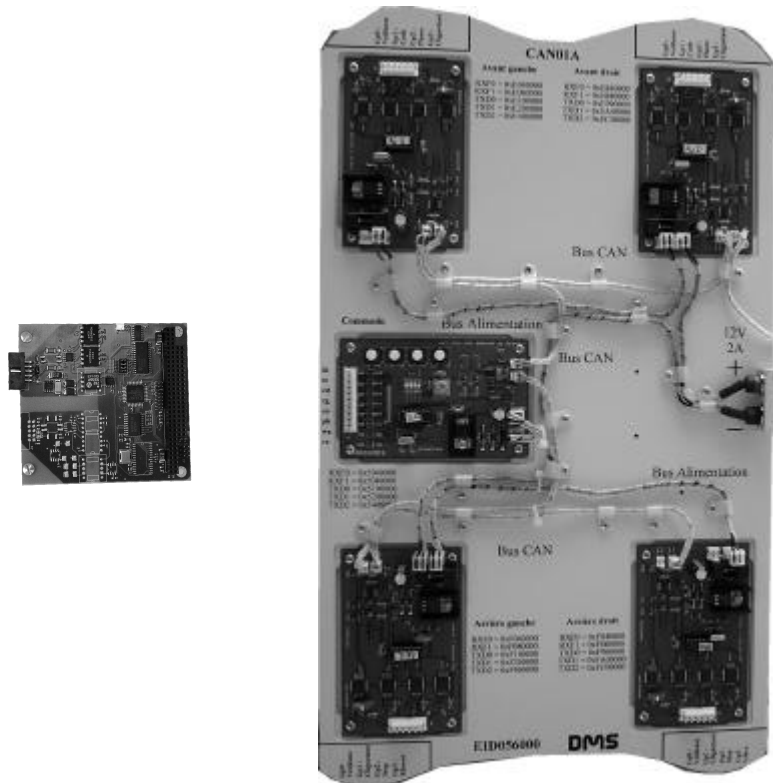


Systeme CAN 01A



Guide technique

SOMMAIRE

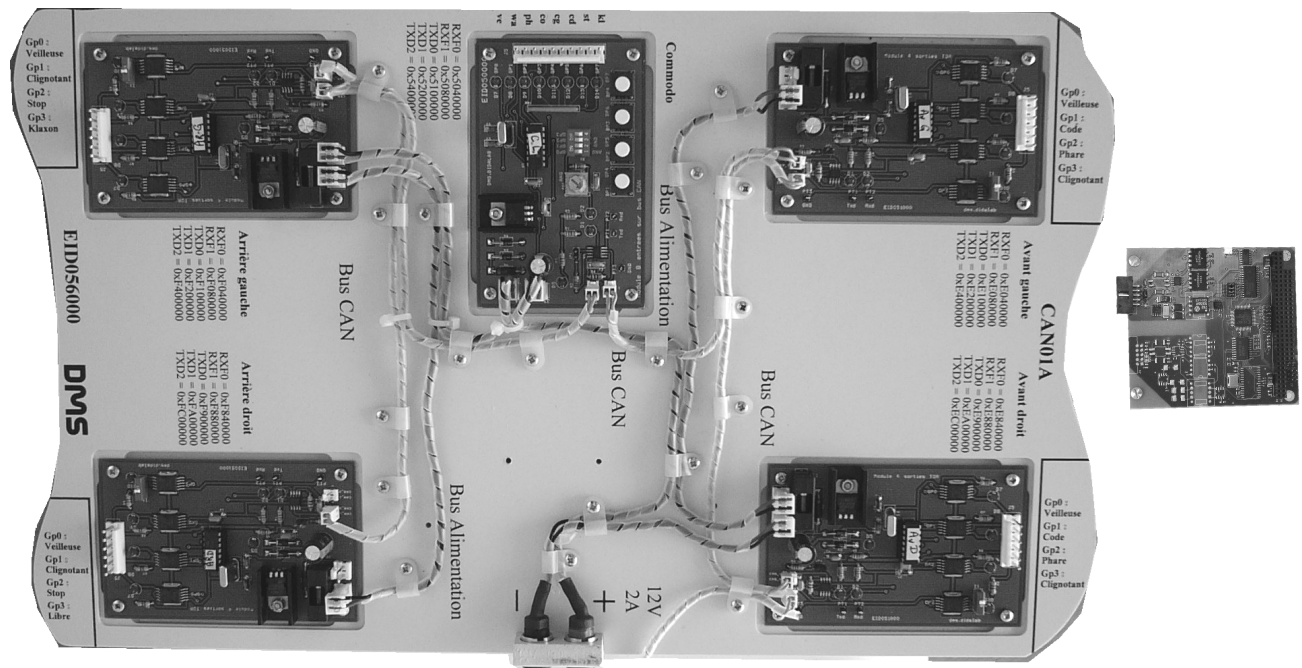
1	<i>Présentation.....</i>	5
2	<i>Installation</i>	7
2.1	Installation matériel.....	7
2.2	Installation logiciel.....	7
3	<i>Organisation logiciel.....</i>	9
3.1	Fichier de définition.....	9
3.2	Bibliothèque de fonctions	10
4	<i>Analyse fonctionnelle descendante</i>	11
4.1	Schéma organisationnel d'ensemble	11
4.2	L'unité centrale programmable	11
4.3	Les modules d'interface CAN configurables.....	12
4.3.1	Les fonctions communes	12
4.3.2	Module 8 entrées logiques.....	13
4.3.3	Module 4 sorties de puissance.....	13
5	<i>Identification des différents modules CAN.....</i>	15
6	<i>Description du MCP25050.....</i>	17
6.1	La configuration de la vitesse de transmission.....	18
6.2	Configuration des entrées/sorties.....	21
6.2.1	Carte 8 entrées (Commodo lumière).....	21
6.2.2	Carte 4 sorties TOR.....	21
7	<i>schémas de principe</i>	22

1 PRESENTATION

Le VMD (Véhicule Multiplexé Didactique) est un système didactique sur les RLI (Réseaux locaux industriels).

Le VMD utilise le bus CAN pour communiquer avec ses différents modules d'entrées et de sorties. Il a été développé en s'appuyant sur ce qui existe dans l'automobile. Nous avons reconstitué le bus signalisation d'un véhicule.

Le module CAN01A est un sous-système du VMD avec uniquement les cartes CAN du bus signalisation et la carte contrôleur CAN ATON_CAN sur bus PC104.



Dans le bus CAN, il n'y a que 2 couches normalisées dans le modèle OSI :

Couche physique (couche 1 OSI)

Couche liaison de donnée (couche 2 OSI).

Les autres couches ne font pas l'objet de normalisation dans le bus CAN.

Le VMD se compose :

D'un support thermoformé représentant un véhicule avec ses organes de signalisation,

D'une carte processeur EID210, représentant l'ordinateur de bord,

D'une carte 8 entrées TOR sur bus CAN gérant le commodo lumière,

De 4 cartes de sortie TOR, gérant :

Les feux avant gauche,

Les feux avant droit,

Les feux arrière gauche,

Les feux arrière droit ;

D'une carte clavier afficheur réalisant le tableau de bord.

