidIP Sortie;
            // GP3=PWM2 Sortie; GP2=PWM1 Sortie; GP1=AN1 Entrée; GP0=AN0 Entrée;
Ecrire_Trame(T_IM_Asservissement);
do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre trame réponse (acquitement)
// Pour mettre à 0 les sorties
T_IM_Asservissement.data[0]=0x1E;          // Adresse du registre GPLAT  (Registre I/O)
T_IM_Asservissement.data[1]=0x1C;          // Masque -> sorties GP4,3,2 sont conse
T_IM_Asservissement.data[2]=0x00;          // Valeur ->  les 3 sorties sont forcée
Ecrire_Trame(T_IM_Asservissement);
do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre réponse (acquitement)
// Pour définir sortie GP2 en PWM1
T_IM_Asservissement.data[0]=0x21;          // Adresse du registre T1CON
T_IM_Asservissement.data[1]=0xB3;          // Masque -> seuls bit 7;5;4                nés
T_IM_Asservissement.data[2]=0x80;          // Valeur ->  TMR1ON=1;
Ecrire_Trame(T_IM_Asservissement);
do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre répons                nt)
// Pour définir fréquence signal sortie PWM1
T_IM_Asservissement.data[0]=0x23;          // Adresse du reg
T_IM_Asservissement.data[1]=0xFF;          // M          > tou          ont con
T_IM_Asservissement.data[2]=0xFF;          // V          -> PR1=25
Ecrire_Trame(T_IM_Asservissement);
do{}while(Lire_Trame(&Trame_Recue)==0); // Attend                (acqui

// Pour trame interrogative envoyée au commod                Glace                (Information Request Message)
// Définir données d'identification
T_IRM_Commodo_EG.trame_info.registre=
T_IRM_Commodo_EG.trame_info.champ.e                1;
T_IRM_Commodo_EG.tra          info.champ.dl          // On a          octets en réponse
T_IRM_Commodo_EG.tra          info.champ.rtr=1
T_IRM_Commodo_EG.ident          ten          teur.                _IRM8_Commodo_EG;
                                                                                    // Voir définitions dans fichier CAN_VMD.h
// Une première acquisition pour          liser          nde          rs mise en mémoire
Ecrire_Trame(T_IRM_Commod          //          envo          première trame interrogative pour initialiser la mémoire
do{}while(Lire_Trame(&Trame_R                );
Commodo_EG.valeur=~(Trame_Rec                ;
Commodo_EG_Mem=                G.va          met en mémoire l'état actuel pour comparaison ultéérieure
Val_Vitesse=Trame_R                n récupère les MSB de la conversion A->N  voie AN0
Val_Vitesse_Mem          tesse; //          t en mémoire la valeur pour comparaison ultérieure
// Pour          indicateu
Ind          eur=0; //          ont positionnés à 0

          initialis          t bie
          TP
//c
//goto
printf("    ********          ********************************************** \n");
printf("     TPs          N    Application: Commande Essui-glace      \n");
printf("     ----------          ---------------------------------------- \n");
printf("     TP n°4       Sequence de commande  moteur E.G.            \n");
printf("     La vitesse moteur depend de la position commodo            \n");
printf("          Pos1 (BP1): Petite vitesse                    \n");
printf("          Pos2 (BP2): Grande Vitesse                    \n");
printf("          Pos3 (BP3): Vitesse Ajustable                                    \n");
printf("          Pos4 (BP4): Fonctionnement périodique                          \n");
printf("    ********************************************************** \n");

// Pour initialiser la base de temps et temporisations
SetVect(96,&irq_bt); // mise en place de l'autovecteur
PITR = 0x0048;                    // Une interruption toutes les 10 millisecondes
PICR = 0x0760;                    //  96 = 60H
```

```
// Boucle principale
//*******************
  do
        {  Ecrire_Trame(T_IRM_Commodo_EG); // On envoi trame interrogative au noeud commodo Essuie-Glace
     // On attend la réponse
          do{}while(Lire_Trame(&Trame_Recue)==0);  // La fonction renvoie 1 dans ce cas
              if (Trame_Recue.ident.extend.identificateur.ident==Ident_T_IRM8_Commodo_EG)
                    {
                    Commodo_EG.valeur=~(Trame_Recue.data[1]&0xF0); // On récupére les états des entrées binaires
                    if(Commodo_EG.valeur!=Commodo_EG_Mem)
                      {// S'il y a eu changement d'une des entrées binaires, on commande le moteur
                            if(Commodo_EG.bit.GP4==1) // On test GP4 (BP1) du commodo EG
                       { // BP1 est actionné  (GP4=0)
                            I_Active_Tempo=0;
                            //gotoxy(2,12);
                        //printf("   Action sur BP1 (GP4)  \n" );
                            T_IM_Asservissement.data[0]=0x1E;       // Adresse du registre GPLAT  (Registre I/O)
                            T_IM_Asservissement.data[1]=0x10;       // Masq                le la sortie ValidIP sera consernée
                            T_IM_Asservissement.data[2]=0x10; // ValidIP =                r en marche
                            Ecrire_Trame(T_IM_Asservissement);
                            do{}while(Lire_Trame(&Trame_Recue)==0                dre rép
                           // Pour modifier le rapport cyclique PWM1 -> c                dule de l             mande
                            T_IM_Asservissement.data[0]=0x25;       // Adres                tre PW   1DC
                            T_IM_Asservissement.data[1]=0xFF;                que -                bits sont consernés
                            T_IM_Asservissement.data[2]=40;                  PW                Vitesse lente
                            Ecrire_Trame(T_IM_Asservisseme           // Le r                ne peut tourner car ValidIP=0
                            do{}while(Lire_Trame(&Trame_R                ==0                tte                réponse
                           }
                      //else {
                            if(Commodo_EG.bit.GP5=                t GP                ommodo EG
                       { // BP2 est actionné  (GP5=0)
                            I_Active_Tempo=0;
                            //gotoxy(2,12);
                        //printf("   Action sur BP2                
                            T_IM_Asservis                ata[0                // Adresse du registre GPLAT  (Registre I/O)
                            T                servis                a[1]=0x                // Masque -> seule la sortie ValidIP sera consernée
                            T_I                Ass                isseme                =0x10;                ValidIP =1 donc moteur en marche
                            Ecrire_T                IM_A                ent);
                            do{}while(                e(&T                cue)==0); // Attendre réponse
                           // Pour modifier                t cyclique                PWM1 -> c'est le module de la commande
                            T                visseme                0]=0x25;       // Adresse du registre PWM1DC
                                        ervissement.                =0xFF;       // Masque -> tous les bits sont consernés
                                        _Asservissement.data[2]=70;       // Valeur ->PWM1DC   Vitesse rapide
                                        Trame(T                Asservissement);  // Le moteur ne peut tourner car ValidIP=0
                                        do{}while(Lire_T                (&Trame_Recue)==0); // Attendre réponse
                           }
                                        e {
                                        mmodo_EG.bit.GP6==1) // On test GP6 (BP3) du commodo EG
                            // BP                actionné  (GP6=0)
                                        _Ac                Tempo=0;
                                        //got                12);
                                                sur BP3 (GP6)\n" );
                                        _Asservissement.data[0]=0x1E;       // Adresse du registre GPLAT  (Registre I/O)
                            T_IM_Asservissement.data[1]=0x10;       // Masque -> seule la sortie ValidIP sera consernée
                            T_IM_Asservissement.data[2]=0x10; // ValidIP =1 donc moteur en marche
                            Ecrire_Trame(T_IM_Asservissement);
                            do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre réponse
                           // Pour modifier le rapport cyclique PWM1 -> c'est le module de la commande
                            T_IM_Asservissement.data[0]=0x25;       // Adresse du registre PWM1DC
                            T_IM_Asservissement.data[1]=0xFF;       // Masque -> tous les bits sont consernés
                                        _IM_Asservissement.data[2]=Val_Vitesse; // Valeur ->PWM1DC   Vitesse réglable
                            Ecrire_Trame(T_IM_Asservissement);  // Le moteur ne peut tourner car ValidIP=0
                            do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre réponse
                           }
                      //else {
                            if(Commodo_EG.bit.GP7==1) // On test GP6 (BP3) du commodo EG
                       { // BP3 est actionné  (GP6=0)
                            I_Active_Tempo=1;
                            //gotoxy(2,12);
                        //printf("   action sur BP4 (GP7)\n" );
                            T_IM_Asservissement.data[0]=0x1E;       // Adresse du registre GPLAT  (Registre I/O)
                            T_IM_Asservissement.data[1]=0x10;       // Masque -> seule la sortie ValidIP sera consernée
                            T_IM_Asservissement.data[2]=0x10; // ValidIP =1 donc moteur en marche
                            Ecrire_Trame(T_IM_Asservissement);
                            do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre réponse
                           // Pour modifier le rapport cyclique PWM1 -> c'est le module de la commande
                            T_IM_Asservissement.data[0]=0x25;       // Adresse du registre PWM1DC
```

```
                                T_IM_Asservissement.data[1]=0xFF;       // Masque -> tous les bits sont consernés
                                T_IM_Asservissement.data[2]=Val_Vitesse; // Valeur ->PWM1DC   Vitesse réglable
                                Ecrire_Trame(T_IM_Asservissement);  // Le moteur ne peut tourner car ValidIP=0
                                do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre réponse
                               }
                               //else  {
                               // ni BP1 ni BP2 ni BP3 ni BP4 actionné  ->  On arrete donc le moteur
                               if(Commodo_EG.valeur==0x0F)
                                {I_Active_Tempo=0;
                                 T_IM_Asservissement.data[0]=0x1E;       // Adresse du registre GPLAT  (Registre I/O)
                                 T_IM_Asservissement.data[1]=0x10;       // Masque -> seule la sortie ValidIP sera consernée
                                 T_IM_Asservissement.data[2]=0x00;       // ValidIP =0 donc moteur à l'arret
                                 Ecrire_Trame(T_IM_Asservissement);
                                 do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre réponse
                                 //gotoxy(2,12);
                                 //printf("   Moteur a l'arret         \n");
                                 }
                               //gotoxy(2,14);
                          printf("Bh=%2.2x\n",Commodo_EG.valeur);  // On affich
                    }// Fin si changement d'une des entrées binaires
               // Pour l'entrée analogique
                         Val_Vitesse=Trame_Recue.data[2]; // On récupére les MS                nversion                voie AN0
                         if(Val_Vitesse!=Val_Vitesse_Mem)
                           {          //gotoxy(2,16);
                              printf("Ad=%3d\n",Val_Vitesse);  // On aff
                                   if((Commodo_EG.bit.GP6==1)||(Co                 o_Ec                 P7==1))
                                   {// Pour modifier le rapport cyclique                 l                 t le                 ule de la commande
                                    T_IM_Asservissement.data[0]=0x25;                 resse                 gistre PWM1DC
                                    T_IM_Asservissement.data[1]=0xFF;                 ...                 us les bits sont consernés
                                    T_IM_Asservissement.dat                 Vitess                 Valeur ->PWM1DC
                                    Ecrire_Trame(T_IM_Asse                 Le moteur ne peut tourner car ValidIP=0
                                    do{}while(Lire_Tram                 e_Re                 // Attendre réponse
                                   }
                           }
               // Mise en mémoire des valeurs acquise
                         Commodo_EG                 = Co                 G.valeu                 et en mémoire l'état des entrées binaires
                         Val_Vitesse_M                 Val_Vitesse                 // On                 en mémoire le résultat de conversion A->N
                         } // Fin si identificate                 t
               // Pour mode périodique
               if(I_Active_Tempo==1)
                         {if(I_Fin_Tem
                                   {                 po=0;
                                             at_Moteur=                 )
                                                 {I_E                 eur=0;
                                    T_                 sserviss                 .data[0]=0x1E;       // Adresse du registre GPLAT  (Registre I/O)
                                    T_                 .data[1]=0x10;       // Masque -> seule la sortie ValidIP sera consernée
                                    T_IM_                 nt.data[2]=0x00;       // ValidIP =0 donc moteur à l'arret
                                    Ecrire_Trame(T_IM_Asservissement);
                                    do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre réponse

