2011 : Projet ERS3-4 Vélo à assistance électrique/Scooter électrique

1. Objectifs :

L'idée est de réaliser une étude de la commande et du fonctionnement d'un vélo à assistance électrique.
Cette étude comprendra l'analyse du fonctionnement d'un vélo à assistance électrique et de ses constituants

- La programmation d'un PLD Altera pour réaliser la commande du collecteur électronique du moteur de VAE
- La programmation en C d'un microcontrôleur Beck qui agira comme superviseur
- La mise en place d'un écran indicateur LCD déporté pour l affichage des paramètres instantanés du contrôleur de VAE (vitesse, % assistance, jauge énergie, température convertisseur

2. Etude fonctionnelle d'un Vélo à Assistance Electrique.

2.1. Schéma fonctionnel d'un VAE :

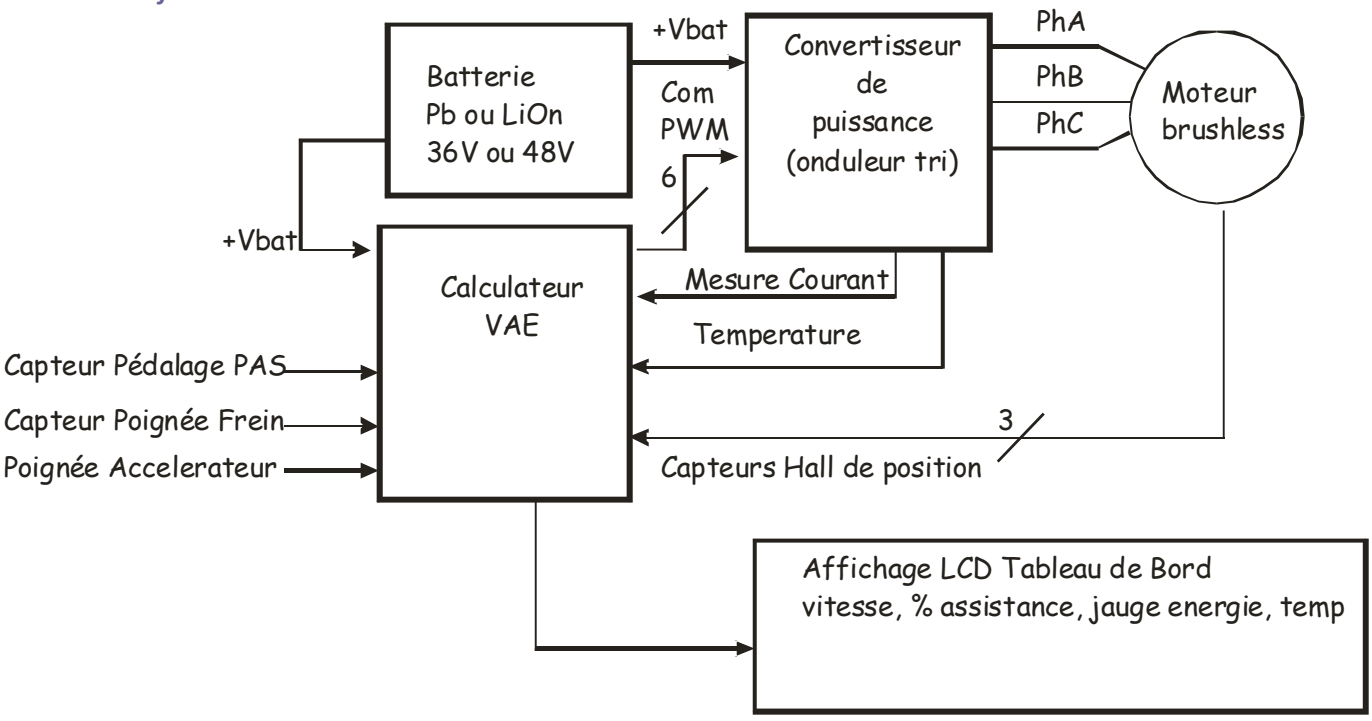


Figure 2-1 : schéma bloc d'un système d'assistance au pédalage d'un VAE

Figure 2-2 : localisation des éléments constitutifs d'une assistance électrique sur un VAE