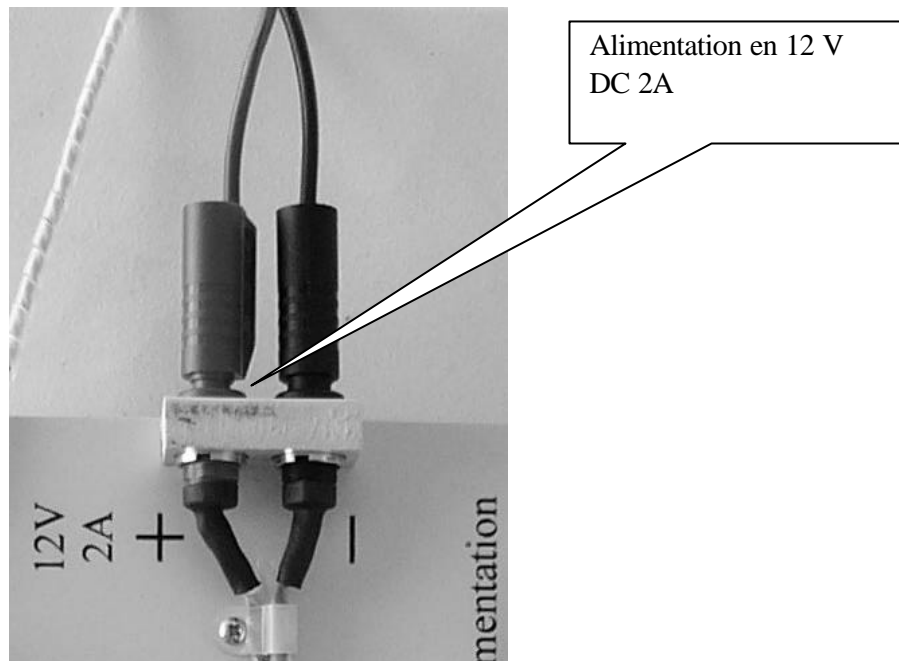
La carte EID210 accède au bus CAN par l'intermédiaire d'une carte PC104 CAN de ATON-SYSTEME à travers un contrôleur CAN SJA1000 de philips.

Les cartes CAN possèdent un contrôleur CAN MCP25050 de microchip

2 INSTALLATION

2.1 Installation matériel

Alimenter le CAN01A en +12 V DC



Raccorder la carte EID210 au PC par l'intermédiaire du cordon de liaison série ou USB.

2.2 Installation logiciel

Pour installer l'environnement de développement EID210, exécuter le programme « SETUP.EXE » depuis le CD. (sous windows 2000 et XP professionnel, il faut avoir les droits suffisants pour pouvoir installer le logiciel).

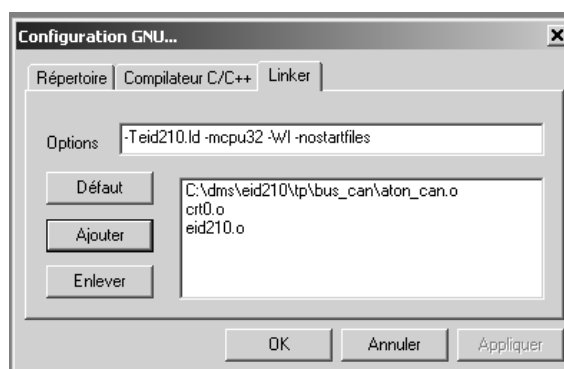
Le logiciel EID210, comprend

- Un éditeur,

- Un cross assembleur 68000,

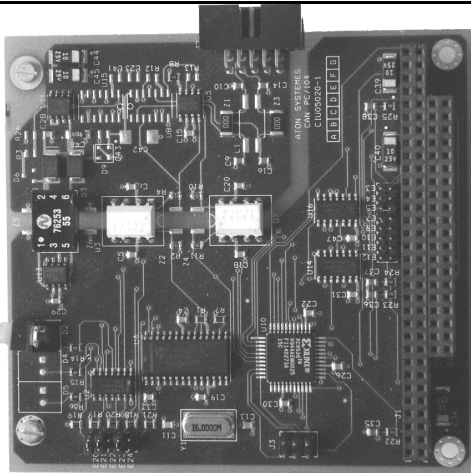
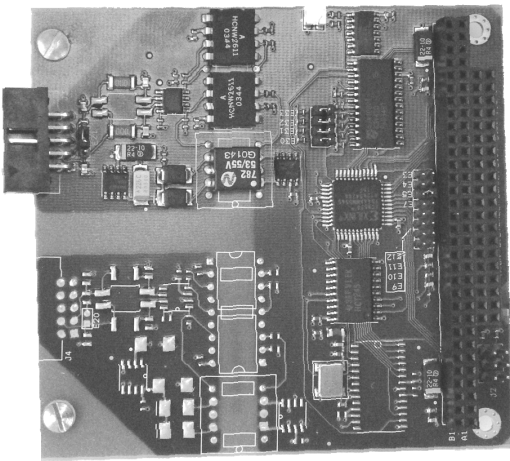
- Un cross compilateur c/c++ GNU C/C++

Toutes les bibliothèques de gestion du bus CAN ont été écrites en C.

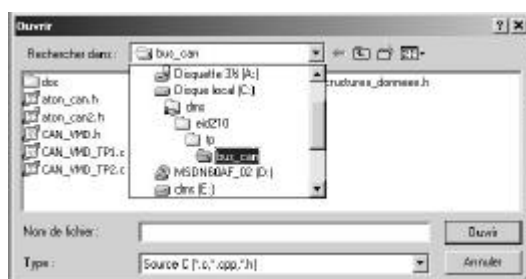


Pour configurer l'outil de développement, il faut lier la bibliothèque « aton_can.o »

Il existe 2 versions de la carte aton can. Il faut sélectionner le fichier de la bibliothèque en fonction de la version de la carte

Aton CAN version1	Aton CAN version 2
	
Aton_can.o Aton_can.h	Aton_can2.o Aton_can2.h

Les sources des différents travaux pratiques sur le bus CAN, sont sous le répertoire « EID210\tp\bus_can ».



3 ORGANISATION LOGICIEL

3.1 Fichier de définition

Pour utiliser le bus can, il faut utiliser le fichier de définition « aton_can.h ». Celui-ci définit les structures de données permettant la gestion des trame CAN.