                                    else
                                                 {I_Etat_Moteur=1;
                                    T_IM_Asservissement.data[0]=0x1E;       // Adresse du registre GPLAT  (Registre I/O)
                                    T_IM_Asservissement.data[1]=0x10;       // Masque -> seule la sortie ValidIP sera consernée
                                    T_IM_Asservissement.data[2]=0x10;       // ValidIP =1 donc moteur en marche
                                    Ecrire_Trame(T_IM_Asservissement);
                                    do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre réponse
                                                 }
                                   }
                         }
                                   la bouc                 pale
}// Fi                 principa
```

PAge laissée vierge

5 TP N°5: REGULATION DE VITESSE MOTEUR ESSUIE GLACE

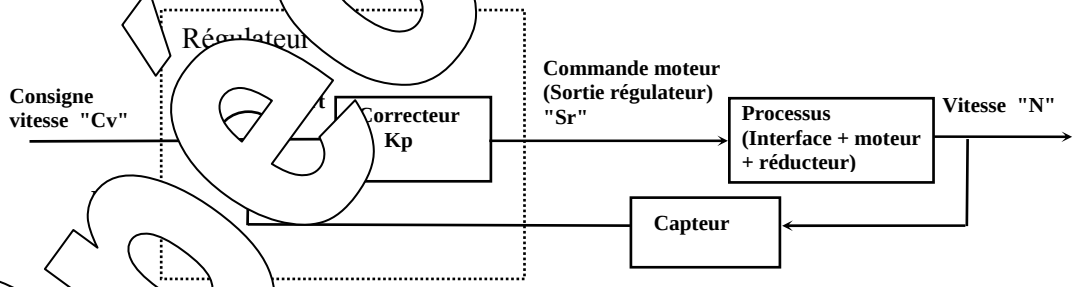
5.1 Sujet

Objectifs :	- Acquérir le résultat d'une convertion Analogique -> Numérique via un réseau CAN. - Réaliser un échantillonnage avec période imposée. - Expérimenter le mode de commande en « boucle fermée » d'un système analogique pilotable par réseau CAN. - Programmer un correcteur numérique (action proportionnelle, action intégrale)
Cahier des charges :	Commander le moteur "essuie gla[illegible] boucle fermée avec correcteur à action proportionnelle. La consigne de vitesse est donnée par l'[illegible] analogique du module « Commodo Essuie-glace » La mesure vitesse est le résultat co[illegible]rsion entrée AN1 -> sortie F/U sur le module « Asservissement »

5.2 Eléments de solution

5.2.1 Principe de la comma[illegible]de en boucle fermée

Dans le cas d'une régulation en vitesse du moteur [illegible]de proportionnel, la grandeur de commande est proportionnelle à la différence (Consigne vitesse - M[illegible]vitesse).
Pour le programme, la consigne vitesse sera [illegible]ltat de conversion de la tension appliquée sur l'entrée analogique AN0 (GP0) du module « Commodo [illegible]Glace » et la mesure vitesse, le résultat de conversion de la sortie du convertisseur F/U [illegible]iquée sur l'entrée AN1 (GP1). Ce sera une régulation numérique, donc échantillonnée.



D[illegible]as d'une régulation par action proportionnelle, Sr aura pour expression: Sr = Kp.(Cv-Mv)
[illegible]alcul[illegible]inter[illegible]e temps régulier appelés "période d'échantillonnage" et notée "Te".
L[illegible]consi[illegible]ans le programme comme un entier, mais réalité ses 4bits de poids faibles repres[illegible]ont la pa[illegible]fractionnaire: Kp=0x10 -> valeur =1 ; Kp=20h -> valeur =2; etc..
Kp=0x08 -> [illegible],5; Kp=0x04 -> valeur =0,25; Kp=0x02 -> valeur =0,125; etc…
En définitive [illegible]a compris dans l'intervalle: $15{,}9375 \leq Kp \leq 0$

5.2.2 Acquisition des entrées "Commodo Essuie Glace" pour la consigne

La technique d'acquisition des entrées binaires et de l'entrée analogique, par réseau CAN, du module "Commodo E.G" est décrite dans le TP n°1.

5.2.3 Acquisition des entrées du module "Asservissement" pour la mesure vitesse

Une solution est d'utiliser l'IRM "Read A/D Regs, ce qui permet d'acquérir à la fois les états des entrées logique (fins de courses) et les résultats de conversion des entrées analogiques (`doc MCP25050 p22).`

→ Définition de variables structurées sous le modèle "**`Trame`**":

`Trame T_IRM_Asservissement;` // Trame destinée à l'interrogation du module asservissement pour acquérir fins de courses ainsi que les résultats de conversion A->N.

Remarque: La variable structurée **`T_IRM_ Asservissement`** comportera 5 octets utiles seulement, 1octet pour trame_info et 4 octets pour l'identificateur en mode étendu

→ Accès et définition des différents éléments de la variable structurée " **`T_IRM_Acquerir_FC_AN`** "

```
T_IRM_Asservissement.trame_info.registre=0x00;  //On initialise tous les bits à 0
T_IRM_Asservissement.trame _info.champ.extend=1; //On travaille en mode étendu
T_IRM_ Asservissement.trame _info.champ.dlc=0x08; //Il y a 8 octets de données demandés
T_IRM_ Asservissement.ident.extend.identificateur.ident = 0x10000;
```

La trame réponse, suite à l'IRM, comportera en données associées 8 octets (`doc MCP25050 p22`):

- octet de rang 0 (data[0])→ valeur IOINTFL non utilisé dans notre cas
- octet de rang 1 (data[1])→ valeur GPIO → Valeur des entrées logiques
- octet de rang 2 (data[2])→ valeur AN0H → 8 bits MSB conversion entrée anologique 0
- octet de rang 3 (data[3])→ valeur AN1H → 8 bits MSB conversion entrée anologique 1
- octet de rang 4 (data[4])→ valeur AN10H → 2 fois 2 bits LSB conversion entrées ana. 1 et 0

Les 3 autres octets ne sont pas utiles dans notre application.

Le résultat de conversion est sur 10bits:

- pour résultat AN0 (potentiométre)

d9 d8 … d4 d3 d2	d1 d0 - - - - -
…0H ->…[2]	**AN10L -> data[4]**

- pour résultat AN1(capteur vi…)

d… d7 d6 … d4 d3 d2	- - - - d1 d0 - -
…H ->data[3]	**AN10L -> data[4]**

5.2.4 Réalisation de pér… d'éch…tillonnage:

Il est possible d'utili… la capacit… …ircuit … P25050 à envoyer spontanément et à intervalles de temps réguliers, un trame "…ad A/D Regs… …dans les "Data" les résultats de conversion (doc MCP25050 page 22).

Il faut pour cela initialis… "Scheduled transmission" (doc MCP25050 page 24).

Il faut pour cela initialis… par … trames de type "IM", les rgistre "STCON" et "IOINTEN".

→ *Pour définir la …iode … …nage (fréquence d'envoi des trames par le"'séquenceur")*

Cette fréquence … la va… …hargée dans le registre "STON"

```
T_IM_Asser…emen…a[0]=0x2C; // adresse du registre STON en écriture
 (doc MCP…  p15)  … décalage = 10H + 1CH = 2CH
T_IM_…serv…nt…[1]=0xFF; // Masque: tous les bits sont concernés
T_IM_…sservisse…a[2]=0xD2; // Valeur: (voir doc MCP25050 page 24).
   b7…N = …pour activer le séquenseur
   …= 1 …r trames à 8 octets (contenant les résultats de conversion)
   …4 = …-> pé…e de base = 16.4096.Tosc
   b3..b0 = …0 -> multiplieur de période = 3
```

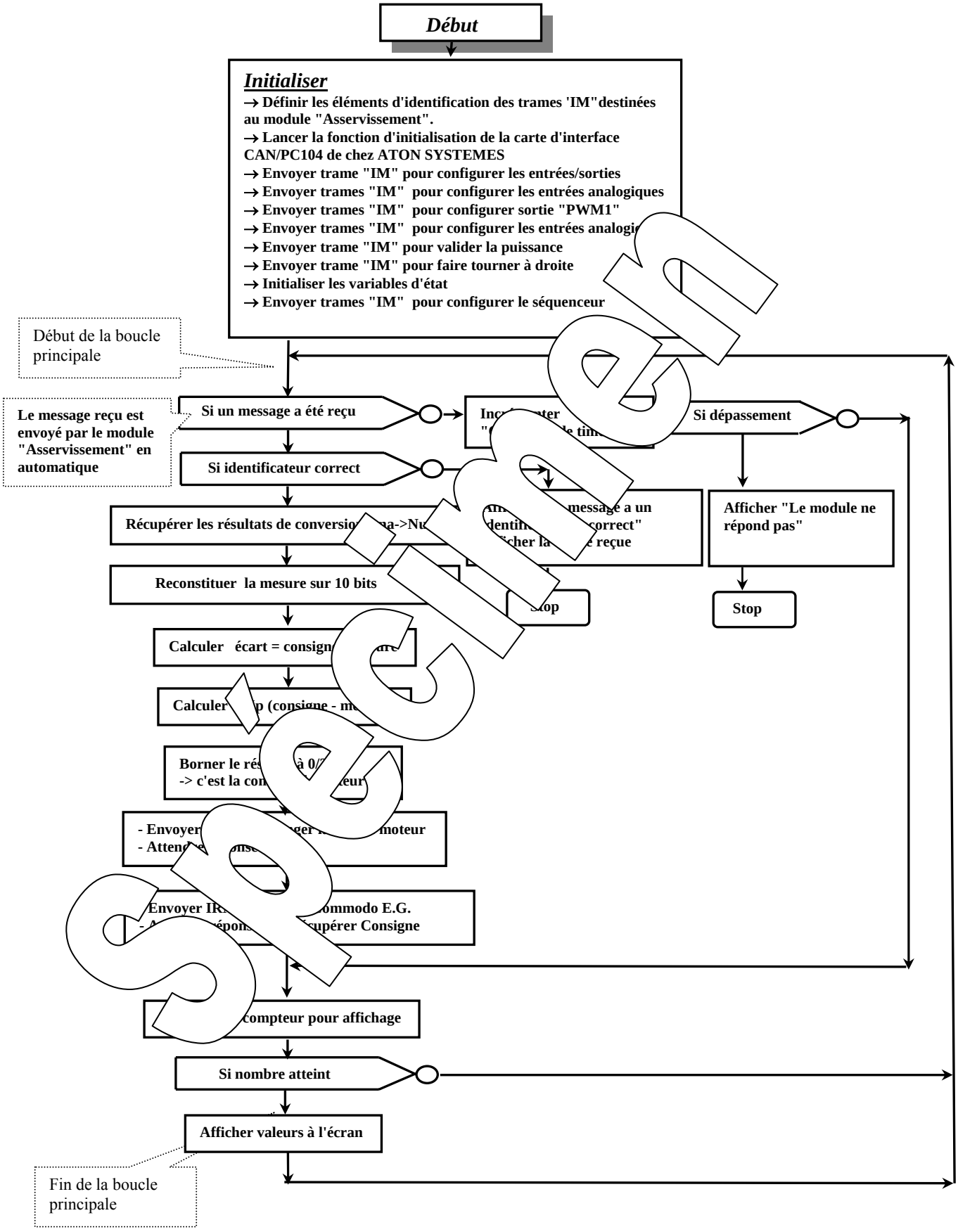
La fréquen…z implanté sur la carte "asservissement" étant égale à 16Mhz (Tosc = $1/16.10^6$), la période d'env… trames sera égale à $16.4096.3/16.10^6 = 12$ mS.

→ *Pour activer l'auto-conversion des convertisseurs Ana -> Num*

IL faut initialiser le registre "IOINTEN" et notamment les deux bits correspondant aux deux entrées analogiques utilisées dans cette application.