2.2. Eléments constitutifs d'un kit d'assistance électrique

- **Capteur de pédalage PAS :**

La législation européenne impose que l'assistance électrique du vélo ne puisse être déclenchée que si l'utilisateur fait lui-même un effort de pédalage (ceci afin de ne pas considérer le VAE comme un cyclomoteur). En supplément, toute assistance doit être coupée au delà de 25Km/H.
Ainsi, la plupart du temps, la cadence de rotation du pédalier est mesurée par une roue mobile qui est fixée sur l'axe tournant du pédalier et dotée de plusieurs petits aimants. Monté fixe sur le pédalier, un capteur de pédalage doté de 2 capteurs magnétiques (à effet hall en général) mesure la cadence de passage des aimants fixés sur la roue mobile.
Le résultat de cette mesure est une tension analogique variable entre 0.8V (lorsque la vitesse de pédalage est faible) et 4.2V (à cadence maximale de 2tr/s). Le gain est en moyenne de 2V/tr/s
Si l utilisateur cesse de pédaler ou si il pédale en arrière, la tension du capteur PAS tombe immédiatement à 0V.L assistance s'arrête.

- **Capteur de poignée de frein :**

Il s'agit d'une poignée de frein classique dotée d'un contacteur électrique (ToR). Cette information booléenne logique informe le calculateur d'un freinage et donc stoppe immédiatement toute alimentation en énergie du moteur.
Certains calculateurs peuvent interpréter cette information pour enclencher une phase de freinage récupératrice et donc recharger les batteries.
Le problème est que les batteries Lithium supportent assez mal les phases de micro recharges et compte tenu de la faible énergie cinétique d'un vélo (poids et vitesse) la plupart des calculateurs du marché ne récupèrent pas l énergie au freinage.

- **Contrôleur/Calculateur :**

C'est lui qui constitue à la fois le cerveau (calculateur) et l'organe de conversion d énergie (convertisseur de puissance) entre la batterie et le moteur d assistance.
On y retrouve l'ensemble de la connectique (batterie, moteur, capteurs) et il est constitué d'un microcontrôleur bon marché associé à un hacheur (moteur DC) ou un onduleur triphasé de puissance à transistors (moteur brushless triphasés)L alimentation DC batterie se fait entre 24V et 48V.

- **Le moteur Roue :**

C'est l'organe de conversion électromécanique de la chaine de puissance. Ce moteur remplace la roue d'origine et est généralement de technologie à courant continu (DC) ou courant alternatif (machine synchrone à aimants permanents.)
Pour la version synchrone, il possède 3 phases et est doté de 3 capteurs à effets hall permettant de déterminer la position du rotor. Les aimants sont situés sur la périphérie du moteur (inducteur) et tournent avec la roue. Les bobinages (induit) sont situes au centre du moteur et sont fixes par rapport à la fourche. (Sorte de machine synchrone inversée)
Il y a en général de 10 à 20 paires de pôles (p) dans ce type de machines (machine à rotation lente donc fortement multipolaires).

La fréquence des signaux électrique observées à l induit est égale à p*vitesse de rotation mécanique du moteur.
La puissance du moteur doit être légalement limitée à 250W (réglementation européenne). Des versions offroad sont proposées en 48V 1Kw !!!! (1.5Ch sur un vélo de 15kg……)

- La batterie :

La batterie est l'élément le plus couteux de ce système.
Elle est fixée en générale au cadre ou sur le porte bagage du vélo sur un support amovible (pour recharger dans son appartement).
Les tensions batterie sont en général de 36V ou 48V (pour des raisons de sécurité et d'isolement électrique).
Les capacités batterie vont de 8Ah à 20Ah.
Deux technologies sont proposées.
Batteries au plomb, économiques mais lourdes et ayant une faible capacité massique. On les retrouve sur les modèles bas de gamme du fait de leur poids élevé.