L'union "ident_standard " permet de définir la partie identification en mode standard

```
typedef union
{
  struct
  {
    // Eléments constitutif de l'identificateur dans une trame
    unsigned short ident:11; // les 11 bits d'identification en mode standard
    unsigned short nul:5;    // 5 bits inutilisés
  } identificateur;
  struct
  {
    // Les mêmes éléments mais dans les registres du circuit SJA1000
    unsigned char ident1; // premier registre 8 bits de définition de l'identificateur
    unsigned char ident2; // deuxième registre 8 bits de définition de l'identificateur
  } registre;
  unsigned short valeur; // La taille globale est de 16 bits
} ident_standard;
```

L'union "ident_extend" permet de définir la partie identification en mode étendu

```
typedef union
{
  struct
  {
    // Eléments constitutif de l'identificateur dans une trame
    unsigned long ident:29; // les 29 bits d'identification en mode étendu
    unsigned long x:3;      // 3 bits inutilisés
  } identificateur;
  struct
  {
    // Les mêmes éléments mais dans les registres du circuit SJA1000
    unsigned char ident1; // premier registre 8 bits de définition de l'identificateur
    unsigned char ident2; // deuxième registre 8 bits de définition de l'identificateur
    unsigned char ident3; // troisième registre 8 bits de définition de l'identificateur
    unsigned char ident4; // quatrième registre 8 bits de définition de l'identificateur
  } registre;
  unsigned long valeur; // La taille globale est de 32 bits
} ident_extend;
```

La structure "Trame" permet de définir une trame complète

```
typedef struct
{
  tr_info trame_info; // Taille 8 bits
  union
  {
    {
      ident_standard standard; // Identificateur en mode standard (2*8bits LSB)
      ident_extend extend;     // Identificateur en mode étendu (4*8bits)
    } ident;
  }
  unsigned char data[8]; // les 8 octets de données (au maximum)
} Trame; // Taille globale: 13 octets
```

3.2 Bibliothèque de fonctions

Pour gérer le VMD, la bibliothèque « aton_can.o » permet la gestion du bus can. Elle est composée des fonctions suivantes :

Void Init_Aton_Can()

- Initialise le contrôleur de bus CAN SJA1000 pour la gestion du VMD :
- Configure la vitesse à 100 Kbits/S,
- Met en place les FIFOs d'émission et de réception en interruption (autovecteur 29)

Void Stop_Aton_Can()

- Stop le module de gestion du bus can (libère le vecteur d'interruption)

void Ecrire_Frame(Frame frame)

- Envoi une trame sur le bus CAN.

Char Lire_Frame(Frame *frame)

- Permet de lire une trame reçue par le contrôleur CAN.
- La fonction renvoi 1 si le contrôleur a reçu une trame.

Void Affiche_Frame(Frame frame)

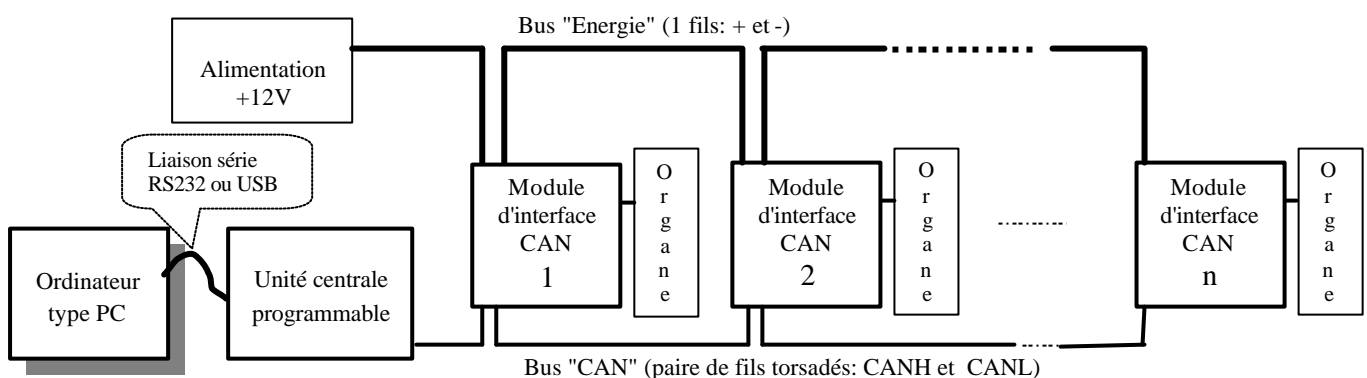
- Affiche sur le port terminal la trame CAN avec son identifiant et ses données éventuelles.

4 ANALYSE FONCTIONNELLE DESCENDANTE

4.1 Schéma organisationnel d'ensemble

Le système comporte les éléments suivants :

- un ensemble "unité centrale" programmable en liaison avec un ordinateur de type PC,
- une source d'énergie (batterie 12V ou alimentation non autonome générant du 12V sous 20 A,
- un certain nombre d'ensembles, chacun pouvant être constitué :
 - * d'un module d'interface CAN configurable,
 - * d'un organe de type capteur, pré-actionneur voir actionneur, compatible avec le module associé,
- câbles de liaison.



Remarques :

- Des éléments de simulation d'entrées/sorties (LEDS, boutons poussoirs, commutateurs) ont été intégrés sur les modules et permettent éventuellement de s'affranchir des organes réels.
- Dans sa version "VMD" (Véhicule Multiplexé Didactique) les organes reliés aux modules d'interface CAN peuvent être un commodo, un bloc optique avant, un bloc optique arrière, un moteur d'essuie glace ou de lève vitre, plafonnier ... etc.

4.2 L'unité centrale programmable

Elle comprend un certain nombre de cartes électroniques reliées entre elles par BUS parallèles :

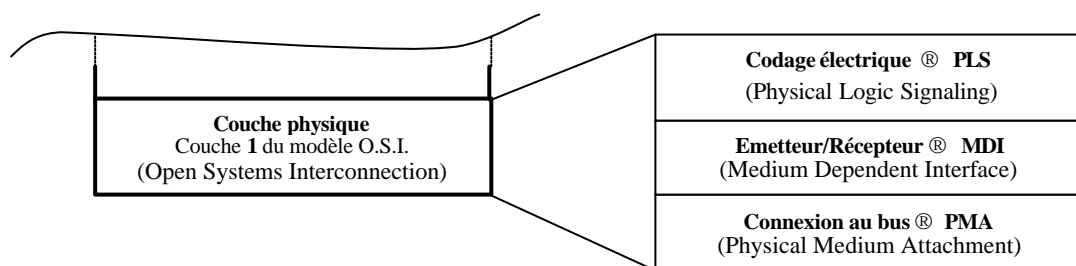
- cartes indispensables
 - Carte processeur EID210 conçue autour du microprocesseur 32 bits Motorola 68332 (carte spécifique DMS),
 - Carte industrielle d'interface parallèle / CAN conçue autour du circuit SJA1000, au format normalisé "PC104" (carte produite par la société ATON SYSTEMES),
- cartes optionnelles
 - Carte clavier 16 touches matricé et afficheur graphique (carte spécifique DMS),
 - Carte d'interface réseau "Ethernet" (carte spécifique DMS),
 - Toute carte au format PC104.

