

L'objectif de ce projet est de commander le modèle **CAN01A** et **CAN VMD** à l'aide de la carte EID210, en utilisant la carte ATON CAN comme interface de communication. Pour cela, des trames CAN spécifiques sont envoyées afin de configurer et contrôler les différents registres nécessaires au fonctionnement du modèle CAN01A ou CAN VMD. Ces trames permettent de gérer les entrées/sorties et de définir l'état des périphériques connectés, tout en exploitant le protocole CAN pour garantir une communication fiable et efficace entre les différentes cartes.

Sommaire

1. Carte EID210 000	3
1.1 Fonctions principales :.....	3
1.2 Ressources matérielles :.....	3
1.3 INSTALLATION ET MISE EN SERVICE :	3
2. module CAN01A :	5
2.1 Présentation :.....	5
2.2 Les modules d'interface CAN configurables :.....	5
2.3 Module 8 entrées logiques :.....	6
2.4 Module 4 sorties de puissance :	7
2.5 DESCRIPTION DU MCP25050 (le circuit intégré principale) :	8
3. Carte ATON CAN.....	10
3.1. Définition :.....	10
3.2. Caractéristiques générales :	10
4. Analyse et implémentation du protocole CAN pour le contrôle des modules via MCP25050 :.....	12
4.1. Introduction :	12
4.2. Définition d'une trame :.....	12
4.3. Types de messages CAN :	13
4.4. Identification des différents modules CAN :	14
4.5. Gestion des données avec le masque :	16
4.6. Exemple de trame pour le TP1 de VMD :	16
4.7. Références Techniques Basées sur les Documents Disponibles sur la Clé USB :.....	19

1. Carte EID210 000

1.1 Fonctions principales :

La carte processeur **EID 210 000** est un module d'étude d'un microsystème architecturé autour du microcontrôleur 68332 (de la famille 68000, fabricant Motorola).

Elle dispose d'un certain nombre de périphériques permettant le pilotage, et l'acquisition de données (tout ou rien ou analogiques) à travers un port d'extension.

La carte dispose également d'interfaces de communication série asynchrone et synchrone, d'un bus USB 1.1, et d'un bus d'extension au format "PC104".

1.2 Ressources matérielles :

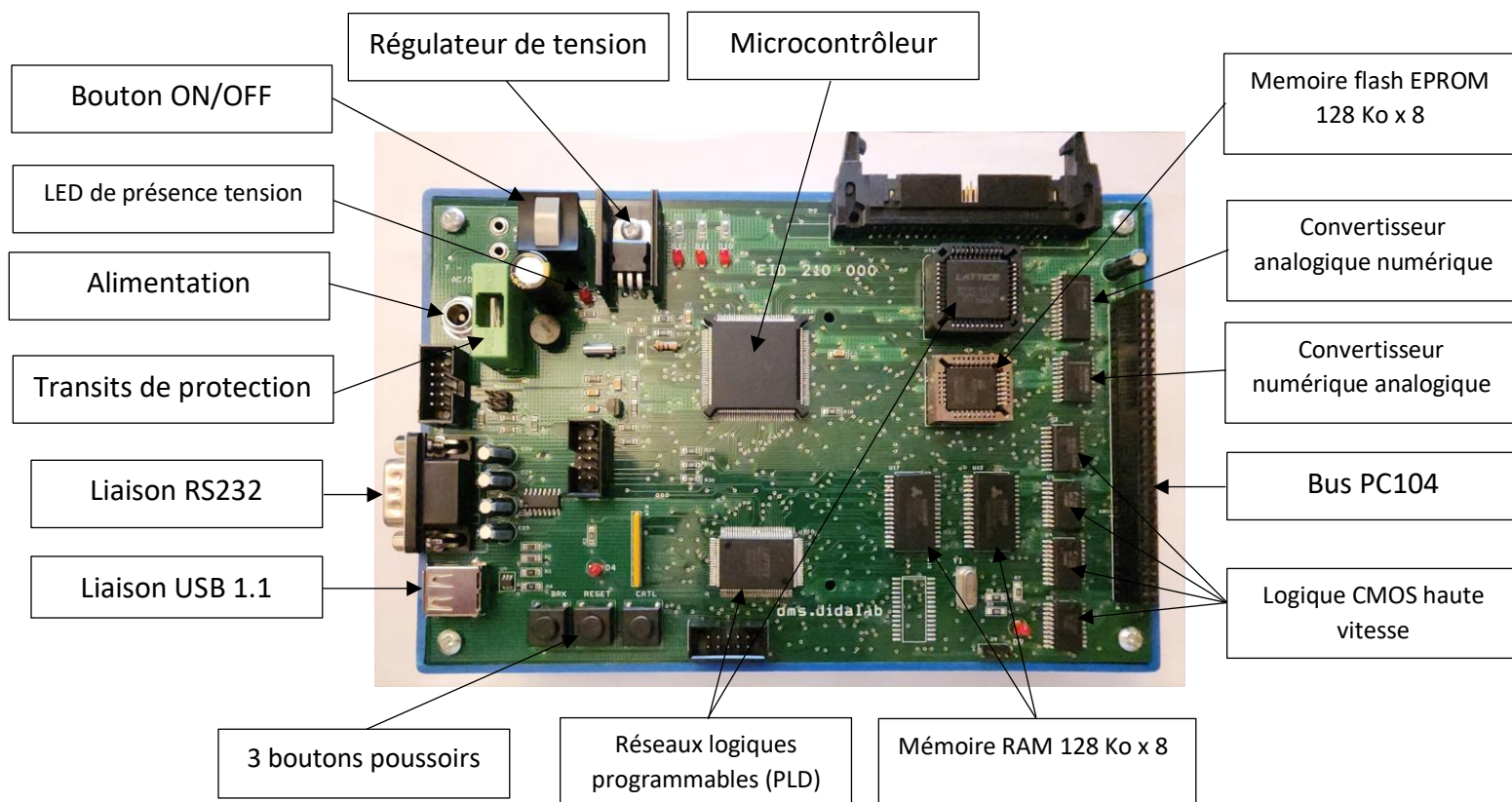
La carte processeur EID210 comporte les éléments matériels suivants :