Des *batteries au lithium* (LiOn ou LiPoFe) plus légères et dotées d'une forte capacité massique.
Les prix sont très élevés ; exemple une batterie 36V/10Ah LiPoFe (360Wh) coute aujourd'hui 600€ TTC
Pour une batterie LiPoFe de 48V/10Ah (480Wh) le cout s élève à 900€ TTC !!! Avec une telle batterie on peut espérer 60Km d autonomie environ en conditions normales.

2.3. Implantation et choix du moteur pour le Vae.

Deux types d'implantation de moteurs sont possibles

- En kit ou seconde monte, les moteurs sont généralement implantés dans la roue avant ou arrière comme sur la fig. 2.2. c est le montage le plus répandu actuellement car il ne nécessite pas de fabrication spéciale du vélo.

- En montage d'origine, le moteur peut être directement implanté sur l'axe du pédalier

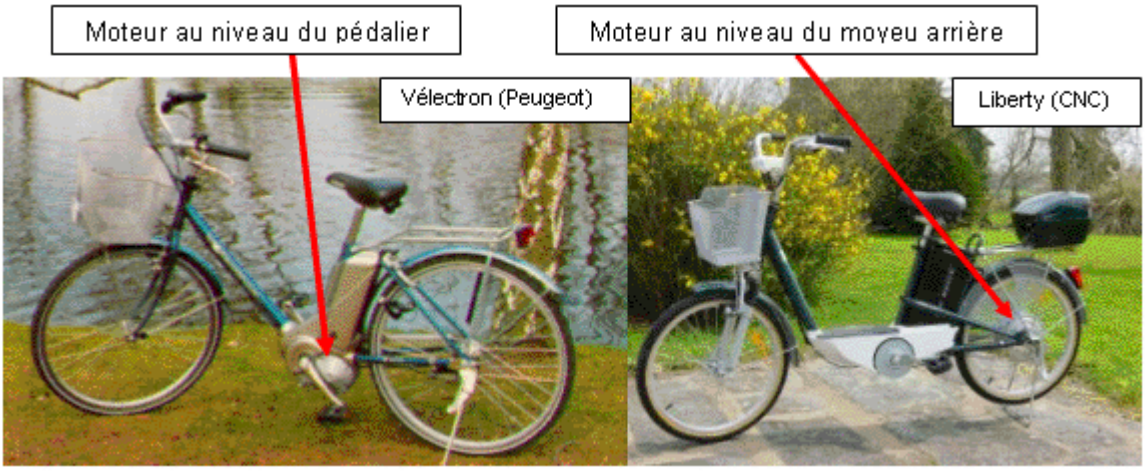
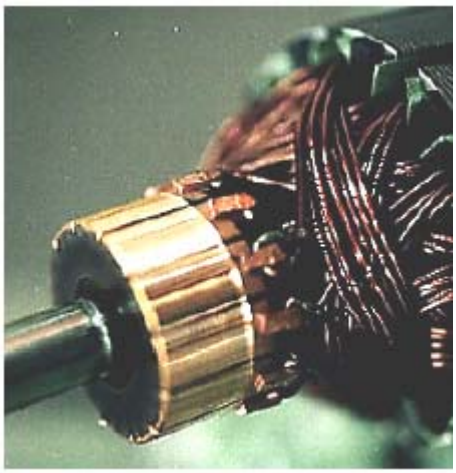


Figure 2-3 : exemple de VAE avec moteur d'assistance implanté sur l'axe du pédalier ou sur la roue arrière

Dans le cadre de notre projet, nous travaillerons sur le cas d'un moteur roue implanté sur la roue avant (pour le projet VAE) ou dans la roue arrière (pour le projet scooter électrique).

Les deux types de motorisation qui coexistent actuellement sont les suivantes :

- **Moteur DC brush à aimants permanent MCC.**



Le plus ancien et le plus simple étant le moteur a courant continu DC avec collecteur et balais.

Avantages : Il a l'avantage d'être simple de commande (la tension d'induit détermine sa vitesse de rotation) et un collecteur mécanique intégré au rotor (induit) du moteur permet d orienter mécaniquement le flux d'induit du rotor avec le flux d'inducteur (stator) pour maximiser le couple. Un simple hacheur 1 ou deux quadrants suffisait à le piloter.