```
T_IM_Asservissement.data[0]=0x1C; // adresse du registre IOINTEN en écriture
 (doc MCP25050 p15) 00H + décalage = 00H + 1CH = 1CH
T_IM_Asservissement.data[1]=0x03; // Masque: seuls les bits 0 et 1 sont concernés
T_IM_Asservissement.data[2]=0x03; // Valeur: (voir doc MCP25050 page 27).
```

5.3 Organigramme



5.4 Programme

```
/*****************************************************************************************************
*      TPs sur  EID210 / Réseau CAN - VMD  (Véhicule Multiplexé Didactique)
*****************************************************************************************************
*      APPLICATION: Commande Essuie-glace
*****************************************************************************************************
*   TP n°5:  Réguler la vitesse du balai d'essuie-glace avec correcteur P
*---------------------------------------------------------------------------------------------------
*   CAHIER DES CHARGES :
*  *********************
*    Réguler la vitesse du moteur EG avec pour consigne, l'entrée analogique
*          de rang 0  du module "Commodo-EG"
*                              La commande du moteur se fait boucle fermé avec correcteur à action proportionnelle
*                                  Commande = Kp*Ecart = Kp * (Consigne - Mesure)
*---------------------------------------------------------------------------------------------------
*  NOM du FICHIER :  TP_5_CAN_CMD_EG.C
* ******************
*****************************************************************************************************/

// Déclaration des fichiers d'inclusion
#include <stdio.h>
#include"Structures_Donnees.h"
#include"cpu_reg.h"
#include "eid210_reg.h"
#include "CAN_vmd.h"
#include "Aton_can.h"

// Déclarations des diverses trames de communication
Trame Trame_Recue;// Pour la trame qui vient d'etre reçue par le contr
// Trmes de type "IM" (Input Message -> trame de commande)
Trame T_IM_Asservissement;    // Pour la commande du moteur
Trame T_IM_Commodo_EG;
Trame T_IRM_Commodo_EG;  // Pour l'acquisition des entr        modo
// Déclaration des variables
// Pour les Indicateurs divers (variables binaires)
union byte_bits Indicateurs; // Structures de bits
#define I_Sens_Rotation Indicateurs.bit.b0
#define I_Attente_Reponse_IRM Indicateurs.bit.b1
#define I_Message_Pb_Affiche Indicateurs.bit.b
// Pour les résultats de conversion
unsigned short S_Mesure_Vitesse,S_Consig        emp;
unsigned char AN0H,AN1H,AN10L;
// Pour régulateur de vitesse
int Ecart,Resultat_Calcul;
unsigned char Cde_Moteur;
// Déclaration constante pour rég
#define Kp 6              // Coeffici      ction      ortie    nelle -> Ce doit etre un entier
                          //i          r ré       p/16 = 6/16


//=================
// FONCTION PRINC
//=================
main()
{
// I          XTIONS
//-----
/      aration          local          ction principale
//
un          rtr_Tim          pr_Acquisition;
// Initi          arte contr          éseau CAN
Init_Aton_CAN(
// Effacer l'écra
clsscr();
// Pour initialiser les liaison en entrées du noeud "Commodo Essuie-Glace"
T_IM_Commodo_EG.trame_info.registre=0x00;
T_IM_Commodo_EG.trame_info.champ.extend=1; // On travaille en mode étendu
T_IM_Commodo_EG.trame_info.champ.dlc=0x03; // Il y aura 3 données de 8 bits (3 octets envoyés)
T_IM_Commodo_EG.ident.extend.identificateur.ident=Ident_T_IM_Commodo_EG;
T_IM_Commodo_EG.data[0]=0x1F;          // première donnée -> "Adresse" du registre concernée
                                                    //(GPDDR donne la direction des I/O) adresse = 1Fh  page 16
T_IM_Commodo_EG.data[1]=0x7F; // deuxième donnée -> "Masque"
                                                  //-> Tous les bits concernés sauf GP0 (voir doc page 16)
T_IM_Commodo_EG.data[2]=0x7F;          // Valeur ->  1 si Entrée et 0 si Sortie  (Toutes en entrées)
```

```
// Pour envoyer trame et test si réponse  (avec Time Out
I_Message_Pb_Affiche=0;
do {Ecrire_Trame(T_IM_Commodo_EG);  // C'est la première trame envoyée
            Cptr_TimeOut=0;                                          // On teste si le module répond bien
            do{Cptr_TimeOut++;}while((Lire_Trame(&Trame_Recue)==0)&&(Cptr_TimeOut<500));
            if(Cptr_TimeOut==500)
                        {if(I_Message_Pb_Affiche==0)
                                    {I_Message_Pb_Affiche=1;
                                    gotoxy(2,10);
                                    printf(" Pas de reponse a la trame de commande en initialisation \n");
                                    printf("  Verifier si alimentation 12 V  est OK   ...  PUIS   \n");
                                    printf("  FAIRE un Reset de la carte processeur EID 210  ...  PUIS  \n");
                                    printf("  RECHARGER le programme et RELANCER \n");
                                    do{}while(1);}}
   }while(Cptr_TimeOut==500);
// La première tramme est bien passée ... on peut continuer!
clsscr();
// Pour afficher titre
gotoxy(1,2);
printf("    ***************************************************************************\n");
printf("     TPs sur Reseau CAN    Application: Commande Essui-Glace            \n");
printf("    ------------------------------------------------------------ \n");
printf("     TP n°5:        REGULER LA VITESSE DU BALAI D'ESSUIE GLACE\n");
printf("    ************************************************************\n");
printf("      - en agissant sur l'entree analogique 0 sur module Commodo_EG\n");
printf("      - en boucle fermee en mode proportionnel Sr = Kp(Consigne -Mesure\n");
printf("    ************************************************************* \n");


// Pour activer les conversions Ana -> Num
T_IM_Commodo_EG.data[0]=0x2A;          // Adresse du registre A
T_IM_Commodo_EG.data[1]=0xF0;          // Masque -> bits 7..4
T_IM_Commodo_EG.data[2]=0x80;          // Valeur ->  ADON=
Ecrire_Trame(T_IM_Commodo_EG);
do{}while(Lire_Trame(&Trame_Recue)==0); //
// Pour définir le mode de conversion
T_IM_Commodo_EG.data[0]=0x2B;          // Adresse
T_IM_Commodo_EG.data[1]=0xFF;          // Masque
T_IM_Commodo_EG.data[2]=0x0C;          // Valeur ->
Ecrire_Trame(T_IM_Commodo_EG);
do{}while(Lire_Trame(&Trame_Recue)==0)
// Trame de type "IRM" (trame interrogative
T_IRM_Commodo_EG.trame_info.registre=
T_IRM_Commodo_EG.tra info.champ.ex
T_IRM_Commodo_EG.tra info.champ.dlc=
T_IRM_Commodo_EG.tram nfo
T_IRM_Commodo_EG.ident.ext

// Initialisation des différentes tra
// Trame de type "IM" (trame de c
T_IM_Asservissement.tr info.re
T_IM_Asservissement amp
T_IM_Asservissemen .dlc=0x03;
T_IM_Asservisse e_info. r=0;
T_IM_Asservissem xtend.i ateur.ident=Ident_T_IM_Asservissement;
// P des Entr
T issement.data[ // Adresse du registre GPDDR  (direction de E/S)
                                                // doc MCP25050 Page 16
                ]=0x Masque -> Bit 7 non concerné
T_ =0xE3;  aleur ->  1 si Entrée et 0 si Sortie
          =fs Enti 6=fcg Enti ; GP5=fcd Entrée;  GP4=ValidIP Sortie;
          // GP3 PWM e; GP2=PWM1 Sortie; GP1=AN1 Entrée; GP0=AN0 Entrée;
Ecrire_Trame(T sement);
do{}while(Lire_ rame_Recue)==0); // Attendre réponse
// Pour mettre à 0 les sorties
T_IM_Asservissement.data[0]=0x1E;          // Adresse du registre GPLAT  (Registre I/O)
T_IM_Asservissement.data[1]=0x1C;          // Masque -> sorties GP4,3,2 sont consernées
T_IM_Asservissement.data[2]=0x00;          // Valeur ->  les 3 sorties à 0
Ecrire_Trame(T_IM_Asservissement);
do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre réponse
// Pour définir sortie GP2 en PWM1
T_IM_Asservissement.data[0]=0x21;          // Adresse du registre T1CON
T_IM_Asservissement.data[1]=0xB3;          // Masque -> seuls bit 7;5;4;1;0 consernés
T_IM_Asservissement.data[2]=0x80;          // Valeur ->  TMR1ON=1; Prescaler1=1
Ecrire_Trame(T_IM_Asservissement);
do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre réponse
```