- un microcontrôleur 68332 cadencé à 16,7 MHz.
- 128 Ko x 8 de flash EPROM.
- 128 Ko x 16 de RAM.
- deux réseaux logiques programmables (PLD) permettant:
 - La mise en forme des différents signaux (EPLD de contrôle).
 - D'avoir un port 8 bits bidirectionnel.
- un convertisseur analogique numérique 6 voies, avec 12 bits de résolution.
- un convertisseur numérique analogique 8 bits 4 sorties.
- un bus PC104 8 bits.
- une liaison RS232.
- une liaison USB 1.1.
- une liaison série synchrone de type SPI ou I2C.

1.3 INSTALLATION ET MISE EN SERVICE :

Pour installer la carte EID210, il faut :

- Relier la liaison RS232 à un port d'un ordinateur de type P.C.
- Relier l'alimenter avec une alimentation 7 à 12 V en AC ou DC.
- Appuyer sur le bouton ON/OFF pour mettre le système sous tension (la LED de présence tension doit s'allumer).



2. module CAN01A :

2.1 Présentation :

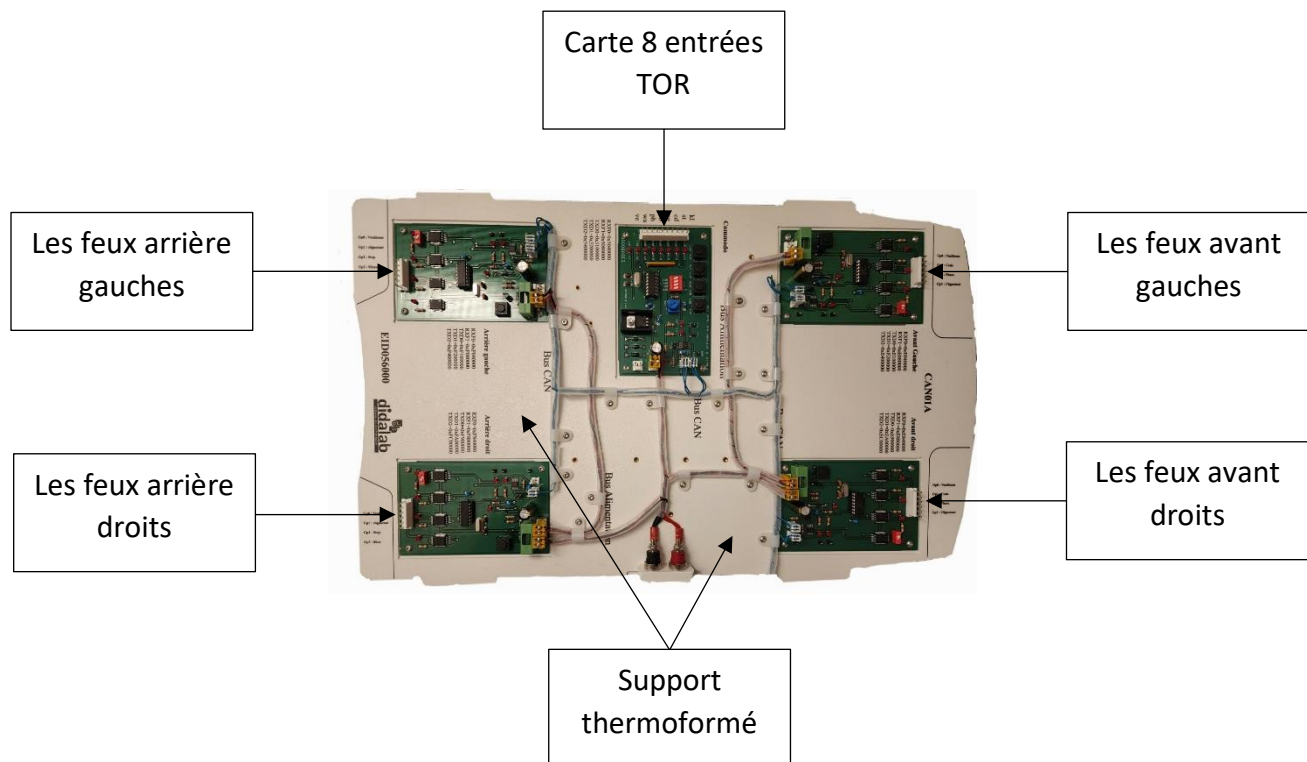
Le VMD (Véhicule Multiplexé Didactique) est un système didactique sur les RLI (Réseaux locaux industriels).