Inconvénients : le collecteur mécanique et ses balais (charbons) s'usent et cohabitent mal avec un milieu extérieur sévère (humidité) auquel est soumis le vélo. Pour cette raison de fragilité au niveau du collecteur, le moteur DC est en voie d'abandon.

- **Moteur DC brushless autopiloté.**

C'est, à la base, un moteur triphasé alternatif doté au rotor d'un inducteur à aimants permanents. L idée est de considérer que, vu de l'extérieur, il se comportera comme un moteur à courant continu mais non pas doté d'un collecteur mécanique (comme le moteur DC) mais plutôt d'un collecteur électronique (réalisé avec des transistors mosfet ou igbt).

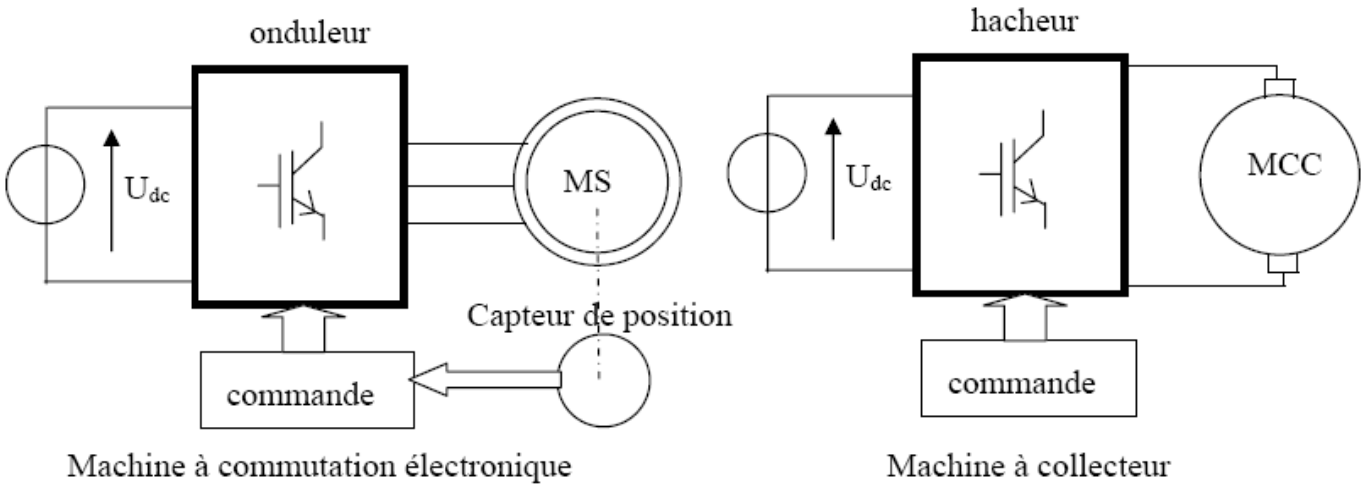


Figure 2-4 : structures de convertisseurs d énergies possibles en fonction de la machine utilisée

Dans une MCC, la rotation mécanique du rotor du moteur DC (MCC) assure la parfaite alimentation des enroulements de l'induit du rotor de manière à créer un champ d'induit perpendiculaire au champs d'inducteur.
Dans le moteur Brushless (MS) la rotation du rotor (inducteur à aimants) est mesurée par un ou des capteurs de position, et on détermine ainsi quelle phase (parmi les 3) sera alimentée à un instant donné. Cette logique de commande est appelée un « autopilotage ».

Enfin dans les deux cas, un convertisseur statique d'énergie (hacheur pour la MCC ou onduleur pour le Brushless) permet de régler la quantité d'énergie électrique qui sera injectée dans la machine.

Vue de la source d'alimentation continu (Udc) tout se passe dans les deux cas comme si on avait à faire à une machine a courant continue (on parle alors Machine DC sans Balais ⇔Brushless DC Motor.).
Aujourd hui 95% des moteurs employés sont des machines brushless triphasées (pour des questions de robustesse) ce qui fait que notre étude se limitera uniquement à ce type de moteurs associés à des onduleurs triphasés.