```
// Pour définir fréquence signal sortie PWM1
T_IM_Asservissement.data[0]=0x23;        // Adresse du registre PR1
T_IM_Asservissement.data[1]=0xFF;        // Masque -> tous les bits sont consernés
T_IM_Asservissement.data[2]=0xFF;        // Valeur ->  PR1=255
Ecrire_Trame(T_IM_Asservissement);
do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre réponse
// Pour Valider le circuit de puissance
T_IM_Asservissement.data[0]=0x1E;        // Adresse du registre GPLAT  (Registre I/O)
T_IM_Asservissement.data[1]=0x10;        // Masque -> sortie GP4 (ValidIP) est conserné
T_IM_Asservissement.data[2]=0x10;        // Valeur ->
Ecrire_Trame(T_IM_Asservissement);
do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre réponse
// Pour initialiser le rapport cyclique  (valeur initiale)
T_IM_Asservissement.data[0]=0x25;        // Adresse du registre PWM1DC (Charger cde vitesse)
T_IM_Asservissement.data[1]=0xFF;        // Masque -> tous les bits sont concernés
T_IM_Asservissement.data[2]=0;           // Valeur ->  Commande vitesse à 0
Ecrire_Trame(T_IM_Asservissement);
do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre réponse
// Pour activer les conversions Ana -> Num
T_IM_Asservissement.data[0]=0x2A;        // Adresse du registre ADCON0
T_IM_Asservissement.data[1]=0xF0;        // Masque -> bits 7..4 concernés
T_IM_Asservissement.data[2]=0x80;        // Valeur ->  ADON=1 et "prescaler rate"=1:32
Ecrire_Trame(T_IM_Asservissement);
do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre réponse
// Pour définir le mode de conversion
T_IM_Asservissement.data[0]=0x2B;        // Adresse du registre ADCON1
T_IM_Asservissement.data[1]=0xFF;        // Masque -> tous les bits sont conce
T_IM_Asservissement.data[2]=0x0C;        // Valeur ->  voir doc MCP25050 pag
Ecrire_Trame(T_IM_Asservissement);
do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre réponse
// Pour valider le séquenceur
T_IM_Asservissement.data[0]=0x2C;        // Adresse du registre S
T_IM_Asservissement.data[1]=0xFF;        // Masque -> tous les          cernés
T_IM_Asservissement.data[2]=0xD2;        // Valeur ->  voir d          e 24
Ecrire_Trame(T_IM_Asservissement);
do{}while(Lire_Trame(&Trame_Recue)==0); //          dre rép
// Pour valider le séquencement sur entrées analo          és ( t 1
T_IM_Asservissement.data[0]=0x2C;        // Adresse          tre ST
T_IM_Asservissement.data[1]=0x03;        // Masque -          its son          dés
T_IM_Asservissement.data[2]=0x03;        // Valeur ->  von          P25050          age 24
Ecrire_Trame(T_IM_Asservissement);
do{}while(Lire_Trame(&Trame_Recue)==0)          e réponse
Cptr_Affichage=0;
Ctpr_Acquisition=0;
Cptr_TimeOut=0;
// BOUCLE PRINCIPALE
//*******************
while(1)
        {// Un trame donnant          ats es          e à          tervalles de temps réguliers
         // Fonction 'Scéduleu          Ass          ement' est activée
        if(Lire_Trame(&Trame          1) //          me résultat n'est pas arrivée ?
                {C          eOu
                          ut=
                          0),gotoxy(2,10);
                          p          as de trame resultats de conversion depuis trop longtemps\n");
                          prin          faut recharger et relancer le programme \n");
                          (1);}} // Stop
        se {Cptr_Time
                rame_          ent.extend.identificateur.ident==Ident_T_OB_Asservissement)
                // C          l'identificateur est correct
                {AN1          me_Recue.data[3]; // On récupére les MSB de la mesure vitesse
                AN10L =          rame_Recue.data[4]; // On récupére les LSB AN1 et AN0
                // Traiter les données et reconstituer les résultats
                S_Mesure_Vitesse=(unsigned short)(AN1H);          //Transfert avec transtypage
                S_Mesure_Vitesse=S_Mesure_Vitesse<<2;             // Décaler de 2 bits vers poids forts
                S_Temp=(unsigned short)(AN10L&0xC0); // Pour ne récupérer que les 2 bits AD1 et AD0
                S_Mesure_Vitesse=S_Mesure_Vitesse|(S_Temp>>6);
                // Calculer la grandeur de commande
                Ecart = S_Consigne - S_Mesure_Vitesse;
                Resultat_Calcul = (Kp*Ecart)>>4;
                if(Resultat_Calcul>255)Resultat_Calcul=255;
                if(Resultat_Calcul<0)Resultat_Calcul=0;
                Cde_Moteur=(unsigned char)(Resultat_Calcul);
                T_IM_Asservissement.data[0]=0x25;        // Adresse du registre PWM1DC (Charger cde vitesse)
                T_IM_Asservissement.data[1]=0xFF;        // Masque -> tous les bits sont concernés
                T_IM_Asservissement.data[2]=Cde_Moteur;          // Valeur ->  Commande vitesse
                Ecrire_Trame(T_IM_Asservissement);
```

```
                                do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre réponse
                                // Acquisition de la consigne
                                Ecrire_Trame(T_IRM_Commodo_EG);
                                do{}while(Lire_Trame(&Trame_Recue)==0);
                                if(Trame_Recue.ident.extend.identificateur.ident==Ident_T_IRM8_Commodo_EG)
                                        {AN0H =Trame_Recue.data[2]; // On récupére les MSB entree analogique Commodo EG
                                         AN10L =Trame_Recue.data[4]; // On récupére les LSB AN1 et AN0
                                    // Traiter les données et reconstituer les résultats
                                        S_Consigne=(unsigned short)(AN0H);       //Transfert avec transtypage
                                        S_Consigne=S_Consigne<<2;   // Décaler de 2 bits vers poids forts
                                        S_Temp=(unsigned short)(AN10L&0x0C); // Pour ne récupérer que les 2 bits AD1 et AD0
                                        S_Consigne=S_Consigne|(S_Temp>>2);
                                        }
                                }} // Fin acquisition et traitement

                Cptr_Affichage++;
                if(Cptr_Affichage==70000)printf("Cs=%d\n",S_Consigne);
                if(Cptr_Affichage==80000)printf("Ms=%d\n",S_Mesure_Vitesse);
                if(Cptr_Affichage==90000)printf("Ec=%d\n",Ecart);
                if(Cptr_Affichage==100000)printf("Sr=%d\n",Cde_Moteur);
                if(Cptr_Affichage==110000)printf("\n");
                if(Cptr_Affichage==120000)
                        {Cptr_Affichage=0;
                         gotoxy(1,12);
                         printf("  Valeur de l'entree Consigne:              \n");
                         printf("  Valeur de l'entree Mesure vitesse:        \n");
                         printf("  Ecart = Consigne - Mesure:                \n");
                         printf("  Valeur de la Commande moteur:             \n");
                        // Afficher grandeurs
                        gotoxy(1,12);
                         printf("  Valeur de l'entree Consigne: %d\n",S_C
                         printf("  Valeur de l'entree Mesure vitesse: %
                         printf("  Ecart = Consigne - Mesure: %d\n
                         printf("  Valeur de la commande moteur:%
                         printf("  On doit verifier la relation:\n"
                         printf("  Commande mot      Kp*Ec      c Kp=      );

                        }
                } //  Fin boucle principale
} // Fin fonction principale
```

Page laissée vierge

TP N°6: Commande du système Essuie Glace

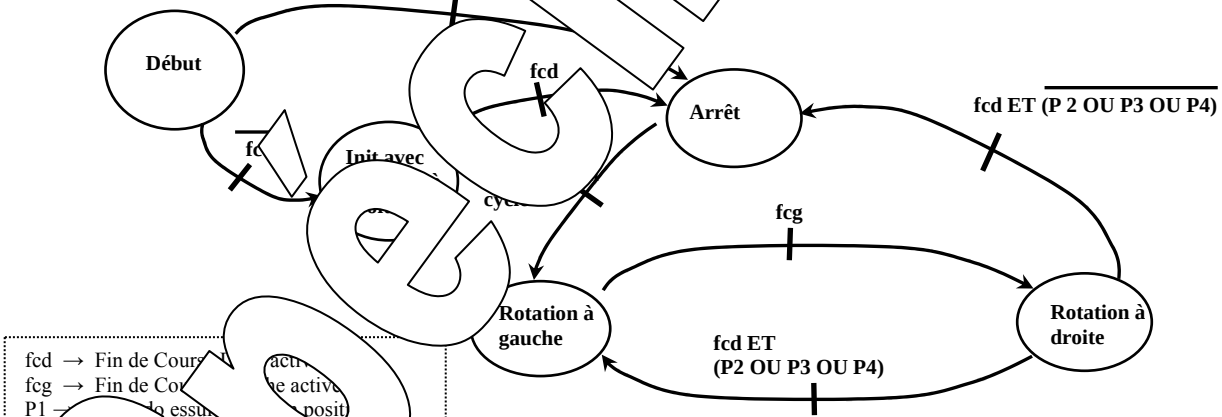
5.5 Sujet

Objectifs :	- Développer une application satisfaisant un cahier des charges imposé
Cahier des charges :	A intervalles de temps réguliers, on interroge le module sur lequel est connecté le commodo essuie glace afin de connaître son état. En fonction de l'état du commodo essuie glace, on commande le moteur - position 'arrêt' (noté P0) (ni P1, ni P2,ni P3) - position P1 ou appui sur BP1 (GP4) 'intermittant' -> le balai fait des "aller-retours" séparés pa une attente dont la durée est réglée pa ique AN0 - position P2 ou appui sur BP2 (GP5) -> le balai fait des "aller-retours" avec e faible - position P3 ou appui sur BP3 (GP6) -> le balai fait des "aller-retours" une vi nne - position P4 ou appui sur BP4 (GP7) -> le balai fait des "aller-ret avec esse éle Remarque : Dans le mode 'intermittent', alle emps entre deux battements est réalisée par le 'temp ur p e intégré dans le micro-controleur

5.6 Eléments de solution

Principe:

Le cycle demandé conduit au diagramm états t:

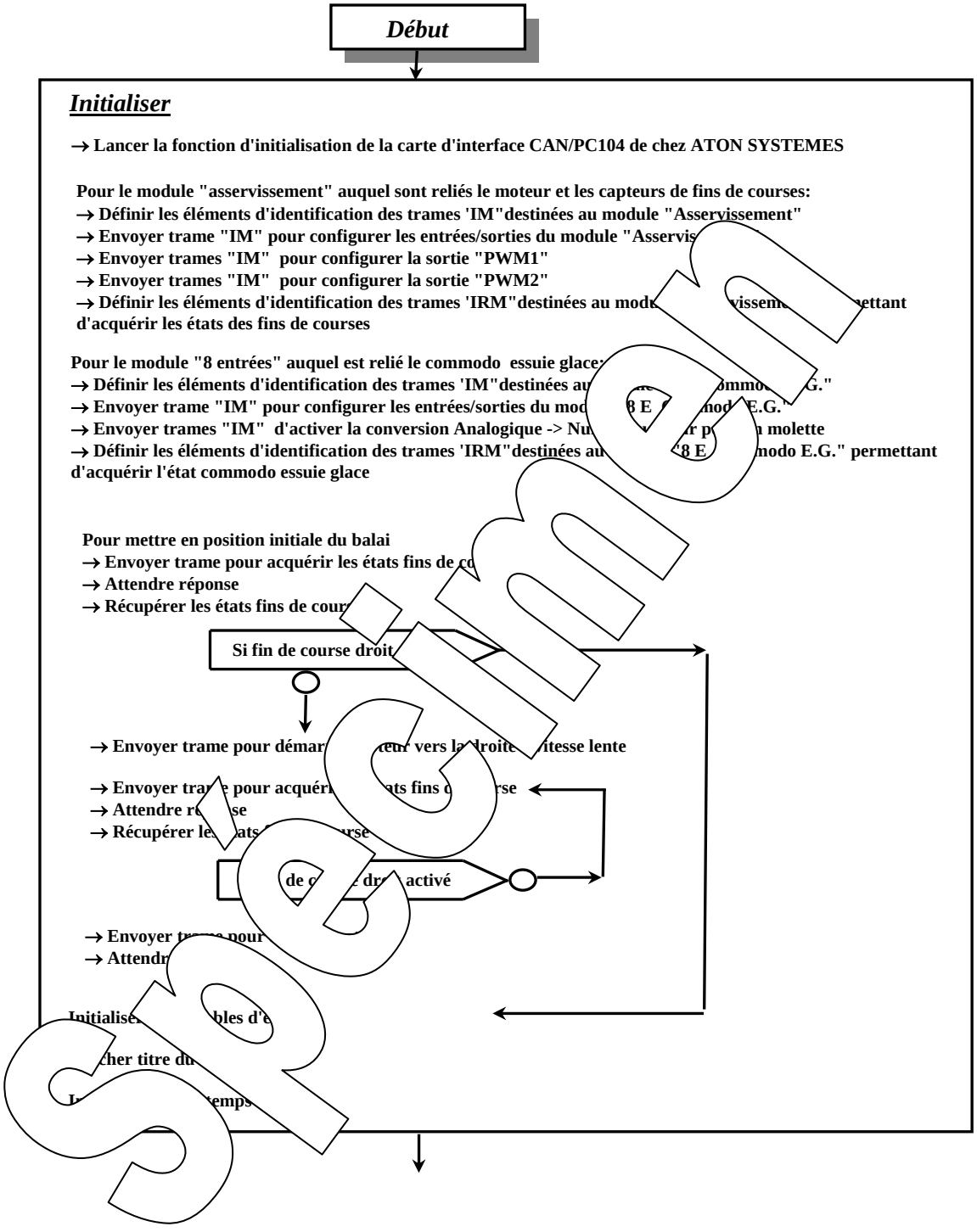


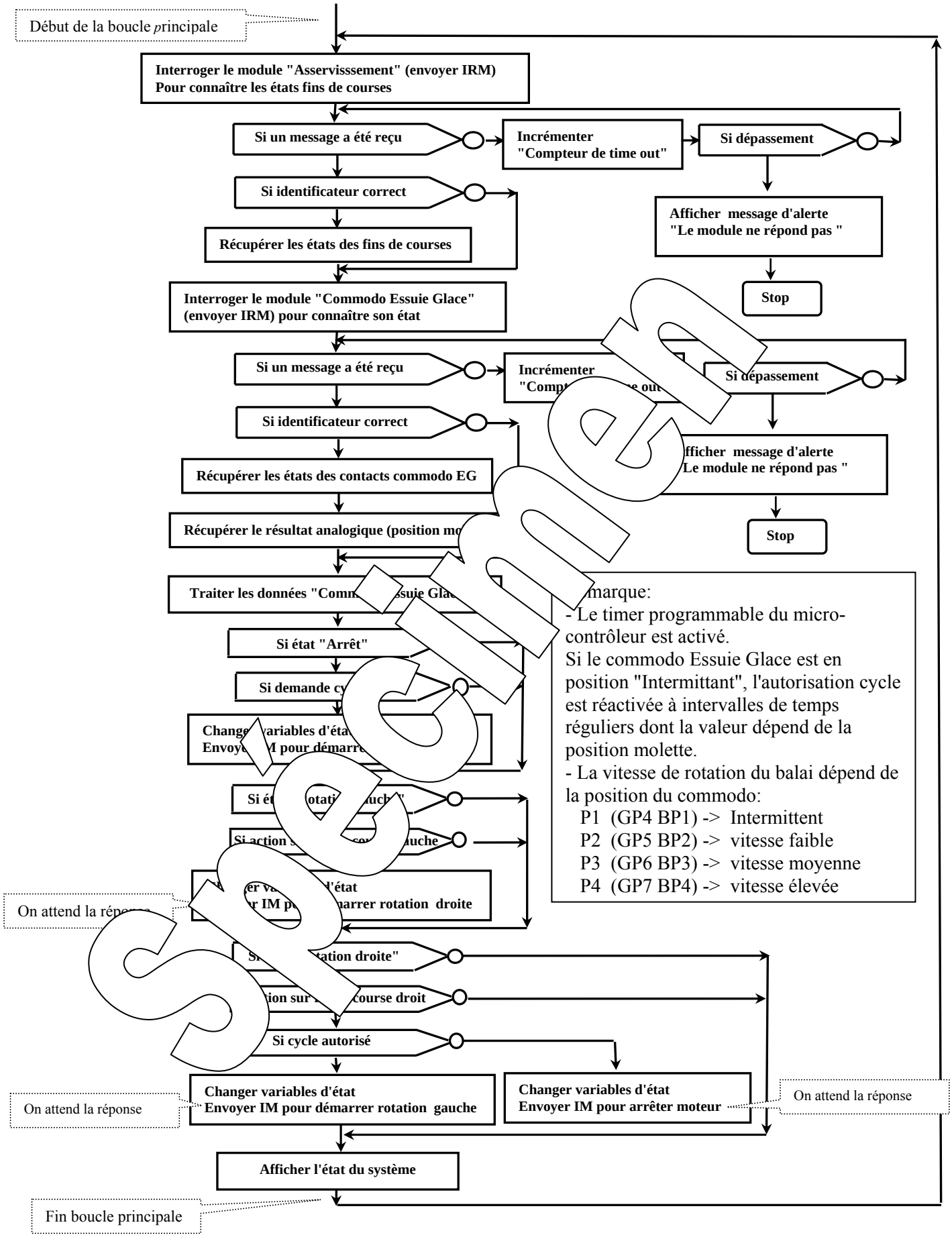
fcd → Fin de Cours ... activ...
fcg → Fin de Cou... e active
P1 → ...do essu... posit...
...i sur BP1)
P... ...ommodo es...ie glace ...2
(Ou a...
P... ...n posi...
...ur BP...
P24 → commodo essuie g... position 4
(Ou appui ...

Remarques:

- Dans les deux états "Rotation Gauche" et "Rotation Droite" la vitesse dépend de la position commodo essuie glace: Position 1 ou intermittent → Vitesse lente, Position 2 → Vitesse rapide.
- Si le commodo est dans la position "Intermittent", une base de temps met régulièrement à 1 la variable "Autorisation cycle". Cette dernière est remise à 0 à l'activation de l'état "rotation gauche". L'intervalle de temps entre deux activation de "Autorise cycle" dépend de la position de la molette du commodo.

5.7 Organigramme




Début de la boucle principale
Interroger le module "Asservisssement" (envoyer IRM)
Pour connaître les états fins de courses
Si un message a été reçu
Incrémenter
"Compteur de time out"
Si dépassement
Si identificateur correct
Afficher message d'alerte
"Le module ne répond pas "
Récupérer les états des fins de courses
Stop
Interroger le module "Commodo Essuie Glace"
(envoyer IRM) pour connaître son état
Si un message a été reçu
Incrémenter
Si dépassement
Si identificateur correct
Récupérer les états des contacts commodo EG
Stop
Si état "Arrêt"
Si demande cy
Si action
Changer variables d'état
Envoyer IM pour démarrer rotation droite
On attend la réponse
Si cycle autorisé
Changer variables d'état
Envoyer IM pour démarrer rotation gauche
Changer variables d'état
Envoyer IM pour arrêter moteur
On attend la réponse
On attend la réponse
Afficher l'état du système
Fin boucle principale
marque:
- Le timer programmable du micro-contrôleur est activé.
Si le commodo Essuie Glace est en position "Intermittant", l'autorisation cycle est réactivée à intervalles de temps réguliers dont la valeur dépend de la position molette.
- La vitesse de rotation du balai dépend de la position du commodo:
P1 (GP4 BP1) -> Intermittent
P2 (GP5 BP2) -> vitesse faible
P3 (GP6 BP3) -> vitesse moyenne
P4 (GP7 BP4) -> vitesse élevée

5.8 Programme