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.

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.

Le module CAN01A se compose :

- D'un support thermoformé représentant un véhicule avec ses organes de signalisation.
- D'une carte 8 entrées TOR sur bus CAN gérant le commodo lumière.
- De 4 cartes de sortie TOR, gérant :
 - Les feux avant gauches.
 - Les feux avant droits.
 - Les feux arrière gauches.
 - Les feux arrière droits.



2.2 Les modules d'interface CAN configurables :

Tous ces modules comportent les éléments suivants :

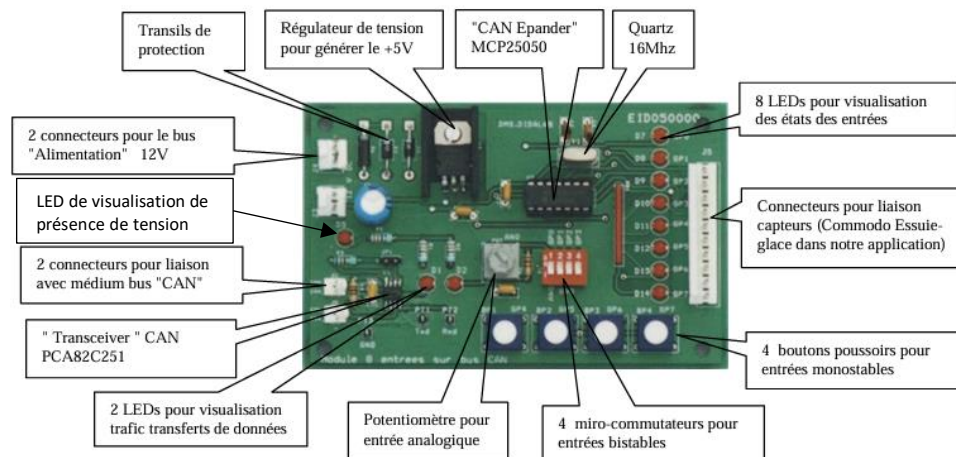
- Deux connecteurs de liaison au médium (fils de liaisons bus CAN) permettant de connecter les modules en série.
- Deux connecteurs d'alimentation (bus "alimentation énergie").
- Un régulateur de tension générant la tension +5V nécessaire aux circuits intégrés inclus sur le module.
- les protections d'usage.
- 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" (82CA251).
- Un circuit intégré d'extension d'entrées sorties (MCP25050) permettant la communication et l'interprétation des messages.
- un oscillateur à quartz nécessaire au circuit MCP25050.
- Deux LEDs de visualisation de communication en réception et en émission.

2.3 Module 8 entrées logiques :

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.



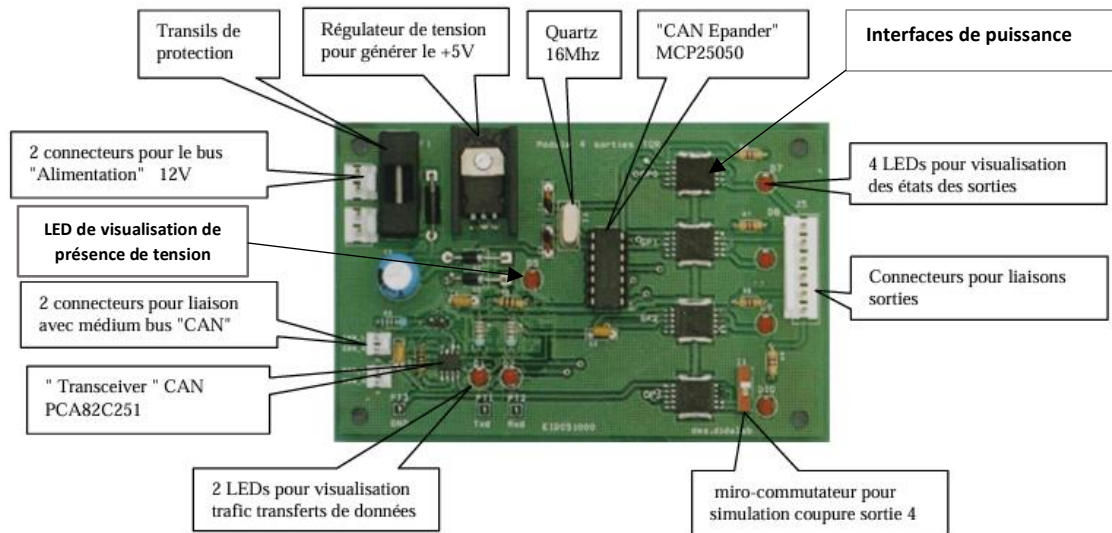
Remarque :

- Ce module peut être utilisé comme module comportant des sorties logiques compatible. 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.
- Une LED s'allume si l'entrée correspondante est forcée à 0 (commutateur en position "fermé" ou bouton poussoir "appuyé").
- Dans le cas d'une liaison configurée en sortie, il est impératif que le commutateur correspondant soit à l'état ouvert.

2.4 Module 4 sorties de puissance :

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.



Remarque :

- Le circuit intégré réalisant l'interface de puissance (**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.

- 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 courts circuits ainsi que contre les dépassements de température.