4.3 Les modules d'interface CAN configurables

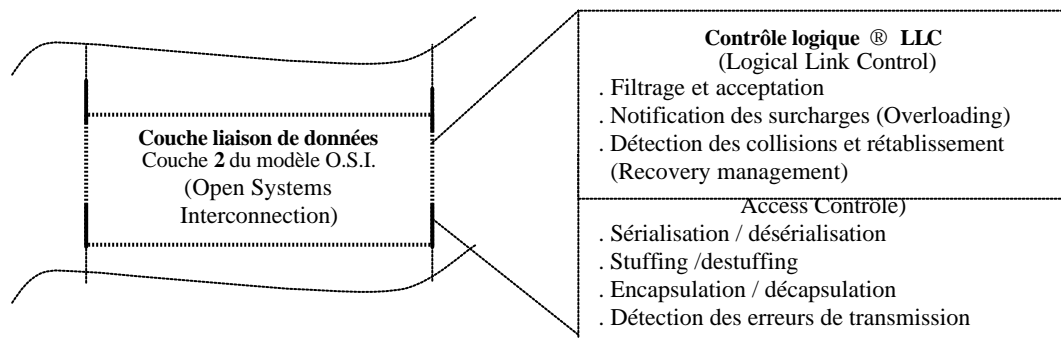
4.3.1 Les fonctions communes

Tous ces modules comportent les éléments suivants :

- deux connecteurs de liaison au médium (fils de liaisons bus CAN) permettant de connecter les modules en série,
- deux connecteurs d'alimentation (bus "alimentation énergie"),
- un régulateur de tension générant la tension +5V nécessaire aux circuits intégrés inclus sur le module ainsi que les protections d'usage et une LED de visualisation de présence tension,
- une résistance de terminaison de bus connectable ou non par "jumper",
- un circuit intégré "émetteur récepteur de ligne" (réf:82CA251) réalisant l'interface électrique (mode différentiel côté bus CAN et mode référencé côté application), c'est à dire la couche 1 du modèle O.S.I.



- un circuit intégré d'extension d'entrées sorties (Réf:MPC25050) permettant la communication et l'interprétation des messages, c'est à dire la couche 2 du modèle O.S.I.



- un oscillateur à quartz nécessaire au circuit MPC25050,
- une LED de visualisation de communication en réception,
- une LED de visualisation de communication en émission.

4.3.2 Module 8 entrées logiques

Le schéma de principe complet de ce module est donné en annexe.

Ce module permet d'interfacer 8 entrées/sorties logiques.

En plus des éléments communs, ce module comprend :

- 4 boutons poussoir,
- 4 commutateurs à 2 positions (fermé ou ouvert),
- 1 connecteur 10 points permettant de relier les capteurs externes de type fin de course,
- 8 LEDs de visualisation des états,
- 1 potentiomètre analogique.

Remarques :

- Si un capteur externe de type fin de course est relié au connecteur, le commutateur (ou bouton poussoir) correspondant doit resté en position "ouvert" (sinon il courtcircuite le fin de course).
 - Une LED s'allume si l'entrée correspondante est forcée à 0 (commutateur en position "fermé" ou bouton poussoir "appuyé").
 - Ce module peut être utilisé comme module comportant des sorties logiques compatible "TTL". En effet le circuit MPC25050 est configurable. On peut donc, via le bus CAN, lui envoyer une trame qui définira si telle ou telle liaison est une entrée ou une sortie. On pourra donc envisager toute combinaison pourvu que la somme des entrées et des sorties ne dépasse pas 8.
- Dans le cas d'une liaison configurée en sortie, il est impératif que le commutateur correspondant soit à l'état ouvert.***
- Le potentiomètre n'est actif que si le commutateur "0" est fermé.

4.3.3 Module 4 sorties de puissance

Le schéma de principe de ce module est donné en annexe.

Ce module comporte:

- 4 interfaces de puissance permettant de piloter 4 charges électriques en "tout ou rien", sous 12V
- 4 Leds de visualisation des états des sorties puissance,
- 4 entrées de contrôle des charges,
- 1 entrée de simulation de coupure de charge.

Remarques :

- Le circuit intégré réalisant l'interface de puissance (Réf: VN05) génère un signal logique indiquant si une charge est connectée (contrôle du courant absorbé). Ces signaux logiques sont considérés comme entrée du système et permettent, dans le cas du VMD et pour la commande d'une ampoule, de contrôler le bon état de fonctionnement de celle-ci.
- Les sorties de puissance sont de type source, c'est à dire que les 4 charges auront pour point commun, la référence de potentiel (dans le cas d'un véhicule, le "-" de la batterie relié à la carcasse).
- Les circuits de puissance "VN05" peuvent accepter des charges électriques consommant jusqu'à 12 A en continu. Ils sont protégés contre les court-circuits ainsi que contre les dépassements de température.

5 IDENTIFICATION DES DIFFERENTS MODULES CAN

Registre MCP25025 Et fonction	- SIDH (en bin)	- SIDL (en bin)	SIDH (Hex)	SIDL (Hex)	Identificateur (Hex) (! sur 29 bits)	Labels définis dans fichier CAN_VMD.h
-------------------------------------	--------------------	--------------------	---------------	---------------	---	---

Nœud "Commodo feux"

RXF0 -> IRM et OM	001 0 10 00	001 - 1-xx	28	28	0504 xx xx	T_Ident_IRM_Commodo_Feux
RXF1 -> IM	001 0 10 00	010 - 1-xx	28	48	0508 xx xx	T_Ident_IM_Commodo_Feux
TXD0 -> On Bus	001 0 10 00	100 - 1-xx	28	88	0510 xx xx	
TXD1 -> Acq IM	001 0 10 01	000 - 1-xx	29	08	0520 xx xx	T_Ident_AIM_Commodo_Feux
TXD2 -> Mes. Auto.	001 0 10 10	000 - 1-xx	2A	08	0540 xx xx	

Nœud "Feux avant gauche"

RXF0 -> IRM et OM	011 1 00 00	001 - 1-xx	70	28	0E04 xx xx	T_Ident_IRM_FVG
RXF1 -> IM	011 1 00 00	010 - 1-xx	70	48	0E08 xx xx	T_Ident_IM_FVG
TXD0 -> On Bus	011 1 00 00	100 - 1-xx	70	88	0E10 xx xx	
TXD1 -> Acq IM	011 1 00 01	000 - 1-xx	71	08	0E20 xx xx	T_Ident_AIM_FVG
TXD2 -> Mes. Auto.	011 1 00 10	000 - 1-xx	72	08	0E40 xx xx	

Nœud "Feux avant droit"

RXF0 -> IRM et OM	011 1 01 00	001 - 1-xx	74	28	0E84 xx xx	T_Ident_IRM_FVD
RXF1 -> IM	011 1 01 00	010 - 1-xx	74	48	0E88 xx xx	T_Ident_IM_FVD
TXD0 -> On Bus	011 1 01 00	100 - 1-xx	74	88	0E90 xx xx	
TXD1 -> Acq IM	011 1 01 01	000 - 1-xx	75	08	0EA0 xx xx	T_Ident_AIM_FVD
TXD2 -> Mes. Auto.	011 1 01 10	000 - 1-xx	76	08	0EC0 xx xx	