```
/*****************************************************************************************
*      TPs sur  EID210 / Réseau CAN - VMD  (Véhicule Multiplexé Didactique)
*****************************************************************************************
*      APPLICATION: Commande Essuie-glace à distance
*****************************************************************************************
*   TP n°6:  Commande du système essuie glace avant
*----------------------------------------------------------------------------
*   CAHIER DES CHARGES :
*  **********************
*  On souhaite une commande normale de l'essuie glace à partir du commodo destiné
*  à cet effet:
*              - position 'arrêt' (noté P0)    (ni P1, ni P2,ni P3)
*              - position P1 (GP4-BP1) 'intermittant' -> le balai fait des "aller-retours" séparés par
*                une attente dont la durée est réglée par l'entrée analogique AN0
*              - position P2 (GP5-BP2) -> le balai fait des "aller-retours" avec une vites
*              - position P3 (GP6-BP3) -> le balai fait des "aller-retours" avec une vi
*              - position P4 (GP7-BP4) -> le balai fait des "aller-retours" avec une
*   Dans le mode 'intermittent', l'intervalle de temps entre deux battements est gé
*   par le 'temporisateur programmable intégré dans le micro-controleur
*   On affichera à l'écran, les états des diverses entrées commodo.
*----------------------------------------------------------------------------
*  NOM du FICHIER :  TP_6_CAN_VMD_EG.C
*  *****************
*****************************************************************/
// Déclaration des fichiers d'inclusion
#include <stdio.h>
#include"Structures_Donnees.h"
#include"cpu_reg.h"
#include "eid210_reg.h"
#include "CAN_vmd.h"
#include "Aton_can.h"

// Déclaration des variables
// Pour les Indicateurs divers (variables binaires)
union byte_bits Indicateurs,FC; // Structures de bits
#define I_Autorise_Cycle Indicateurs.bit.b0              // Po          r 1 bat    ent
#define I_Intermittent Indicateurs.bit.b1
#define I_Message_Pb_Affiche Indicateurs.bi
#define Etat_Arret Indicateurs.bit.b3                    // Etat "      ai à l'a  êt en fin de course droite"
#define Etat_Rot_Droite Indicateurs.bit.b4  //           balai en r      droite"
#define Etat_Rot_Gauche In  cateurs.bit.b5//           lai en ro       gauche"
// Pour les fins de course
#define Etat_FC FC.valeur //      ur
#define fs FC.bit.b7    // Pour fin
#define fcg FC.bit.b6 // Pour fin      urse
#define fcd FC.bit.b5 // Pour fin      s
// Pour la position Commodo_EG
#define Commodo_EG_P     Comm
#define Commodo_EG       odo_
#define Commodo_EG       EG.bit.GP6
#define Commodo          4 Com      G.bit.GP7
// Déc      ons des      mes de      unication
Tr       Recue;//          ent d'etre reçue par le controleur
//       type "IM" (Inp             trame de commande)
T_IM       ent;                 commande du moteur
//                              / Pour l'initialisation Commodo Essuie Glace
//                 Messag       me interrogative)
Trame       Acquisiti       ; // Pour l'acquisition de l'état des fins de courses
Trame T_IRM_E      Co       _EG;                    // Pour l'acquisition de l'état Commodo Essuie Glace
// Varibles dive
unsigned char Va       ogique,Cde_Vitesse,Tempo_Fin,Compteur_Passage_Irq,Compteur_Secondes;
unsigned char Valeur_Commodo_EG_Mem,Valeur_Analogique_Mem;
// Pour les temporisation
// Déclaration constantes
#define Vitesse_Lente 30
#define Vitesse_Moyenne 50
#define Vitesse_Rapide 70

#define Tempo1 3     // en secondes
#define Tempo2 6
#define Tempo3 9
#define Tempo4 12
#define Tempo5 15
```

```
//  Fonction d'interruption "Base de Temps"
//=======================================
void irq_bt()
// Fonction exécutée toute les 10 mS
{if(I_Intermittent) // Si mode intermittent actif
        {Compteur_Passage_Irq++;
        if(Compteur_Passage_Irq==100)  // Une  Seconde s'est écoulée
                {Compteur_Passage_Irq=0;
                Compteur_Secondes++;
                if(Compteur_Secondes>=Tempo_Fin)
                        {Compteur_Secondes=0;
                        I_Autorise_Cycle=1;}
                }}
} // Fin de la fonction d'interruption

//======================
// FONCTION PRINCIPALE
//======================
main()
{
// Déclaration de variables locales à la fonction principale
int Cptr_Affichage=0,Cptr_TimeOut;
//  INITIALISATIONS
//------------------
// Pour initialiser la carte industrielle controleur CAN
Init_Aton_CAN();
clsscr(); // Pour effacer l'écran
// Trame de type "IM" (trame de commande): Données d'identification
T_IM_Asservissement.trame_info.registre=0x00;
T_IM_Asservissement.trame_info.champ.extend=1;
T_IM_Asservissement.trame_info.champ.dlc=0x03;
T_IM_Asservissement.trame_info.champ.rtr=0;
T_IM_Asservissement.ident.extend.identificateur.ident=Ident_T
// Pour définir des Entrées/Sorties
T_IM_Asservissement.data[0]=0x1F;        // Adresse du reg        DR        de E/S)
                                                 MCP2        e 16
T_IM_Asservissement.data[1]=0xEF;        // M        -> Bit 7 n
T_IM_Asservissement.data[2]=0xE3;        // Valeur -        Entrée
            // GP7=fs Entrée; GP6=fcg Entree; GP5=fcd        P4=Va        tie;
            // GP3=PWM2 Sortie; GP2=PWM1 Sortie; GP1        ée; G        AN0 Entrée;
I_Message_Pb_Affiche=0;
do {Ecrire_Trame(T_IM_Asservissement); //        mière tram        ée au module
            Cptr_TimeOut=0;                 rvissement'        on te        s'il répond bien
            do{Cptr_TimeOut++;}while((Li        me(&Tra        ue)==0)&&(Cptr_TimeOut<100));
            if(Cptr_TimeOu        =100)
                    {if(I_        sage_Pb_Affich
                            {I        Pb_A

                    f(" P        repo        se a la trame de commande en initialisation \n");
                    "        ier        mentation 12 V  est OK \n");}}