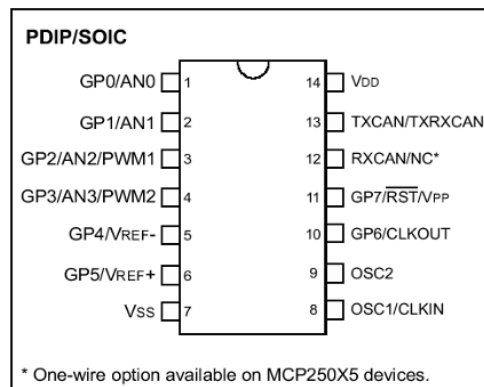
2.5 DESCRIPTION DU MCP25050 (le circuit intégré principale) :

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

Côté interface CAN : 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.

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 être commandées indépendamment l'une de l'autre. Certaines versions du circuit intègrent un convertisseur analogique numérique (4 voies) sur 10 bits.

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 "**schéduleur**" 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 :

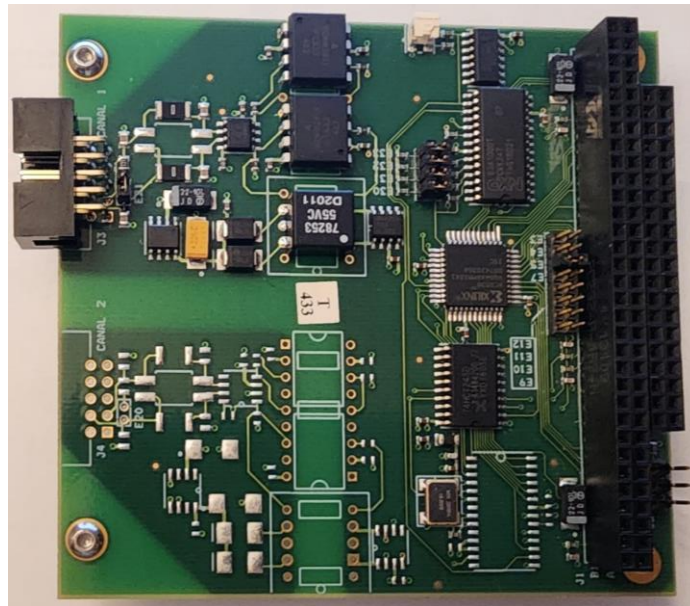
- Ces circuits ne sont pas reprogrammables.
- Le constructeur (MICROCHIP) commercialise des outils logiciels et matériels de configuration et de programmation.
- Un circuit déjà programmé peut être lu par le logiciel de configuration et de programmation.

3. Carte ATON CAN

3.1. Définition :

La carte **ATON CAN PC/104** est une carte d'interface CAN (Controller Area Network) basée sur le standard PC/104. Elle est conçue pour des applications industrielles ou embarquées nécessitant une communication via le bus CAN.

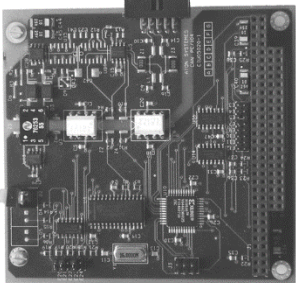
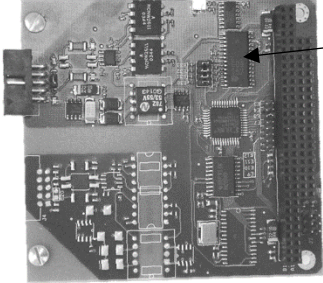
La carte **ATON CAN** assure la communication entre la carte EID210 000 et le modèle CAN01A via le protocole CAN bus. Elle gère l'envoi et la réception des trames entre ces deux composants.



3.2. Caractéristiques générales :

1. **Standard PC/104 :**
 - Compact et empilable, idéal pour les systèmes embarqués.
 - Compatible avec les systèmes PC/104.
2. **Protocole CAN :**
 - Supporte généralement le CAN 2.0A (Standard) et CAN 2.0B (Étendu).
 - Vitesses de communication configurables (jusqu'à 1 Mbit/s).
3. **Microcontrôleur ou contrôleur CAN :**
 - Intègre un contrôleur CAN comme le SJA1000 ou similaire.
 - Couplée avec un microcontrôleur pour gérer les trames et protocoles.
4. **Connectivité :**
 - Connecteurs pour interfacer avec les réseaux CAN via des ports DB9.
 - Options pour des communications dual-CAN ou CAN-FD selon le modèle.
 - pour faciliter l'intégration dans des applications spécifiques.
5. **Alimentation :**
 - Utilise l'alimentation disponible sur le bus PC/104.

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, dans notre cas on utilise la deuxième version alors on ajoutera la bibliothèque «**aton_can2.o**».

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

Contrôleur SJA1000

Pour configurer la carte **aton can**, il faut lier la bibliothèque «**aton_can.o**» ou «**aton_can2.o**» pour l'initialisation de contrôleur de bus CAN SJA1000 pour la gestion du VMD.

4. Analyse et implémentation du protocole CAN pour le contrôle des modules via MCP25050 :

4.1. Introduction :

Pour contrôler une carte **Module 4 sorties de puissance** via le bus CAN, il faut envoyer des trames CAN à le MCP25050.