2.4. Méthode d'alimentation simplifiée de la Machine Synchrone : Commande d'alimentation trapèze 120°

Solution classique de la commande sinusoïdale.

Normalement, pour créer un couple moteur dans une machine synchrone dont les FEM sont sinusoïdales il faut injecter dans chaque phase un courant lui-même sinusoïdal et qui soit rigoureusement en phase avec la fem interne /phase pour obtenir une puissance active maximale sans présence de puissance réactive.

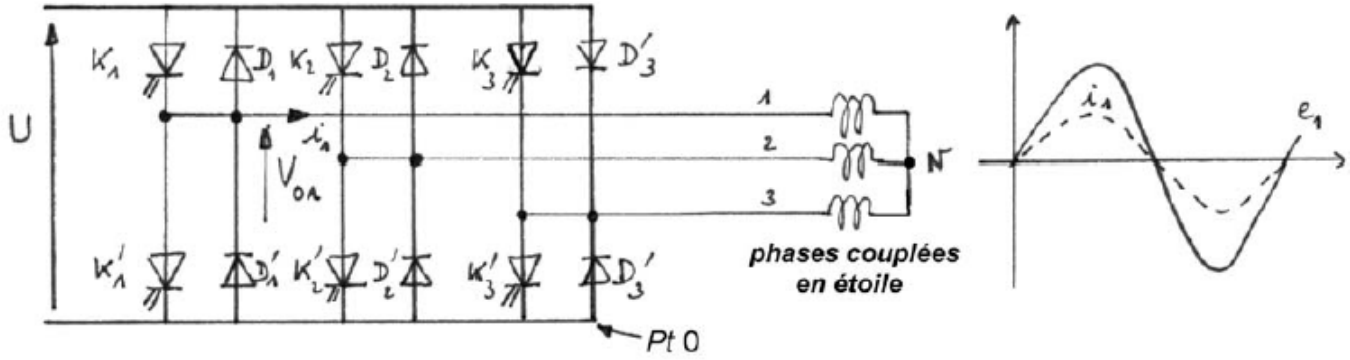


Figure 2-5 : schéma classique d'alimentation en courants sinusoïdaux triphasés d'une machine synchrone.

L'onduleur présenté sur la figure 2.5 est constitué de 3 bras de pont composés chacun de 2 interrupteurs silicium commandés à l'ouverture ou à la fermeture et réversibles en courant.
En régime permanent, pour avoir un courant sinusoïdal, la tension issue de la batterie (U) doit découpée et modulée sinusoïdalement, car la f.é.m. du moteur est sinusoïdale. La forme des courants fournis par l'onduleur est en fait, asservie à une consigne sinusoïdale liée de la position instantanée du moteur.
Chaque bras peut être commandé de manière complémentaire: lorsque K est bloqué K' conduit et vice-versa, K et K' ne sont jamais conducteurs simultanément (sinon court circuit sur le bus continu U).

La variation sinusoïdale du rapport cyclique d'un bras d'onduleur permet d'obtenir une tension à valeur moyenne glissante (à l'échelle de la période de découpage) sinusoïdale.