Nœud "Feux arrière gauche"

RXF0 -> IRM et OM	011 1 10 00	001 - 1-xx	78	28	0F04 xx xx	T_Ident_IRM_FRG
RXF1 -> IM	011 1 10 00	010 - 1-xx	78	48	0F08 xx xx	T_Ident_IM_FRG
TXD0 -> On Bus	011 1 10 00	100 - 1-xx	78	88	0F10 xx xx	
TXD1 -> Acq IM	011 1 10 01	000 - 1-xx	79	08	0F20 xx xx	T_Ident_AIM_FRG
TXD2 -> Mes. Auto.	011 1 10 10	000 - 1-xx	7A	08	0F40 xx xx	

Nœud "Feux arrière droit"

RXF0 -> IRM et OM	011 1 11 00	001 - 1-xx	7C	28	0F84 xx xx	T_Ident_IRM_FRD
RXF1 -> IM	011 1 11 00	010 - 1-xx	7C	48	0F88 xx xx	T_Ident_IM_FRD
TXD0 -> On Bus	011 1 11 00	100 - 1-xx	7C	88	0F90 xx xx	
TXD1 -> Acq IM	011 1 11 01	000 - 1-xx	7B	08	0FA0 xx xx	T_Ident_AIM_FRD
TXD2 -> Mes. Auto.	011 1 11 10	000 - 1-xx	7E	08	0FC0 xx xx	

6 DESCRIPTION DU MCP25050

Pour plus de renseignement relatif au mcp25050, se reporter à la datasheet du MCP25050 de MICROCHIP.

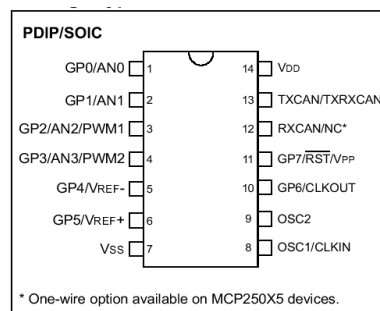
C'est un circuit intégré 14 broches disponible en version PDIP ou SOIC.

Côté interface CAN il satisfait la norme CAN V2.0B c'est à dire qu'il peut communiquer avec une vitesse de transmission qui peut atteindre 1Mbit/s. Certaines versions du circuit permettent une communication sur 1 fil (One-wire).

Côté application, il possède 8 lignes d'entrées/sorties (GP0 à GP7) configurables individuellement en entrée ou en sortie. Seule la ligne GP7 ne peut être utilisée en sortie.

Si on le souhaite, il est capable d'envoyer un message sans qu'il soit interrogé si l'une de ses entrées change d'état.

Deux liaisons (GP2 et GP3) peuvent être configurées en sorties modulées (PWM). Ces deux sorties peuvent



Device	A/D	One-wire CAN
MCP25020	No	No
MCP25025	No	Yes
MCP25050	Yes	No
MCP25055	Yes	Yes

être commandées indépendamment l'une de l'autre, fréquences et rapports cycliques sur 10 bits.

Certaines versions du circuit intègrent un convertisseur analogique → numérique (4 voies) sur 10 bits.

Si on le souhaite, il est capable d'envoyer un message sans qu'il soit interrogé si l'une de ses entrées analogique dépasse des seuils de tension que l'on peut choisir.

Il possède un "scheduler" qui lui permet d'envoyer un message à intervalles de temps réguliers sans qu'on le lui demande (par exemple l'état de ses entrées ou la valeur convertie d'une des entrées analogiques).

C'est un circuit configurable grâce à une banque de registres qui sont gravés dans le circuit lors d'une phase de programmation.

Remarque :

- Ces circuits ne sont pas reprogrammables.
- Le constructeur (MICROCHIP) commercialise des outils logiciels et matériel de configuration et de programmation.
- Un circuit déjà programmé peut être lu par le logiciel de configuration et de programmation.

6.1 La configuration de la vitesse de transmission

Cette vitesse dépend de la fréquence du signal d'horloge interne (imposée par le quartz) notée t_{osc} . Cette fréquence est divisée (passage dans un « perscaler ») pour obtenir la période t_Q (Time Quantum). Le coefficient de division BRP est choisi grâce à un mot de 6 bits BRP5 ... BRP0 (Ces bits font partie du registre de configuration n°1 CNF1 du circuit).

On obtient la valeur de t_Q grâce à l'expression :

$$t_Q = 2 * t_{osc} * (BRP + 1)$$

La durée de transmission d'un bit (Bit Time) est fonction de 3 paramètres :

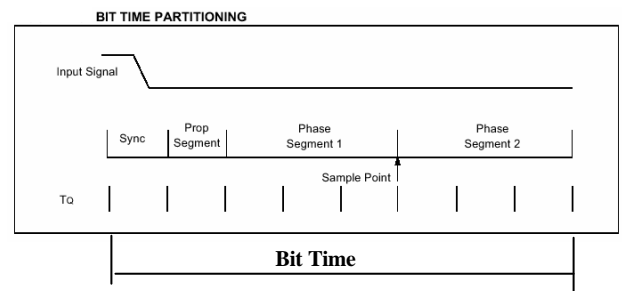
- un paramètre de synchronisation « SYN_Seg »
-> sur deux bits (SJW1 et SJW0) inclus dans le registre de configuration n°1
- un paramètre de propagation « Prop Segment »
-> sur 3 bits PRSEG2, PRSEG1 et PRSEG0 dans le registre de configuration n° 2
- un premier paramètre de phase « Phase Segment 1 »
-> sur 3 bits PHSEG12, PHSEG11 et PHSEG10 dans le registre de configuration n° 2
- un deuxième paramètre de phase « Phase Segment »
-> sur 3 bits PHSEG22, PHSEG21 et PHSEG20 dans le registre de configuration n° 2

Les paramètres de phase permettent de définir l'instant d'échantillonnage (« Sample time»), instant où l'on prend la valeur logique du bus ce qui va donner l'état du bit.

On obtient la valeur de t_{BIT} (durée de transmission d'un bit) grâce à l'expression :

$$t_{BIT} = t_Q * (S_Seg + P_Seg + PH_Seg1 + PH_Seg2)$$

La vitesse de transmission est l'inverse de t_{BIT}



La répartition des paramètres dans les registres est donnée ci-contre.