Une première trame qui configure les broches en **mode sortie ou en mode entrée** en écrivant dans le registre **GPDDR (registre est uniquement responsable de définir l'état des broches en mode entrée ou sortie)**.

Et une deuxième trame qui définit l'état logique des sorties (haut ou bas) en écrivant dans le registre **GPLAT (registre gère l'état des sorties)**.

La première étape consiste à configurer les broches en **mode sortie** via GPDDR, une fois les broches configurées, en envoi une **deuxième trame** ciblant GPLAT pour définir l'état des sorties (allumer/éteindre les lampes).

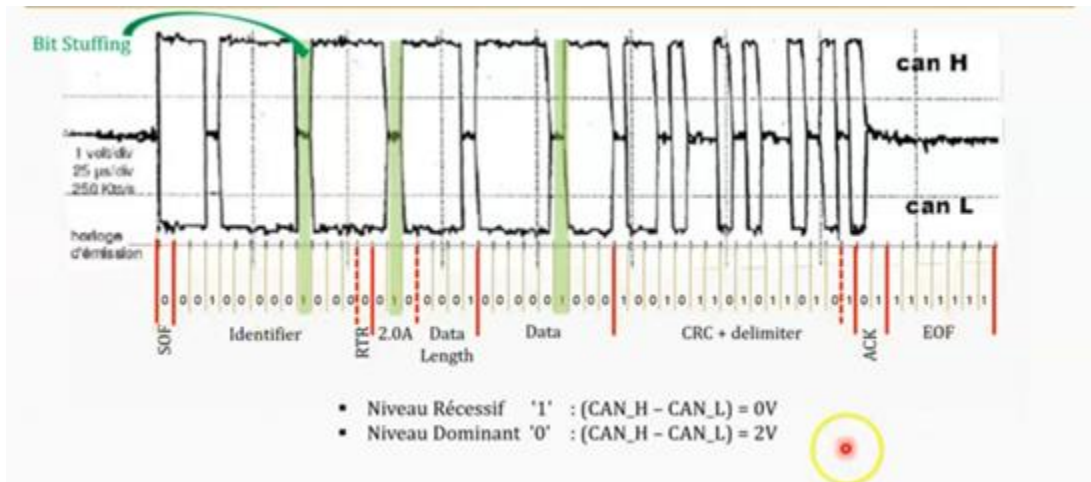
Pour effectuer cela, il est nécessaire de définir les paramètres de la trame CAN. Avant de les détailler, il est important de comprendre ce qu'est une trame CAN et quels éléments elle contient.

4.2. Définition d'une trame :

Une **trame CAN** est un message structuré qui permet la communication entre différents modules ou dispositifs connectés à un bus CAN. Elle est composée des éléments suivants :

1. Start of frame : début de la trame.
2. **Identifiant (ID)** : Un numéro unique qui identifie le message et son destinataire.
3. **DLC (Data Length Code)** : Spécifie le nombre d'octets de données que contient la trame.
4. **Données (Data)** : Contient les informations nécessaires, comme l'adresse d'un registre, un masque, et une valeur à appliquer.
5. **Contrôle** : Comprend des bits de gestion (CRC, ACK) pour assurer la fiabilité de la communication. Ces (CRC, ACK) champs sont générés et gérés automatiquement par le microcontrôleur.
6. End of frame : fin de la trame.





Les trames CAN se divisent en deux formats, **standard** et **étendu**, en fonction de la longueur de leur identifiant.

Trame standard :

- L'identifiant est codé sur **11 bits**.
- C'est le format original de la norme CAN 2.0A.

Trame étendue :

- L'identifiant est codé sur **29 bits**.
- Utilisé dans la norme CAN 2.0B.

4.3. Types de messages CAN :

- **IM (Input Message)** : Ce type de trame est utilisé pour **envoyer des commandes** ou des informations vers les modules (par exemple, allumer un feu).
- **IRM (Information Request Message)** : Ce type de trame est utilisé pour **demandeur une information** à un module (par exemple, l'état d'un feu).
- **OM (Output Message)** : C'est la **réponse** d'un module à une trame IRM, contenant les informations demandées.
- **Acp IM (Message d'acquittement)** : Le message d'acquittement est une réponse envoyée par un dispositif CAN pour confirmer qu'une requête ou une commande a été reçue.

4.4. Identification des différents 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	

Identifiants CAN uniques pour chaque module :

- Chaque module (ex: feux avant gauche, feux avant droit) possède un **identifiant unique** pour chaque type de message (IM, IRM, OM.etc).
- Ces identifiants sont définis dans le tableau sous la colonne "Identificateur (Hex)".

Relations entre les registres et les messages :

- Les registres comme RXF0, RXF1, TXD0, TXD1, etc., sont associés aux différents types de messages (IM, IRM, OM).
- Par exemple, pour le "Feux avant droit" :
 - RXF0 correspond à un message IRM.
 - TXD0 correspond à un message IM (utilisé pour envoyer des ordres).
 - TXD1 correspond à un message Acq IM.

Le tableau suivant détaille comment compléter les valeurs des identifiants pour former les trames CAN en fonction du type de message (IM, IRM, OM) et de l'action (écriture ou lecture).