   }while(Cptr_TimeOut==100);
clsscr(); // Pour effacer l'        au cas
// Pour mettre à 0 les s
T_IM_Asservissement        // Adresse du registre GPLAT  (Registre I/O)
T_IM_Asservisse        1]=0x        // Masque -> sorties GP4,3,2 sont consernées
T_IM_Asservissem        1=0x00;        // Valeur ->  les 3 sorties à 0
Ec        T_IM_A
d        re_Trame(&        =0); // Attendre réponse
        définir        en P
        0]=0x2        // Adresse du registre T1CON
T        =0xB3;        Masque -> seuls bit 7;5;4;1;0 consernés
T_IM        sement.da        0x80;        // Valeur ->  TMR1ON=1; Prescaler1=1
Ecrire_Trame(T_IM_As        ement);
do{}while(Lire        ne_Recue)==0); // Attendre réponse
// Pour définir fr        nal sortie PWM1
T_IM_Asservissement.data[0]=0x23;        // Adresse du registre PR1
T_IM_Asservissement.data[1]=0xFF;        // Masque -> tous les bits sont consernés
T_IM_Asservissement.data[2]=0xFF;        // Valeur ->  PR1=255
Ecrire_Trame(T_IM_Asservissement);
do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre réponse
// Pour définir sortie GP3 en PWM2
T_IM_Asservissement.data[0]=0x22;        // Adresse du registre T2CON
T_IM_Asservissement.data[1]=0xB3;        // Masque -> seuls bit 7;5;4;1;0 consernés
T_IM_Asservissement.data[2]=0x80;        // Valeur ->  TMR2ON=1; Prescaler2=1
Ecrire_Trame(T_IM_Asservissement);
do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre réponse
// Pour définir fréquence signal sortie PWM2
```

```
T_IM_Asservissement.data[0]=0x24;          // Adresse du registre PR2
T_IM_Asservissement.data[1]=0xFF;          // Masque -> tous les bits sont consernés
T_IM_Asservissement.data[2]=0xFF;          // Valeur ->  PR2=255
Ecrire_Trame(T_IM_Asservissement);
do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre réponse
// Pour initialiser le rapport cyclique du signal PWM1
T_IM_Asservissement.data[0]=0x25;          // Adresse du registre PWM1DC
T_IM_Asservissement.data[1]=0xFF;          // Masque -> tous les bits sont consernés
T_IM_Asservissement.data[2]=0;             // Valeur ->  PWM1DC=0
Ecrire_Trame(T_IM_Asservissement);
do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre réponse
// Pour initialiser le rapport cyclique du signal PWM2 à 0
T_IM_Asservissement.data[0]=0x26;          // Adresse du registre PWM2DC
T_IM_Asservissement.data[1]=0xFF;          // Masque -> tous les bits sont consernés
T_IM_Asservissement.data[2]=0;             // Valeur ->  PWM2DC=0
Ecrire_Trame(T_IM_Asservissement);
do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre réponse
// Pour Valider le circuit de puissance
T_IM_Asservissement.data[0]=0x1E;          // Adresse du registre GPLAT  (Registre I/O)
T_IM_Asservissement.data[1]=0x10;          // Masque -> sortie GP4 (ValidIP) est conserné
T_IM_Asservissement.data[2]=0x10;          // Valeur ->  ValidIP=1
Ecrire_Trame(T_IM_Asservissement);
do{}while(Lire_Trame(&Trame_Recue)==0); // Attendre réponse
// Masque pour les commandes IM futures
T_IM_Asservissement.data[1]=0xFF;          // Masque -> tous les bits sont conser
// Pour acquérir l'état des fin de course
// Trame de type "IRM" (trame interrogative): Données d'identification
T_IRM_Acquisition_FC.trame_info.registre=0x00;
T_IRM_Acquisition_FC.trame_info.champ.extend=1;
T_IRM_Acquisition_FC.trame_info.champ.dlc=1;
T_IRM_Acquisition_FC.trame_info.champ.rtr=1;
T_IRM_Acquisition_FC.ident.extend.identificateur.ident=Ident_T                 Asser
                                           // Demande état regi

// Pour initialiser le module "commmodo EG"
// Trame de type "IM" (trame de commande): D                 d'iden
T_IM_Commodo_EG.trame_info.registre=0x00;
T_IM_Commodo_EG.trame_info.champ.extend=1;
T_IM_Commodo_EG.trame_info.champ.dlc=0x03; // O                 les va                 8 rgistres
T_IM_Commodo_EG.trame_info.champ.rtr=0;
T_IM_Commodo_EG.ident.extend.identificateu                 dent_T_                 modo_EG; //
// Pour activer les conversions Ana -> Num
T_IM_Commodo_EG.data[0]=0x2A;              sse du regis                 ADCON0
T_IM_Commodo_EG.data[1]=0xF0;              ue -> bits                 oncernés
T_IM_Commodo_EG.data[2]=0x80;              >  ADC                 t "prescaler rate"=1:32
Ecrire_Trame(T_IM_Comm                 EG);
do{}while(Lire_Trame(&Tra                 // Att
// Pour définir le mode de conver
T_IM_Commodo_EG.data[0]=0                 ress                 du registre ADCON1
T_IM_Commodo_EG.data[1]=0x                 Mas                 > tous les bits sont concernés
T_IM_Commodo_EG.data[2]=0x                 Val                 voir doc MCP25050 page 36 (GP0 -> Entrée Analogique)
Ecrire_Trame(T_IM_Co                 EG);
do{}while(Lire_Trame                 ue)                 ndre réponse
// Pour acquérir les r                 n A->N et états commmodo EG
// Trame de type "I                 me inte                 e) Commodo EG: Données d'identification
T_IRM                 Comm                 rame_I                 istre=0x00;
T                 Commodo                 mp.extend=1;
T                 Commodo_E                 champ.dlc=0x08; // On demande les valeurs de 8 rgistres
                 d_Etat                 EG.tra                 hamp.rtr=1;
T                 ident.e                 ntificateur.ident=Ident_T_IRM8_Commodo_EG; //

// Pou                 ef système                 ener le balai en position sur fin de course droit)
 T_IM_Asservis                 ent                 =0x25;          // Adresse du registre PWM1DC
 T_IM_Asservi                 ]=0xFF;          // Masque -> tous les bits sont consernés
 Ecrire_Trame(T                 quisition_FC);    // Envoi trame d'acquisition de l'état des fins de course
 do{}while(Lire_Trame(&Trame_Recue)==0);          //On attend la réponse
 Etat_FC=~Trame_Recue.data[0];              // On récupére l'état des fin de course
 if(fcd==0)              //SI le balai EG n'est pas en position droit, on commande une rotation à droite
              {T_IM_Asservissement.data[2]=Vitesse_Lente;          // Valeur ->  Pour vitesse lente
              Ecrire_Trame(T_IM_Asservissement);       // Envoi trame de commande moteur
              do{}while(Lire_Trame(&Trame_Recue)==0);          //On attend la réponse
              while(fcd==0)
                           {Ecrire_Trame(T_IRM_Acquisition_FC);   // Envoi trame d'acquisition états fins de course
                            do{}while(Lire_Trame(&Trame_Recue)==0);          //On attend la réponse
                            Etat_FC=~Trame_Recue.data[0];              // On récupére l'état des fin de course
                           }
              }
```

```
 T_IM_Asservissement.data[2]=0;              // Valeur ->  Pour vitesse nulle
 Ecrire_Trame(T_IM_Asservissement);
 do{}while(Lire_Trame(&Trame_Recue)==0);           //On attend la réponse

// Initialisation des variables d'état système
Etat_Arret=1,Etat_Rot_Droite=0,Etat_Rot_Gauche=0;
I_Autorise_Cycle=0,I_Intermittent=0;
Compteur_Secondes=0,Compteur_Passage_Irq=0;
// Pour afficher titre du TP
gotoxy(1,2);
printf("************************************************\n");
printf("         TPs sur Reseau CAN                  \n");
printf("    Application:  Commande Essui-Glace       \n");
printf(" ---------------------------------------------\n");
printf("     TP n°6:   Commande systeme ESSUI GLACE    \n");
printf("************************************************\n");
// Pour initialiser la base de temps et temporisations
SetVect(96,&irq_bt); // mise en place de l'autovecteur
PITR = 0x0048;                    // Une interruption toutes les 10 millisecondes
PICR = 0x0760;                    //  96 = 60H

// BOUCLE PRINCIPALE
//******************
while(1)
 {          // Acquérir l'état des fins de courses
            //-------------------------------------
            Ecrire_Trame(T_IRM_Acquisition_FC); // Envoi trame d'acquisition
            Cptr_TimeOut=0;
            do{Cptr_TimeOut++;}while((Lire_Trame(&Trame_Recue)==0)&&(Cptr_TimeOut<10000));
             if(Cptr_TimeOut==10000)
                        {clsscr(),gotoxy(2,10);
                         printf(" Pas de reponse a la trame interrogati
                         printf("  Il faut recharger et relancer le pro
                         do{}while(1);} // Stop
             else { if(Trame_Recue.ident.extend.id
                                    // On teste si l'
                                    {Etat_FC=~Tra
            // Acquérir l'état Commodo Essuie Glace
            //-------------------------------------
            Ecrire_Trame(T_IRM_Etat_Commodo_EG); //
            Cptr_TimeOut=0;
            do{Cptr_TimeOut++;}while((Lire
             if(Cptr_TimeOut==10000)
                        {clsscr(),gotoxy(2,10);
                         printf
                         print
                         do{}w
             else { if(Trame_Recu
                                    teste
                                    ir
                                    alogi
            // Traiter l'état
            //-------------
            if(Valeur_C
            {if(Con
            // (on au
                        {if(Con
            (on autorise le
            els
                                    cycles
                                    EG_Pos
                                    {I_Intermittent=1,Cde_Vitesse=Vitesse_Lente; // Position "Intermittent"
                                     Compteur_Passage_Irq=0;
                                     Compteur_Secondes=0;
                                     }
            else {I_Intermittent=0; // Position "Arret"
            }}}}
            printf("Bh %x\n",Valeur_Commodo_EG);
            }// Fin si changement etat entrées binaires
            if(Valeur_Analogique!=Valeur_Analogique_Mem)
            {printf("Ad %d\n",Valeur_Analogique);
                        if(Valeur_Analogique>=200)Tempo_Fin=Tempo5;      // Selon la position de la molette
                        else {if(Valeur_Analogique>=150)Tempo_Fin=Tempo4;            // la tempo plus ou moins longue
                        else {if(Valeur_Analogique>=100)Tempo_Fin=Tempo3;
                        else {if(Valeur_Analogique>=50)Tempo_Fin=Tempo2;
                        else {if(Valeur_Analogique>=1)Tempo_Fin=Tempo1;}}}}
            }
```

```
        // Mise à jour des grandeurs mémorisées
        Valeur_Analogique_Mem=Valeur_Analogique;
        Valeur_Commodo_EG_Mem=Valeur_Commodo_EG;