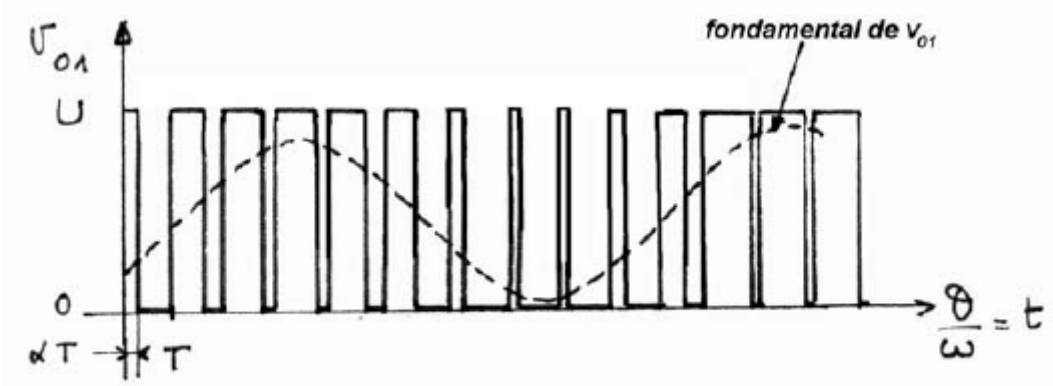


Figure 2-6 : principe de la modulation sinusoïdale et commande des interrupteurs de l onduleur.

On peut s imaginer la chose suivante pour essayer de comprendre l'échange d'énergie entre la batterie et le moteur.
Si les tensions fabriquées par l'onduleur triphasé sont rigoureusement en phase avec les FEM internes du moteur et d'amplitude égales à la FEM interne, aucun courant n'est échangé entre le convertisseur et la machine. Donc pas d'échange d'énergie (le moteur tourne à vide sans frottement).

Si on pilote l'onduleur pour fournir des tensions à vides légèrement supérieures en amplitudes aux FEM internes du moteur, alors de la puissance active sera absorbée par le moteur et se traduira pas un couple mécanique moteur.
Cette puissance électrique est fournie à partir bus continu (U) (généré par la batterie) et convertie par l'onduleur.

Si au contraire la tension délivrée par l'onduleur est légèrement inférieure à la FEM interne du moteur alors de la puissance active est fournie par le moteur à notre onduleur (il marche alors en redresseur MLI). Cette puissance est prélevée sur le coté mécanique. Le moteur agit comme un frein et l'énergie mécanique est convertie en énergie électrique récupérée puis renvoyée vers les batteries (U). On parle d'un freinage récupératif.

Problèmes liés à la commande sinus : Il faut être capable de fabriquer des tensions de forme parfaitement sinusoïdales (pas trop compliqué avec un micro contrôleur) mais en phase avec les FEM internes du moteur. Cela nécessite en général de disposer de nombreux points de mesure sur la position angulaire du rotor du moteur d'où un capteur de position précis (capteur optique ou magnétique précis de 256 pts/tour minimum) et très couteux.
Impossible dans le cas des VAE à faible cout !

Solution de la commande trapézoïdale 120°.

Pour simplifier l'alimentation du moteurs synchrones, les courants i dans chaque phases peuvent être en créneaux alternatifs (par exemple de largeur 120°). On part du principe que la FEM interne de la machine est de forme trapézoidale (voir en rouge sur la figure ci-dessous) et que le max de FEM dure 120° coté positif et 120° coté négatif.

Dans ce cas on peut se contenter d'injecter dans la machine un créneau de courant de largeur 120° dans chaque phase en synchronisme avec le max ou le min de la FEM interne. La somme des couples de chaque phase donne un couple résultant global qui devrait être constant (Pac=E1*I1+E2*I2+E3*I3=Cm*Ω=cst).
Fréquemment, dans les machines synchrones autopilotées le couple n'est pas constant, soit parce que les f.é.m. n'ont pas exactement la forme trapézoïdales optimale pour un créneau de courant, soit parce que la forme du courant n'est pas parfaitement adaptée à la f.é.m., soit pour les deux raisons simultanées.