TABLE 4-3: COMMAND MESSAGES (EXTENDED IDENTIFIER)

Information Request Messages (to MCP2502X/5X)																								
	Standard ID										Extended ID		Data Bytes											
	10	9	8	7	6	5	4	3	2	1	0	R T D E R E	DLC	11 7 6	RXBEID8 (8 bits)	RXBEID0 (8 bits)								
Read A/D Regs	x	x	x	x	x	x	x	x	x	x	x	1	1	0	0	0	8	n/a	n/a	n/a	n/a	n/a	n/a	n/a
Read Control Regs	x	x	x	x	x	x	x	x	x	x	x	1	0	1	1	1	7	n/a	n/a	n/a	n/a	n/a	n/a	n/a
Read Config Regs	x	x	x	x	x	x	x	x	x	x	x	1	0	1	0	1	5	n/a	n/a	n/a	n/a	n/a	n/a	n/a
Read CAN Error	x	x	x	x	x	x	x	x	x	x	x	1	0	0	1	1	3	n/a	n/a	n/a	n/a	n/a	n/a	n/a
Read PWM Config	x	x	x	x	x	x	x	x	x	x	x	1	0	1	1	0	6	n/a	n/a	n/a	n/a	n/a	n/a	n/a
Read User Mem	x	x	x	x	x	x	x	x	x	x	x	1	1	0	0	0	8	n/a	n/a	n/a	n/a	n/a	n/a	n/a
Read User Mem (bank	x	x	x	x	x	x	x	x	x	x	x	1	1	0	0	0	8	n/a	n/a	n/a	n/a	n/a	n/a	n/a
Read Register	x	x	x	x	x	x	x	x	x	x	x	1	1	0	0	0	1	addr	n/a	n/a	n/a	n/a	n/a	n/a

Output Messages (from MCP2502X/5X)																									
	Standard ID										Extended ID		Data Bytes												
	10	9	8	7	6	5	4	3	2	1	0	R T D E R E	DLC	11 7 6	RXBEID8 (8 bits)	RXBEID0 (8 bits)									
Read A/D Regs	x	x	x	x	x	x	x	x	x	x	x	0	1	1	0	0	8	IOINTFL	GPIO	AN0H	AN1H	AN10L	AN2H	AN3H	AN23L
Read Control Regs	x	x	x	x	x	x	x	x	x	x	x	0	1	1	1	1	7	ADCON0	ADCON1	OPTREG1	OPTREG2	STCON	IOINTEN	IOINTPO	n/a
Read Config Regs	x	x	x	x	x	x	x	x	x	x	x	0	1	0	1	0	5	DDR	GPIO	CNF1	CNF2	CNF3	n/a	n/a	n/a
Read CAN Error	x	x	x	x	x	x	x	x	x	x	x	0	1	0	0	1	3	EFLG	TEC	REC	n/a	n/a	n/a	n/a	n/a
Read PWM Config	x	x	x	x	x	x	x	x	x	x	x	0	1	1	1	0	6	PR1	PR2	T1CON	T2CON	PWM1DCH	PWM2DCH	n/a	
Read User Mem (bank1	x	x	x	x	x	x	x	x	x	x	x	0	1	1	0	0	8	USERID0	USERID1	USERID2	USERID3	USERID4	USERID5	USERID6	USERID7
Read User Mem (bank	x	x	x	x	x	x	x	x	x	x	x	0	1	1	0	0	8	USERID8	USERID9	USERID10	USERID11	USERID12	USERID13	USERID14	USERID15
Read Register	x	x	x	x	x	x	x	x	x	x	x	0	1	0	0	0	1	addr	n/a	n/a	n/a	n/a	n/a	n/a	n/a

Input Messages (to MCP2502X/5X)																									
	Standard ID										Extended ID		Data Bytes												
	10	9	8	7	6	5	4	3	2	1	0	R T D E R E	DLC	11 7 6	RXBEID8 (8 bits)	RXBEID0 (8 bits)									
Write Register	x	x	x	x	x	x	x	x	x	x	x	0	1	0	1	1	3	addr	mask	value	n/a	n/a	n/a	n/a	n/a
Write TX Message ID 0	x	x	x	x	x	x	x	x	x	x	x	0	1	0	0	0	4	TX0SIDH	TX0SIDL	TX0EID8	TX0EID0	n/a	n/a	n/a	n/a
Write TX Message ID 1	x	x	x	x	x	x	x	x	x	x	x	0	1	0	0	0	4	TX1SIDH	TX1SIDL	TX1EID8	TX1EID0	n/a	n/a	n/a	n/a
Write TX Message ID 2	x	x	x	x	x	x	x	x	x	x	x	0	1	0	0	0	4	TX2SIDH	TX2SIDL	TX2EID8	TX2EID0	n/a	n/a	n/a	n/a
Write I/O Configuration	x	x	x	x	x	x	x	x	x	x	x	0	1	0	1	0	5	IOINTEN	IOINTPO	DDR	OPTREG1	ADCON1	n/a	n/a	n/a
Write RX Mask	x	x	x	x	x	x	x	x	x	x	x	0	1	0	0	0	4	RXMSIDH	RXMSIDL	RXMEID8	RXMEID0	n/a	n/a	n/a	n/a
Write RX Filter0	x	x	x	x	x	x	x	x	x	x	x	0	1	0	0	0	4	RXF0SIDH	RXF0SIDL	RXF0EID8	RXF0EID0	n/a	n/a	n/a	n/a
Write RX Filter1	x	x	x	x	x	x	x	x	x	x	x	0	1	0	0	0	4	RXF1SIDH	RXF1SIDL	RXF1EID8	RXF1EID0	n/a	n/a	n/a	n/a

Identifiant CAN :

- L'identifiant est constitué de plusieurs parties, dont certaines sont fixes et d'autres variables.
- Par exemple, pour un modèle **feux arrière droit**, pour un message IM (Input Message) de type "Write Register", l'identifiant de base est :
 - 0xF88** suivi de **xx xx**, où xx xx est une valeur dynamique déterminée par le tableau.