(Voir « DATA SHEET » du circuit MCP25050 page 8 et 9)

Exemple de calcul de la vitesse de transmission:

La fréquence du quartz sur la carte est 16 Mhz

On a choisit BRP = 4

$$\rightarrow t_Q = 2 * (4 + 1) * t_{osc} = 10 / 16 \mu s = 0.625 \mu s$$

On a choisit :

SJW = 0, PRSEG = 0

PHSEG1 = 8 et PHSEG2 = 8

D'après « DATA SHEET »

$$\rightarrow t_{BIT} = 16 * t_Q$$

En définitive :

$$t_{bit} = 16 * t_{scl} = 16 * 0.625 \mu s = 10 \mu s$$

Soit une vitesse de $1 / t_{bit} = 100 \text{ Kbit/s}$

CNF1 - CAN CONFIGURATION REGISTER 1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0
bit7							bit0

CNF2 - CAN CONFIGURATION REGISTER 2

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BTLMODE	SAM	PHSEG12	PHSEG11	PHSEG10	PRSEG2	PRSEG1	PRSEG0
bit7							bit0

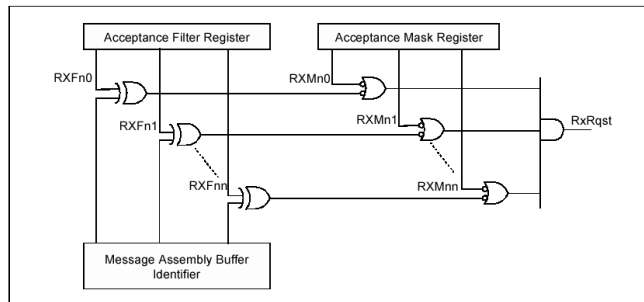
CNF3 - CAN CONFIGURATION REGISTER 3

U-0	R/W-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
—	WAKFIL	—	—	—	PHSEG22	PHSEG21	PHSEG20
bit7							bit0

La configuration des masques et des filtres "d'acceptance"

Un message circulant sur le bus ne sera pris en considération par le circuit que si l'identificateur associé au message a passé avec succès les obstacles du filtre et du masque.

Cette technique peut être déduite du schéma logique donné ci-après, ainsi que de la table de vérité.



FILTER/MASK TRUTH TABLE

Mask Bit n	Filter Bit n	Message Identifier bit n001	Accept or reject bit n
0	X	X	Accept
1	0	0	Accept
1	0	1	Reject
1	1	0	Reject
1	1	1	Accept

Note: X = don't care

En résumé, si on donne à un bit de masque l'état 1 logique un message ne sera accepté que si le bit correspondant de l'identificateur associé est du même état que celui donné au filtre.

Par contre, si on donne à un bit de masque l'état 0 logique, le bit correspondant de l'identificateur est masqué c'est à dire que son état n'est pas vérifié.

La définition des masques et des filtres s'effectue grâce à un certain nombre de registres destiné à cet effet (voir tableau ci-après).

Rappel :

Dans la version "standard" du bus CAN (version V2.0A) l'identificateur est sur 11 bits. Dans ce cas les bits de l'identificateur sont repérés **SID10 ... SID0**.

Dans la version "étendue" du bus CAN (version V2.0B) l'identificateur est sur 29 bits. Dans ce cas les bits de l'identificateur sont repérés **EID17 ... EID0** (ce sont les poids faibles de l'identificateur, les poids forts étant l'identificateur standard).

Il y a deux séries de registres dans lesquels on charge les valeurs des masques et des filtres :

- la série d'indice 0 destinée aux trames interrogatives ("Information Request"),
- la série d'indice 1 destinée aux entrées de données ("Input message").

Les registres définissant les masques									
RXMSIDH - ACCEPTANCE FILTER MASK STANDARD IDENTIFIER HIGH									
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	
SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3*		
bit7								bit0	
<div style="float: right; font-size: small;"> R = Readable bit W = Writable bit U = Unimplemented, read as '0' </div>									
RXMSIDL - ACCEPTANCE FILTER MASK STANDARD IDENTIFIER LOW									
R/W-x	R/W-x	R/W-x	U-0	R/W-x	U-0	R/W-x	R/W-x		
SID2	SID1	SID0	—	EXIDE	—	EID17	EID16		
bit7							bit0		
RXMEID8 - ACCEPTANCE FILTER MASK EXTENDED IDENTIFIER MID									
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x		
EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8		
bit7							bit0		
RXMEID0 - ACCEPTANCE FILTER MASK EXTENDED IDENTIFIER LOW									
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x		
EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0		
bit7							bit0		

D'après "DATA SHEET" du circuit pages 12, 13 et 14

Les registres définissant les filtres									
RXFNSIDH - ACCEPTANCE FILTER N STANDARD IDENTIFIER HIGH									
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x		
SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3*		
bit7							bit0		
<div style="float: right; font-size: small;"> R = Readable bit W = Writable bit U = Unimplemented, read as '0' </div>									
RXFNSIDL - ACCEPTANCE FILTER N STANDARD IDENTIFIER LOW									
R/W-x	R/W-x	R/W-x	U-0	R/W-x	U-0	R/W-x	R/W-x		
SID2	SID1	SID0	—	EXIDE	—	EID17	EID16		
bit7							bit0		
RXFNEID8 - ACCEPTANCE FILTER N EXTENDED IDENTIFIER MID									
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x		
EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8		
bit7							bit0		
RXFNEID0 - ACCEPTANCE FILTER N EXTENDED IDENTIFIER LOW									
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x		
EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0		
bit7							bit0		

Les différents messages

Pour communiquer avec le circuit via le bus CAN, on distingue 3 types de messages :

- messages de demande d'information (Information Request Messages) → IRM qui permettent de demander au circuit de renvoyer la valeur d'un ou plusieurs de ses registres internes (par exemple pour lire l'état de ses entrées),
- messages d'entrée (Input Messages) qui permettent de modifier la valeur d'un ou plusieurs registres internes (par exemple pour modifier l'état de ses sorties),
- messages de sortie (Output messages) qui permettent de répondre à un message d'interrogation.

Interrogations sur la valeur des registres	
Name	Description
Read A/D Registers	Transmits a single message containing the current state of the analog and I/O registers including the configuration
Read Control Registers	Transmits several control registers not included in other messages
Read Configuration Registers	Transmits the contents of many of the configuration registers
Read CAN error states	Transmits the error flag register and the error counts
Read PWM Configuration	Transmits the registers associated with the PWM modules
Read User Registers 1	Transmits a the values in bytes 0 - 7 of the user memory
Read User Registers 2	Transmits a the values in bytes 8 -15 of the user memory
Read Register*	Transmits a single byte containing the value in an addressed user memory register

Pour changer la valeur des registres	
Write Register	Uses a mask to write a value to an addressed register
Write TX Message ID0 (TXID0)	Writes the identifiers to a specified value
Write TX Message ID1 (TXID1)	Writes the identifiers to a specified value
Write TX Message ID2 (TXID2)	Writes the identifiers to a specified value
Write I/O Configuration Registers	Writes specified values to the three IOCON registers
Write RX Mask	Changes the receive mask to the specified value
Write RX Filter0	Changes the specified filter to the specified value
Write RX Filter1	Changes the specified filter to the specified value
*The Read Register command is available when using extended message format only. Not available with standard message format.	

Se reporter à la "DATA SHEET" du circuit (pages 21 et 22) pour connaître la constitution des différents messages.