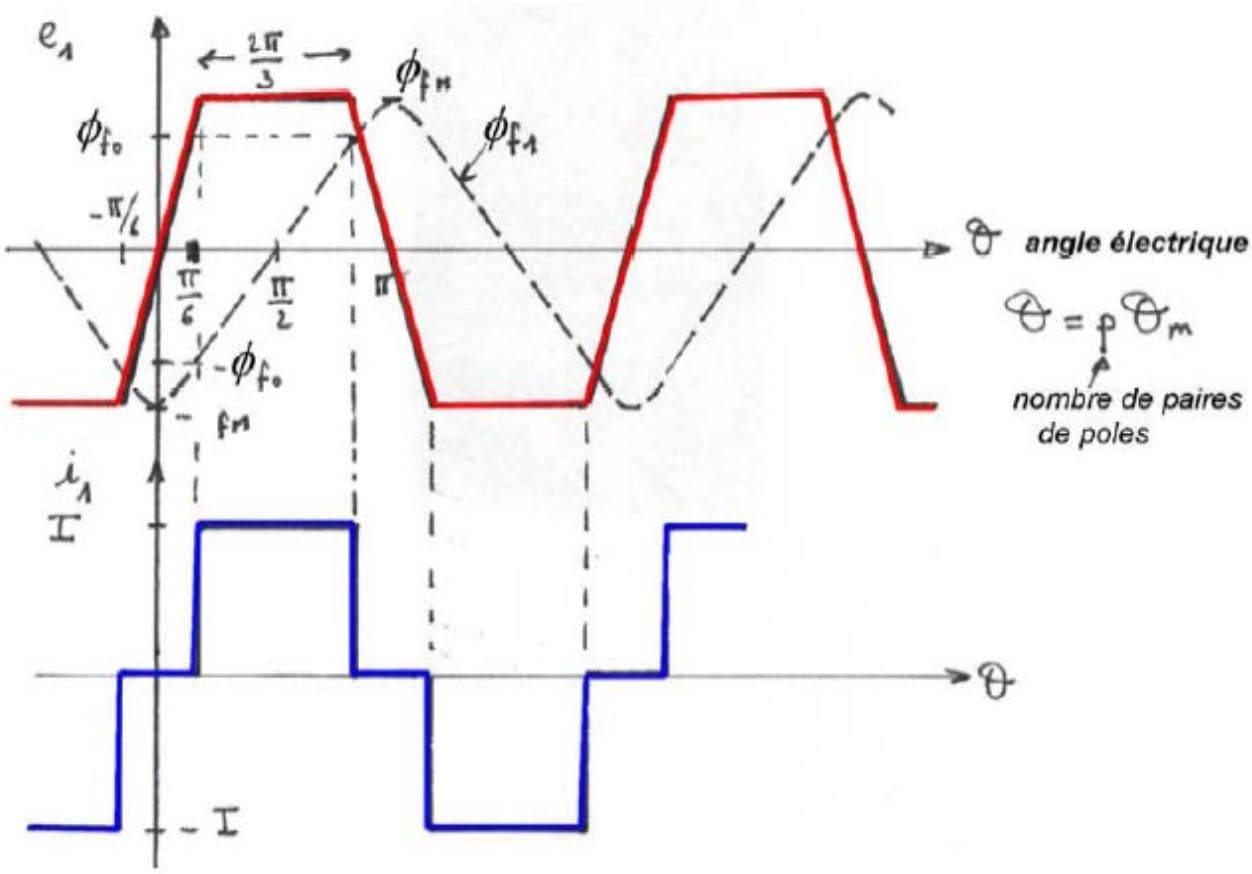


Figure 2-7 : forme de tension (rouge) et de courant (bleu) dans une commande trapèze 120°

Avantages de la commande trapèze :
La forme de courant étant plus simple et la largeur des créneaux étant de 120° un capteur de position plus simple peut être utilisé pour synchroniser le courant avec les FEM internes du moteur.
On utilise dans ce cas 3 capteurs magnétiques ToR à effet hall décalés de 120° électrique et positionnés au stator de la machine (coté induit fixe dans le cas du VAE).
Ces capteurs mesurent donc le champs magnétique créé par le déplacement d'aimants de l'inducteur (rotor). Une résolution de 6 points par période électrique suffira pour fournir les angles de commutations des courants en créneaux. Avec des forces électromotrices sensiblement trapézoïdales de durée angulaire adéquate (120° en

triphasé), une alimentation en quasi créneaux de courant donne un couple instantané peu ondulé (légère vibrations peu gênantes dans le cas du VAE).