Complément des identifiants (xx xx) :

- En fonction du type de message et de l'action, on remplit les **xx xx** avec les valeurs spécifiques décrites dans le tableau.
- Par exemples :

Input Messages (to MCP2502X/5X)																	
	Standard ID										Extended ID		Data Bytes				
	10	9	8	7	6	5	4	3	2	1	0	R T D E R E	DLC	11 7 6	RXBEID8 (8 bits)	RXBEID0 (8 bits)	
Write Register	x	x	x	x	x	x	x	x	x	x	x	0	1	0	1	3	addr

- 1** : Pour "Write Register" dans IM dans le champ **Extended ID**: xxxx xxxx xxxx x000.
Cela donne :

- Complément des xx xx = 0x00 00
- Identifiant final = 0x F 88 00 00.

Ou on a que des x les en remplace par des 0.

Et sur le champ Data en voit qu'on a trois donnée à envoyées adresse de registre cible (par exemple GPDDR ou GPLAT), masque et une valeur.

- **2** : Pour "Read Register" dans IRM dans le champ **Extended ID** : Addr xxxx x111.
 - Complément des xx xx = 0xAA 07 (AA adresse de registre utilisé **"c'est une adresse par défaut"**)
 - Identifiant final = 0x F 84 AA 07.

4.5. Gestion des données avec le masque :

Lorsque la valeur d'un bit de **masque** est à **0**, nous ignorons la donnée correspondante dans T_IM_Feux.data[2]. En d'autres termes :

- Si un bit de **masque** vaut **0**, la donnée associée dans T_IM_Feux.data[2] n'est **pas prise en compte**.
- Si un bit de **masque** vaut **1**, la donnée associée est **appliquée**.

4.6. Exemple de trame pour le TP1 de VMD :

Cette trame configure les broches en mode **sortie (feux arrière droit)**.

```
T_IM_Feux.trame_info.registre=0x00;
T_IM_Feux.trame_info.champ.extend=1; // On travaille en mode étendu
T_IM_Feux.trame_info.champ.dlc=0x03; // Il y aura 3 données de 8 bits (3 octets envoyés)
T_IM_Feux.ident.extend.identificateur.ident=0xF880000; // C'est l'identificateur du bloc optique arrière droit
T_IM_Feux.data[0]=0x1F; // première donnée -> "Adresse" du registre concernée (GPDDR donne la direction des I/O)
T_IM_Feux.data[1]=0x7F; // deuxième donnée -> "Masque" -> Les sorties sont sur les 4 bits de poids faibles
T_IM_Feux.data[2]=0xF0; // troisième donnée -> "Valeur" -> Les sorties sont sur les 4 bits lsb
```

Le nom de la trame : **T_IM_Feux**

- **T_IM_Feux.trame_info.registre=0x00** : initialise tous les bits de registre à zéro.
- **T_IM_Feux.trame_info.champ.extend=1** : Mode étendu activé.
- **T_IM_Feux.trame_info.champ.dlc=0x03** : Envoi de 3 données (car il s'agit d'un message **write register** en mode IM).
- **T_IM_Feux.ident.extend.identificateur.ident=0xF880000** : Identificateur spécifique au modèle arrière droit pour un message IM.

Contenu du champ champ data :

Addr*	Name	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Value on POR	Value on RST
1Fh**	GPDDR	—	DDR6	DDR5	DDR4	DDR3	DDR2	DDR1	DDR0	-111 1111	-111 1111
18h	EFLG	ESCF	RBO	TXBO	TXEP	RXEP	TXWAR	RXWAR	EWARN	0000 0000	0000 0000
19h	TEC	Transmit Error Counters								0000 0000	0000 0000
1Ah	REC	Receive Error Counters								0000 0000	0000 0000
50h	ADRES3H	AN3.9	AN3.8	AN3.7	AN3.6	AN3.5	AN3.4	AN3.3	AN3.2	xxxx xxxx	uuuu uuuu
51h	ADRES3L	AN3.1	AN3.0	—	—	—	—	—	—	xx-- ----	uu-- ----
52h	ADRES2H	AN2.9	AN2.8	AN2.7	AN2.6	AN2.5	AN2.4	AN2.3	AN2.2	xxxx xxxx	uuuu uuuu
53h	ADRES2L	AN2.1	AN2.0	—	—	—	—	—	—	xx-- ----	uu-- ----
54h	ADRES1H	AN1.9	AN1.8	AN1.7	AN1.6	AN1.5	AN1.4	AN1.3	AN1.2	xxxx xxxx	uuuu uuuu
55h	ADRES1L	AN1.1	AN1.0	—	—	—	—	—	—	xx-- ----	uu-- ----
56h	ADRES0H	AN0.9	AN0.8	AN0.7	AN0.6	AN0.5	AN0.4	AN0.3	AN0.2	xxxx xxxx	uuuu uuuu
57h	ADRES0L	AN0.1	AN0.0	—	—	—	—	—	—	xx-- ----	uu-- ----

T_IM_Feux.data[0]=0x1F : l'adresse de registre **PGDDR** (voir tableau pour détails).