6.2 Configuration des entrées/sorties

6.2.1 Carte 8 entrées (Commodo lumière)

Le port 8 bits du can expander MCP25050 est configuré en entrée TOR.

GP7	GP6	GP5	GP4	GP3	GP2	GP1	GP0
E	E	E	E	E	E	E	E

Avec E: entrée TOR.

L'affectation des entrées sur la carte entrée est la suivante :

GP7	GP6	GP5	GP4	GP3	GP2	GP1	GP0
klaxon	stop	Clignotant droit	Clignotant gauche	code	phare	warning	Veilleuse

6.2.2 Carte 4 sorties TOR

Le port 8 bit du can expander MCP25050 est configuré :

GP7	GP6	GP5	GP4	GP3	GP2	GP1	GP0
E	E	E	E	S	S	S	S

Avec : E: entrée TOR,
S : sortie TOR

6.2.2.1 Feux avant

L'affectation des entrées sur la carte entrée est la suivante :

GP7	GP6	GP5	GP4	GP3	GP2	GP1	GP0
Etat clignotant	Etat phare	Etat code	Etat veilleuse	clignotant	phare	code	Veilleuse

6.2.2.2 Feux arrières

L'affectation des entrées sur la carte entrée est la suivante :

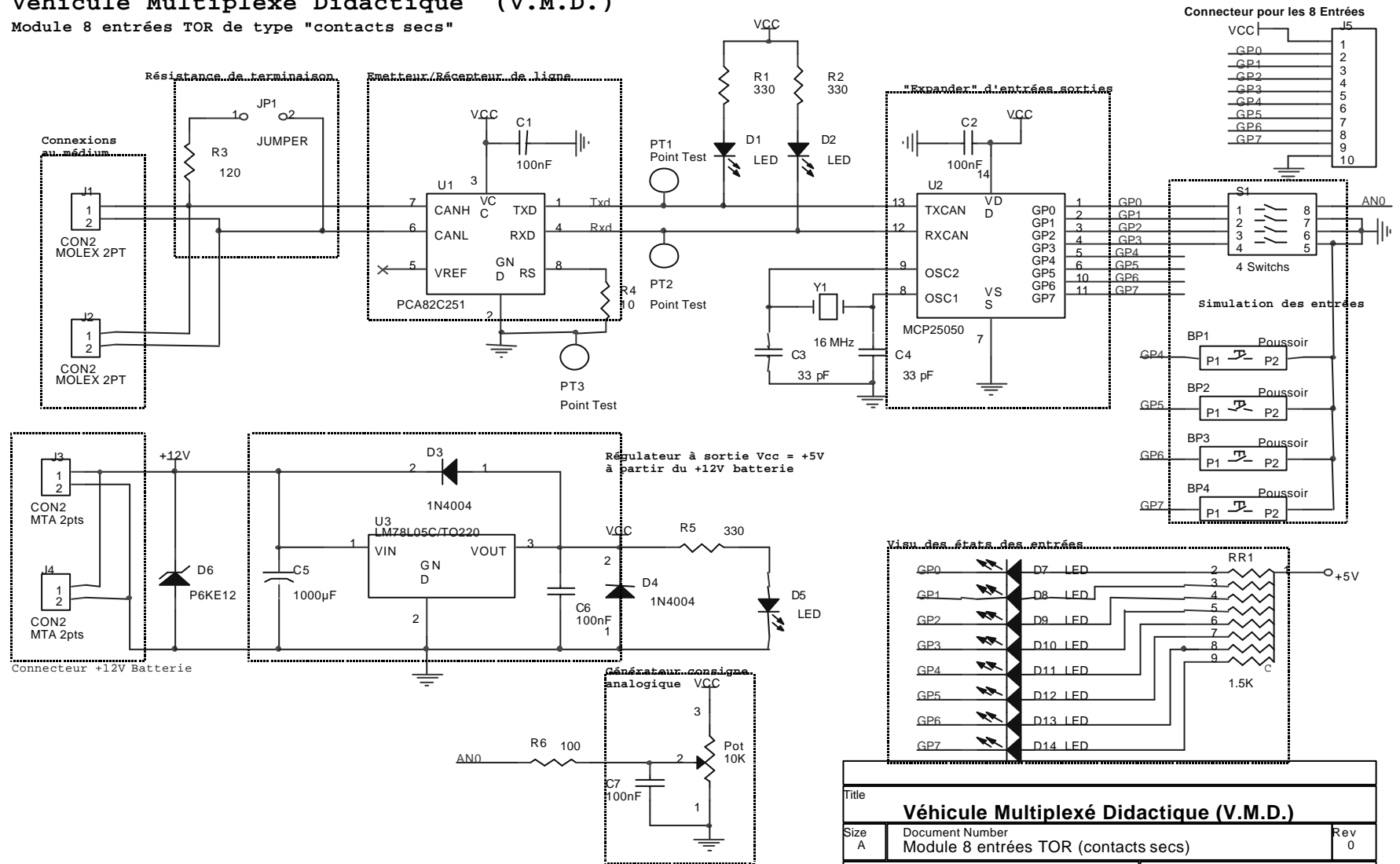
GP7	GP6	GP5	GP4	GP3	GP2	GP1	GP0
Etat GP3	Etat clignotant	Etat code	Etat veilleuse	(klaxon)	clignotant	code	Veilleuse

Remarques : la commande klaxon est active sur le module feux arrières gauche.

7 SCHEMAS DE PRINCIPE

Véhicule Multiplexé Didactique (V.M.D.)