        // Traitement du diagramme des états "système"
        //-----------------------------------------------
        if(Etat_Arret)          // Si le système est dans l'état "Arret"
                {if(I_Autorise_Cycle)                   // Si on un cycle a été autorisé
                        {I_Autorise_Cycle=0;
                         Etat_Arret=0,Etat_Rot_Gauche=1;        // Evolution etat système
                         T_IM_Asservissement.data[2]=Cde_Vitesse; // On commande la rotation à gauche
                         T_IM_Asservissement.data[0]=0x26;      // Adresse du registre PWM2DC
                         Ecrire_Trame(T_IM_Asservissement);
                         while(Lire_Trame(&Trame_Recue)==0){};          // On attend la réponse
                         //if(I_Intermittent)Compteur_Secondes=0,Compteur_Passage_Irq=0;
                        }}
        if(Etat_Rot_Gauche) // Si le système est dans l'état "Rotation à Gauche"
                {if(fcg)                // Si on est arrivé en fin de course gauche
                        {Etat_Rot_Gauche=0,Etat_Rot_Droite=1;   // Evolution etat s
                         T_IM_Asservissement.data[2]=0;         // On arrête la r
                         T_IM_Asservissement.data[0]=0x26;      // Adresse du                PWM2
                         Ecrire_Trame(T_IM_Asservissement);
                         while(Lire_Trame(&Trame_Recue)==0){};          // On att          onse
                         T_IM_Asservissement.data[2]=Cde_Vitesse; // O          le la r          uche
                         T_IM_Asservissement.data[0]=0x25;      // A                 PWM
                         Ecrire_Trame(T_IM_Asservissement);
                         while(Lire_Trame(&Trame_Recue)==0){};          //          ten          éponse
                        }}
        if(Etat_Rot_Droite)   // Si le système est dans l'état "Rotation à          ite"
                {if(fcd)                // Si on est arrivé en fin de
                  {if(Commodo_EG_Pos2|Commodo_EG_Pos3|                          Pos4)
                        {Etat_Rot_Droite=0,Etat_Rot_G                   // E          tat système
                         T_IM_Asservissement.data[2]                    // On ar          ation à droite
                         T_IM_Asservissement.data[0]                    dresse d          egistre PWM1DC
                         Ecrire_Trame(T_IM_Asse                         nt);
                         while(Lire_T           %Trame_                 =0){};          // On attend la réponse
                         T_IM_Asserv            nem.data[2]-            sse; // C          commande la rotation à gauche
                         T_IM_Asservissem           0]=0x2              Adresse du registre PWM2DC
                         Ecrire_Trame(T_IM_                             ment);
                         while(Lire_Trame(&Tram                         =0){};          // On attend la réponse
                        }
                   else
                        {Etat_Rot_                      =0,Etat_Arr             1;          // Evolution etat système
                         T_IM_Ass                       ment.data               // On arrête la rotation à droite
                         T_IM_Asse                      nt.data[                25;         // Adresse du registre PWM1DC
                         Ecrire_Trame(                                          ient);
                         w                      ame(                            cue)==0){};          // On attend la réponse
                                                Compteur_Passage_Irq=0,Compteur_Secondes=0;

        // Afficher l'état systè
        //-------------------------
        /*
    Cptr_Affichage++;
        if(Cptr_A
                  Affich
                (1,6);
                f("  Essuie glace avant en intermittent     \n");
                _EG_Pos2)printf("  Essuie glace avant en vitesse lente     \n");
                {if(C          EG_Pos3)printf("  Essuie glace avant en vitesse rapide    \n");
                  rintf(          lace avant à l'arret          \n");}}}
                  Comma          e glace avant: %d\n",Cde_Lave_Glace_Av);
                go          ,10);
                  Commande essuie glace arriere: %d\n",Cde_EG_Ar);
                  Commande lave glace arriere: %d\n",Cde_Lave_Glace_Ar);
                in "Afficher l'état"
        */
        } //  Fin boucle principale
} // Fin fonction principale
```

6 ANNEXES

6.1 Fichier de définition propre au système CAN_VMD

```
//*****************************************************
//  Structures de données pour application  CAN VMD
//  Nom du fichier: CAN_VMD.h
//*****************************************************
#ifndef _VMD_H
#define _VMD_H
// Format de message
typedef struct {
   /* nombre d'octets à envoyer, -1 si c'est une Remote Frame */
   int dlc;
   unsigned char id1;  /* 8 bits de poids forts de l'ID. */
   unsigned char id2;  /* 3 bits de poids faible de l'ID. */
   unsigned char data[8];
} Message;
//  Pour l'identificateur en mode standard
typedef union
{struct      {unsigned short ident:11;
              unsigned short rtr:1;
              unsigned short nul:4;
             } identificateur;
 struct      {unsigned char ident1;
              unsigned char ident2;
             } registre;
 unsigned short valeur;
} ident_standard;
//  Pour l'identificateur en mode étendu
typedef union
{struct      {unsigned long ident:29;
              unsigned long rtr:1;
              unsigned long x:2;
             } identificateur;
 struct      {unsigned char ident1;
              unsigned char ident2;
              unsigned char ident3;
              unsigned char ident4;
             } registre;
 unsigned long valeur;
} ident_extend;
// Pour registre d'information
typedef union
              {struct      {unsigned
                            unsigned
                            unsigned c
                            unsigned cha
                            }
              unsigned c
              } tr_info
// Po          circula          au CA
ty
t           tram
u                                ndard;
                 extendu         d;
              } ident;
unsigned char da
} Trame;
 typedef union
{struct      {unsigned char GP7:1;
              unsigned char GP6:1;
              unsigned char GP5:1;
              unsigned char GP4:1;
              unsigned char GP3:1;
              unsigned char GP2:1;
              unsigned char GP1:1;
              unsigned char GP0:1;
             }bit;
unsigned char valeur;
}Port_8ES;
```

```
Port_8ES Etat_Commodo_Feux;
#define Valeur_Commodo_Feux Etat_Commodo_Feux.valeur
#define Cde_Veilleuse Etat_Commodo_Feux.bit.GP0
#define Cde_Warning Etat_Commodo_Feux.bit.GP1
#define Cde_Phare Etat_Commodo_Feux.bit.GP2
#define Cde_Code Etat_Commodo_Feux.bit.GP3
#define Cde_Clign_Gauche Etat_Commodo_Feux.bit.GP4
#define Cde_Clign_Droit Etat_Commodo_Feux.bit.GP5
#define Cde_Stop Etat_Commodo_Feux.bit.GP6
#define Cde_Klaxon Etat_Commodo_Feux.bit.GP7
// Pour la Commande des feux
#define Cde_Nulle 0x00
#define Cde_FV_V 0x01 // Feux aVant en Veilleuse
#define Cde_FR_V 0x01 // Feux aRrière en Veilleuse
#define Cde_FV_C 0x03 // Feux aVant en Code
#define Cde_FR_C 0x01 // Feux aRrière en Code
#define Cde_FV_P 0x05 // Feux aVant en Phare
#define Cde_FR_P 0x01 // Feux aRrière en Phare
#define Masque_Clign_AV 0x08 // Pour Clignotant AVant
#define Masque_Clign_AR 0x04 // Pour Clignotant ARrière
#define Masque_Klaxon 0x08 // Pour Claxon
#define Masque_Stop 0x02 // Pour Stop
//  Variables pour la commande et le control des feux
//-------------------------------------------------------
//  Pour variables images de la commande effectuée des différents feux
union byte_bits Image_FVG,Image_FVD,Image_FRD,Image_FRG;
// Pour Feux aVant Gauche
#define Valeur_FVG Image_FVG.valeur     // Pour un accés port complet
#define Veilleuse_FVG Image_FVG.bit.b0
#define Code_FVG Image_FVG.bit.b1
#define Phare_FVG Image_FVG.bit.b2
#define Clignot_FVG Image_FVG.bit.b3
// Pour Feux aVant Droit
#define Valeur_FVD Image_FVD.valeur     // Pour un accés port
#define Veilleuse_FVD Image_FVD.bit.b0
#define Code_FVD Image_FVD.bit.b1
#define Phare_FVD Image_FVD.bit.b2
#define Clignot_FVD Image_FVD.bit.b3
// Pour Feux aRrière Droit
#define Valeur_FRD Image_FRD.valeur     // Pour un accés                let
#define Veilleuse_FRD Image_FRD.bit.b0
#define Stop_FRD Image_FRD.bit.b1
#define Clignot_FRD Image_FRD.bit.b2
#define Klaxon_FRD Image_FRD.bit.b3
// Pour Feux aRrière Gauche
#define Valeur_FRG Image_FRG.valeur     // Po                mplet
#define Veilleuse_FRG Imag_FRG
#define Stop_FRG Image_FRG.b
#define Clignot_FRG Image_FR      b2
#define Klaxon_FRG Image_FR      3
// Pour les variables images des "S          sortie          uissance
union byte_bits Image_S      VG Im          age_Stat_FRD,Image_Stat_FRG;
// Pour image "Status"          auch
#define Valeur_Stat          FVG.valeur // Pour un accés port complet
#define S_Veilleu          mage_S          G.bit.b4
#defi          Code_FV          Stat_F          b5
#d          e_FVG I          b6
#          ignot_FVG Im          G.bit.b7
/          image          x aVa
#          Image_          D.valeur // Pour un accés port complet
#d          e_Stat_          t.b4
#defin          e_FVD Im          tat_FVD.bit.b5
#define S_Phare          Stat_FVD.bit.b6
#define S_Clig          e_Stat_FVD.bit.b7
// Pour image "St          x aRrière Droit
#define Valeur_Status_FRD Image_Stat_FRD.valeur // Pour un accés port complet
#define S_Veilleuse_FRD Image_Stat_FRD.bit.b4
#define S_Stop_FRD Image_Stat_FRD.bit.b5
#define S_Clignot_FRD Image_Stat_FRD.bit.b6
#define S_Klaxon_FRD Image_Stat_FRD.bit.b7
// Pour image "Status" Feux aRrière Gauche
#define Valeur_Status_FRG Image_Stat_FRG.valeur // Pour un accés port complet
#define S_Veilleuse_FRG Image_Stat_FRG.bit.b4
#define S_Stop_FRG Image_Stat_FRG.bit.b5
#define S_Clignot_FRG Image_Stat_FRG.bit.b6
#define S_Klaxon_FRG Image_Stat_FRG.bit.b7
```

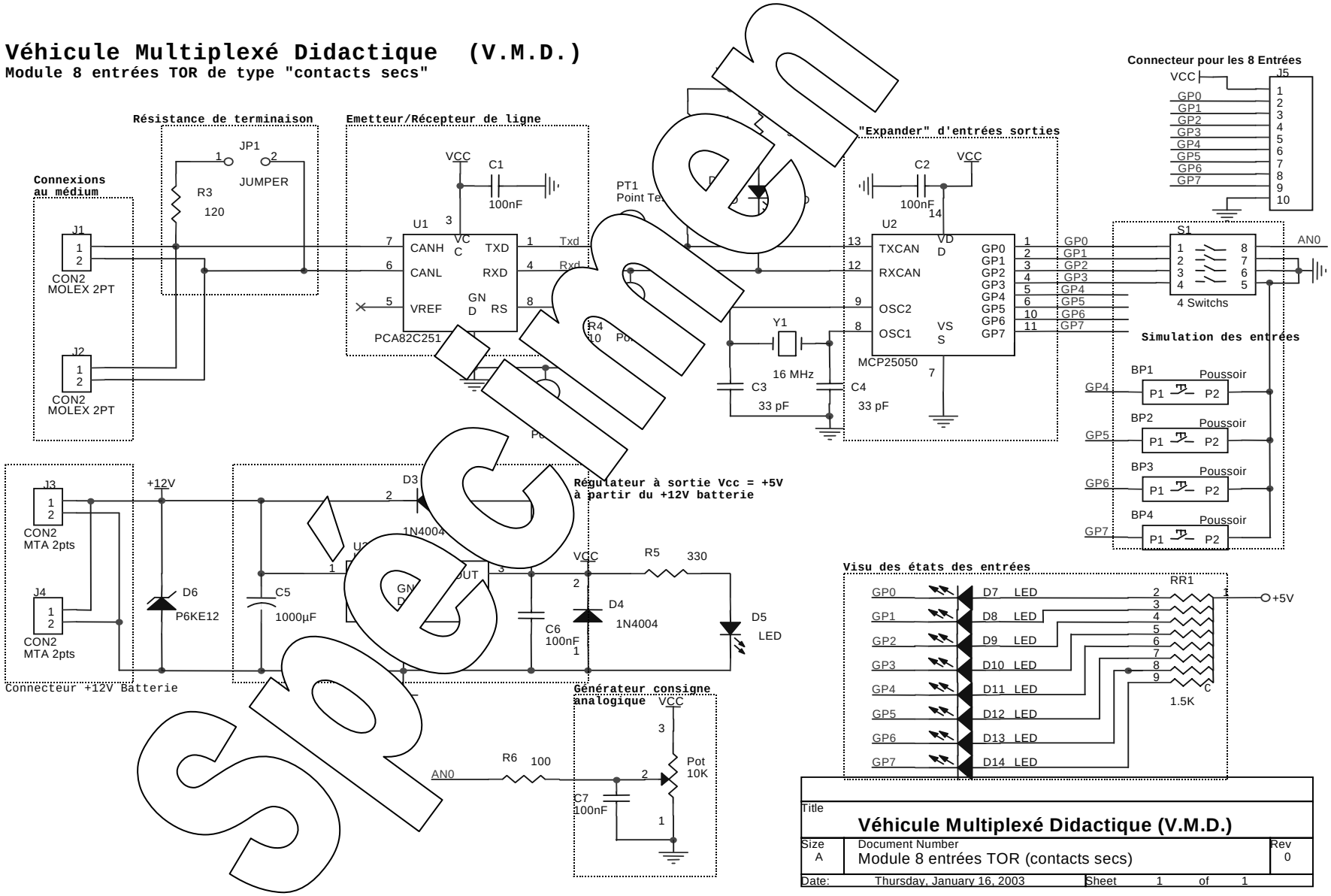
```
//  Déclaration des identificateurs, pour les différents modules sur le bus  VMD
//-------------------------------------------------------------------------------
// Pour Feux aVant Gauche
#define Ident_T_IRM_FVG 0x0E041E07   // Feux aVant Gauche en interrogation (Information Request Message)
#define Ident_T_IM_FVG 0x0E080000    // Feux aVant Gauche en commande (Input Message)
#define Ident_T_AIM_FVG 0x0E200000              // Feux aVant Gauche en Acquittement commande
// Pour Feux aVant Droit
#define Ident_T_IRM_FVD 0x0E841E07   // Feux aVant Droit en interrogation (Information Request Message)
#define Ident_T_IM_FVD 0x0E880000    // Feux aVant Droit en commande (Input Message)
#define Ident_T_AIM_FVD 0x0EA00000   // Feux aVant Droit en Acquittement commande
// Pour Feux aRierre Gauche
#define Ident_T_IRM_FRG 0x0F041E07   // Feux aRrière Gauche en interrogation (Information Request Message)
#define Ident_T_IM_FRG 0x0F080000    // Feux aRrière Gauche en commande (Input Message)
#define Ident_T_AIM_FRG 0x0F200000   // Feux aRrière Gauche en Acquittement commande
// Pour Feux aRierre Droit
#define Ident_T_IRM_FRD 0x0F841E07   // Feux aRrière Droit en interrogation (Information Request Message)
#define Ident_T_IM_FRD 0x0F880000    // Feux aRrière Droit en commande (Input Message)
#define Ident_T_AIM_FRD 0x0FA00000   // Feux aRrière Droit en Acquittement commande
// Pour Commodo Feux
#define Ident_T_IM_Commodo_Feux 0x05080000 // Commodo feux en commande (Infor      essage
#define Ident_T_AIM_Commodo_Feux 0x05200000 // Commodo feux : Acquittement suite
#define Ident_T_IRM_Commodo_Feux 0x05041E07 // Commodo feux en interrogation (Inform.      uest Me  age)
// Pour Essuie Glace
#define Ident_T_IM_Commodo_EG 0x05880000 // Commodo EG en commande      ssage
#define Ident_T_AIM_Commodo_EG 0x05A00000 // Commodo EG en comma      nform      Message)
#define Ident_T_IRM1_Commodo_EG 0x05841E07 // Commodo EG en interro      l      der      é
#define Ident_T_IRM8_Commodo_EG 0x05840000 // Commodo EG en interroga      ets d      lés
#define Ident_T_OB_Commodo_EG 0x05900000 // Commodo EG "On B      " (Par s
// Pour module Asservissement
#define Ident_T_IM_Asservissement 0x00880000       // "Asservissem      de (Input Message)
#define Ident_T_IRM1_Asservissement 0x00841E07  // "Asservis      n in      n: 1 octet en réponse
#define Ident_T_IRM8_Asservissement 0x00840000  // "Asserv      interr      octets en réponse
#define Ident_T_AIM_Asservissement 0x00A00000   // "Asserv      quittem      commande
#define Ident_T_OB_Asservissement 0x00900000    // "As      ment      s (par scéduleur)
// Pour le "Commodo Essuie Glace"
Port_8ES Commodo_EG;
#define Etat_Commodo_EG Commodo_EG.valeur
#define Valeur_Commodo_EG Commodo_EG.valeur
#define Cde_EG_Av_Int Commodo_EG.bit.GP0        // Co      Essuie      ce Avant en Intermittant
#define Entree1 Commodo_EG.bit.GP1
#define Entree2 Commodo_EG.bit.GP2
#define Cde_EG_Av_Pos1 Commodo_EG.b      // Comman      Essuie Glace Avant en Position1
#define Cde_EG_Av_Pos2 Commodo_EG.b      // Comm      ssuie Glace Avant en Position2
#define Cde_EG_Ar Comm      o_EG.bit.GP5          // Com      e Essuie Glace Arrière
#define Cde_Lave_Glace_A      ommodo_EG.bi
#define Cde_Lave_Glace_Av      om      it.G
#endif
```

6.2 Fichier de définition propre à la carte ATON