T_IM_Feux.data[1]=0x7F : Masque appliqué : seuls les 7 bits de poids faibles sont modifiés (le bit 8 reste inchangé car il est en mode entrée et non reprogrammable).

6.2.2 Carte 4 sorties TOR

Le port 8 bit du can expander MCP25050 est configurer :

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.

La photo montre que les **sorties** se trouvent sur les **4 bits LSB (GP0 à GP3)**.
Pour configurer un bit en **mode sortie** ou en **mode entrée** :

- **Mode sortie** : mettre le bit à 0.
- **Mode entrée** : mettre le bit à 1.

T_IM_Feux.data[2]=0xF0 : Les bits **GP7 à GP4** sont configurés en **entrée**.

Les bits **GP3 à GP0** sont configurés en **sortie**.

Remarque : Les bits GP7 à GP4 du circuit MCP25050 sont déjà programmés en mode entrée et ne peuvent pas être reprogrammés, car le circuit n'est pas reprogrammable.

En conséquence, de notre travail sur le projet nous ne sommes pas intéressés par un masque comme 0x7F, qui inclut des bits inutilisables (**GP7 à GP4**). À la place, nous pouvons utiliser un masque 0x0F, qui cible uniquement les bits **GP3 à GP0**, configurables en mode sortie.

Exemple d'utilisation d'un masque :

Avec T_IM_Feux.data[2] = 0xF0 et un masque 0x0F :

- **Masque** : 0x0F :
 - Les bits **GP3 à GP0** sont pris en compte.
 - Les bits **GP7 à GP4** sont ignorés.
- **Donnée** : T_IM_Feux.data[2] = 0xF0 :
 - Les bits **GP7 à GP4** (qui valent **1**) sont ignorés grâce au masque.
 - Les bits **GP3 à GP0** sont configurés comme suit :
 - GP3 : 0 (sortie).
 - GP2 : 0 (sortie).
 - GP1 : 0 (sortie).
 - GP0 : 0 (sortie).

4.7. Références Techniques Basées sur les Documents Disponibles sur la Clé USB :

[illegible]

EID210100 >> Documents >> Travaux Pratiques >> PDF >> CAN
>> EID056010 Notice_Technique_CAN01A

Page 15.

6.2.2 Carte 4 sorties TOR
Le port 8 bit du can expander MCP23050 est configurer :

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'affichage des entrées sur la carte entré est la suivante :

GP7	GP6	GP5	GP4	GP3	GP2	GP1	GP0
Feux régulant	Feux prior	Feux code	Feux vitesse	régulant	prior	code	vitesse

6.2.2.2 Feux arrières
L'affichage des entrées sur la carte entré est la suivante :

GP7	GP6	GP5	GP4	GP3	GP2	GP1	GP0
Feux GP1 la commande	Feux GP0 code	Feux GP4 vitesse	Feux GP3 (klaxon)	régulant	code	vitesse	

Remarque : la commande klaxon est active sur le module Feux arrières gauche.

EID210100 >> Documents >> Travaux Pratiques >> PDF >> CAN
>> EID056010 Notice_Technique_CAN01A

Page 21.

[illegible]

EID210100 >> Documents >> Travaux Pratiques >> PDF >> CAN
>> Datasheet mcp25050

Page 22.

Addr*	Name	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Value on POR	Value on RES
1F0h*	GP0DR	---	DDR8	DDR9	DDR4	DDR3	DDR2	DDR1	DDR0	-111 1111	-111 1111
18h	EFLG	ESC	RBO	TXBO	TXEP	RXEP	TXWAR	RXVAR	EWARN	0000 0000	0000 0000
19h	TECF	Transmit Error Counters								0000 0000	0000 0000
1Ah	REC	Receive Error Counters								0000 0000	0000 0000
50h	ADRES0	AND.0	AND.1	AND.2	AND.3	AND.5	AND.4	AND.3	AND.2	XXXX XXXX	XXXX XXXX
51h	ADRES1	AND.1	AND.0	---	---	---	---	---	---	1111 1111	1111 1111
52h	ADRES2H	AND.9	AND.8	AND.7	AND.6	AND.5	AND.4	AND.3	AND.2	XXXX XXXX	XXXX XXXX
53h	ADRES3	AND.1	AND.0	---	---	---	---	---	---	1111 1111	1111 1111
54h	ADRES1H	AND.9	AND.8	AND.7	AND.6	AND.5	AND.4	AND.3	AND.2	XXXX XXXX	XXXX XXXX
55h	ADRES1L	AND.11	AND.0	---	---	---	---	---	---	1111 1111	1111 1111
56h	ADRES0H	AND.9	AND.8	AND.7	AND.6	AND.5	AND.4	AND.3	AND.2	XXXX XXXX	XXXX XXXX
57h	ADRES0L	AND.1	AND.0	---	---	---	---	---	---	1111 1111	1111 1111

EID210100 >> Documents >> Travaux Pratiques >> PDF >> CAN
>> Datasheet mcp25050

Page 37.