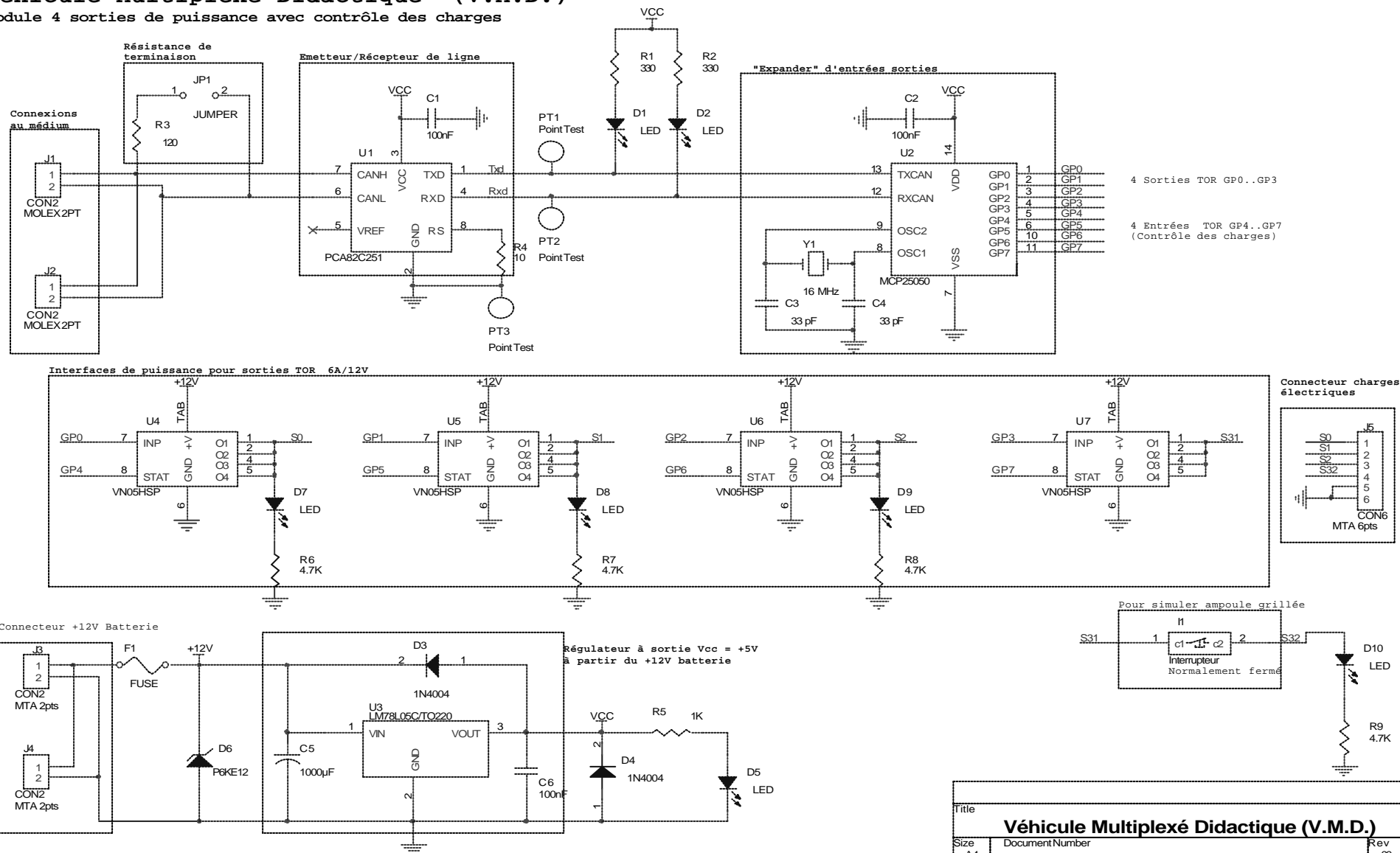
Module 8 entrées TOR de type "contacts secs"



Title		
Véhicule Multiplexé Didactique (V.M.D.)		
Size	Document Number	Rev
A	Module 8 entrées TOR (contacts secs)	0
Date:	Thursday, January 16, 2003	Sheet 1 of 1

Véhicule Multiplexé Didactique (V.M.D.)

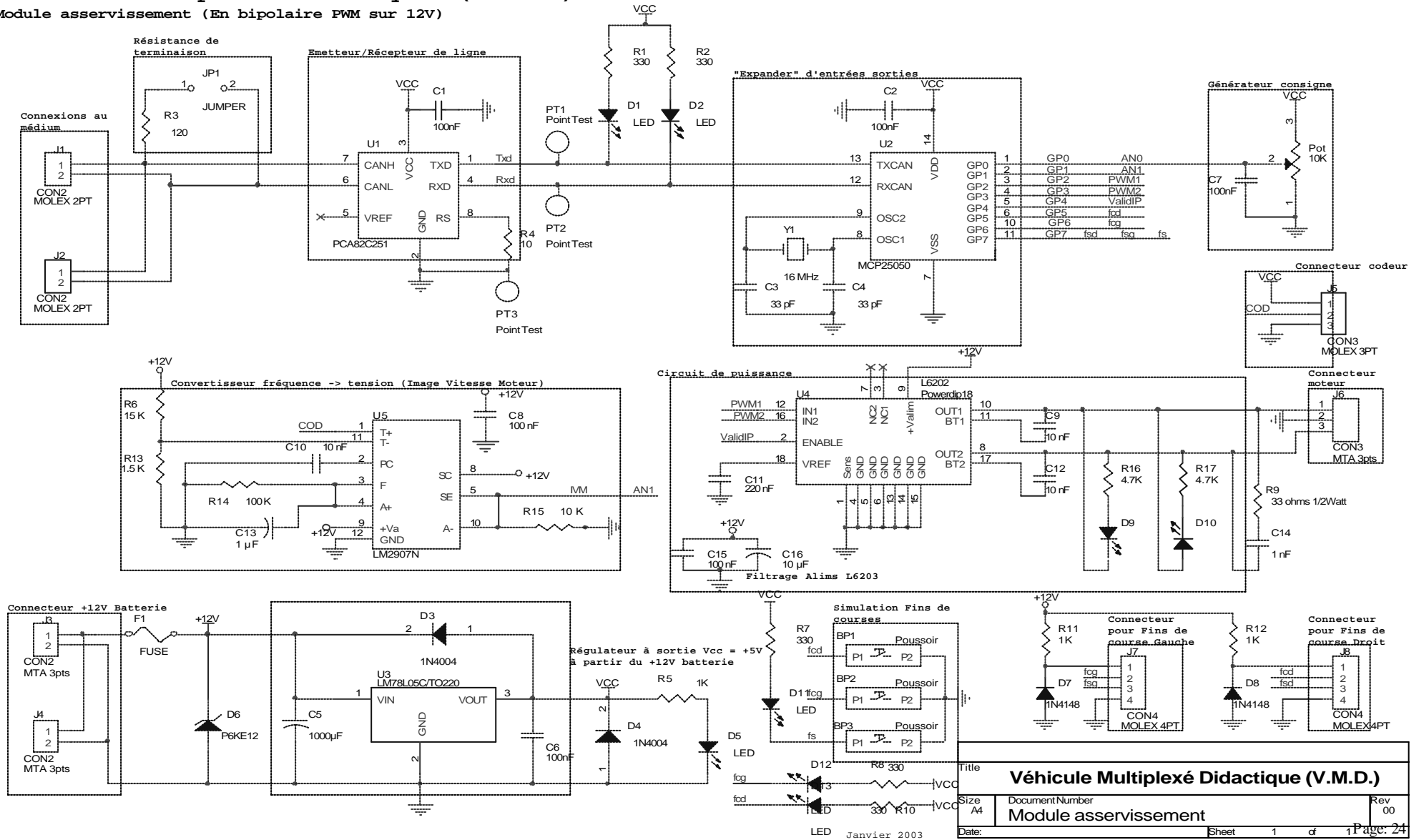
Module 4 sorties de puissance avec contrôle des charges



Title		
Véhicule Multiplexé Didactique (V.M.D.)		
Size	Document Number	Rev
A4	Module 4 sorties TOR	00
Date:	Sheet	1 of 1

Véhicule Multiplexé Didactique (V.M.D.)

Module asservissement (En bipolaire PWM sur 12V)



Véhicule Multiplexé Didactique (V.M.D.)

Document Number
Module asservissement

Rev 00

Date: _____ Sheet 1 of 1 Page: 24