```
// ************************************************************
//  Fichier de définition pour carte controleur CAN "ATON_CAN"
//  Nom de fichier:  Aton_CAN.h
//************************************************************
#ifndef _PELICAN_H
#define _PELICAN_H
#define SJA             0xB30280                                /* Adresse de la carte SJA          */
#define MODE            *(unsigned char *)    (SJA)             /* Registre de controle */
#define COMMAND *(unsigned char *)  (SJA+0x01) /* Registre de commande          */
#define STATUS      *(unsigned char *)  (SJA+0x02) /* Registre status */
#define INTERRUPT               *(unsigned char *)  (SJA+0x03) /* Registre d'interruption */
#define INTERRUPT_ENABLE*(unsigned char *)  (SJA+0x04)
#define BUS_TIMING_0        *(unsigned char *)  (SJA+0x06) /* Registre bus timing 0          */
#define BUS_TIMING_1        *(unsigned char *)  (SJA+0x07) /* Registre bus timing 1
#define OUTPUT_CONTROL   *(unsigned char *)  (SJA+0x08) /* Registre output control
#define ARBITRATION_LOST_CAPTURE            *(unsigned char *) (SJA+0x0B)
#define ERROR_CODE_CAPTURE                  *(unsigned char *) (SJA+0x0C)
#define ERROR_WARNING_LIMIT         *(unsigned char *) (SJA+0x0D)
#define RX_ERROR_COUNTER                    *(unsigned char *) (SJA+0x0E)
#define TX_ERROR_COUNTER                    *(unsigned char *) (SJA+0x0
#define RX_FRAME_INFO               *(unsigned char *) (SJA+0x10)
#define TX_FRAME_INFO               *(unsigned char *) (SJA+0x10)

/* Mode SIMPLE FRAME */
#define RX_ID_1_S       *(unsigned char *) (SJA+0x11)
#define RX_ID_2_S       *(unsigned char *) (SJA+0x12)
#define RX_DATA_S         (unsigned char *) (SJA+0x13)      /* A                 ssage */
#define TX_ID_1_S       *(unsigned char *) (SJA+0x11)
#define TX_ID_2_S       *(unsigned char *) (SJA+0x12)
#define TX_DATA_S         (unsigned char *) (SJA+0x13)                 t mess        */

/* Mode EXTENDED */
#define RX_ID_1_E       *(unsigned char *) (S      0x1
#define RX_ID_2_E       *(unsigned char *) (SJA+0
#define RX_ID_3_E       *(unsigned char *) (SJA+0x
#define RX_ID_4_E       *(unsigned char *) (SJA+0x14)
#define RX_DATA_E         (unsigned char *) (      5)          sse debut message */

#define TX_ID_1_E       *(unsigned char *          (SJA+0x11)
#define TX_ID_2_E       *(unsigned char *          +0x12)
#define TX_ID_3_E       *(unsigned char *          0x13)
#define TX_ID_4_E       *(unsigned char *) (S
#define TX_DATA_E         (unsig                (SJA                /* Adresse debut message */

/* Les 2 modes */
#define RX_MESSAGE_COUN                 (unsi          char *) (SJA+0x1D)
#define RX_BUFFER_START_A               *(u          ed char *) (SJA+0x1E)
#define CLOCK_DIVID                         ) (SJA+0x1F)

/* acceptance code et                    et uniquement */
#define ACCEPT                          *(unsigned char *)    (SJA+0x10)
#defi CCEPT_C                           *(unsigned char *)    (SJA+0x11)
#d     PT_COD                           *(unsigned char *)    (SJA+0x12)
#     EPT_CODE3                         *(unsigned char *)    (SJA+0x13)

#                                       *(unsigned char *)    (SJA+0x14)
#d                                      (unsigned char *)    (SJA+0x15)
#defin      T_MASK                      *(unsigned char *)    (SJA+0x16)
#define ACCEPT_MASK                     *(unsigned char *)    (SJA+0x17)

/*** Configuratio          s des registres ***/
/* Registre STATUS */
#define BUS_STATUS              0x80
#define ERROR_STATUS            0x40
#define TRANSMIT_STATUS         0x20
#define RECEIVE_STATUS          0x10
#define TRANSMISSION_COMPLETE   0x08
#define TRANSMIT_BUFFER_STATUS  0x04
#define DATA_OVERRUN_STATUS     0x02
#define RECEIVE_BUFFER_STATUS   0x01
```

```
// ************** TYPES ********************
// ***************************************************************
// *                                DECLARATIONS
// *   des prototypes des fonctions spécifiques CAN - SJA1000
// ***************************************************************
/** Initialisation du SJA1000 en mode PeliCAN. */
void init_sja1000_peli_acc (char acc_code0, char acc_code1, char acc_code2, char acc_code3,
                    char acc_mask0, char acc_mask1, char acc_mask2, char acc_mask3);
/** Initialisation du SJA1000 en mode PeliCAN. */
void init_sja1000_peli ();
Trame Receive_Trame();
/** Envoi d'une trame */
void send_trame_peli (Message mes);
/** Reception d'un message. */
Message receive_trame_peli ();
/** Affichage des regsitres en PeliCAN */
void print_reg_peli ();
/* Affiche l'etat du STATUS REGISTER */
void show_status ();
/* Affiche le message passe en parametres sur une ligne */
void print_little (Message mes);
// initialisation de la carte CAN ATON
void Init_Aton_CAN();
char Ecrire_Trame(Trame message);
char Lire_Trame(Trame *message_recu);
void Affiche_Trame(Trame trame);
```

6.3 Schéma structurel de la carte 8 entrées



6.4 Schéma structurel de la carte "Asservissement"