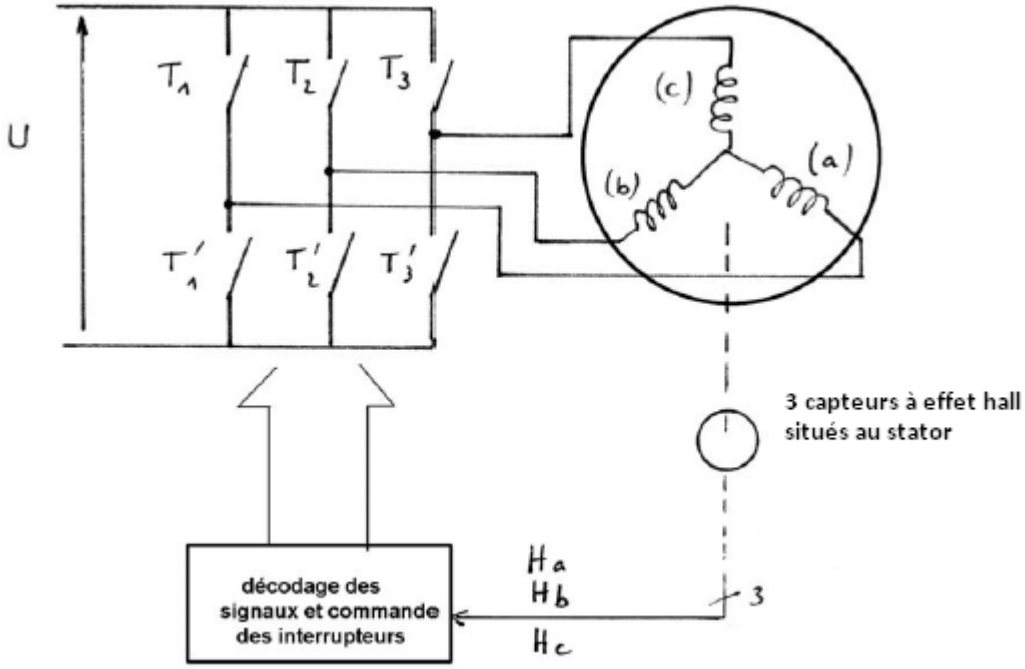
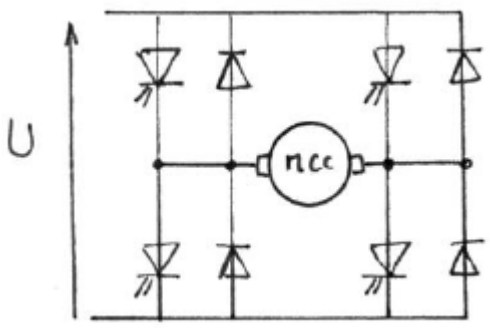


Figure 2-8 : structure de commande d'une machine DC brushless en mode trapèze.

Les 3 capteurs de position Ha, Hb,Hc délivrent par exemple 3 signaux logiques en synchronisme avec les f.e.ms internes du moteur comme le montre la figure 2.9.
Une simple logique combinatoire permet de déterminer alors quels sont les transistors de l'onduleur qui devront être pilotés pour obtenir des créneaux de courant synchronisés avec les FEM.

En général et pour des questions de simplicité de commande de l'onduleur , les transistors du bas de l'onduleur (T1'=MLIAL, T'2=MLIBL, T'3=MLICL) sont commandés en pleine onde (1 ou 0 suivant la combinaison des capteurs hall).
Pour régler l'amplitude du créneau de courant qui sera injecté dans la phase du moteur, la commande des 3 transistors du haut (T1=MLIAH , T2=MLIBH et T3=MLICH) est modulée en largeur d impulsion (PWM) en fonction de la consigne de la poignée de gaz ou du capteur de pédalage.



Cela revient à dire que cette machine comporte une sorte de « collecteur électronique » (T'1, T'2 et T'3) et un hacheur permettant de réguler le courant dans l'induit (constitué par T1, T2 et T3)
Tout se passe donc comme si, vu de la batterie (source U), on avait le schéma équivalent ci à gauche.
On a donc le droit de parler de machine à courant continu sans balais (les balais sont remplacés par T'1, T'2 et T'3).

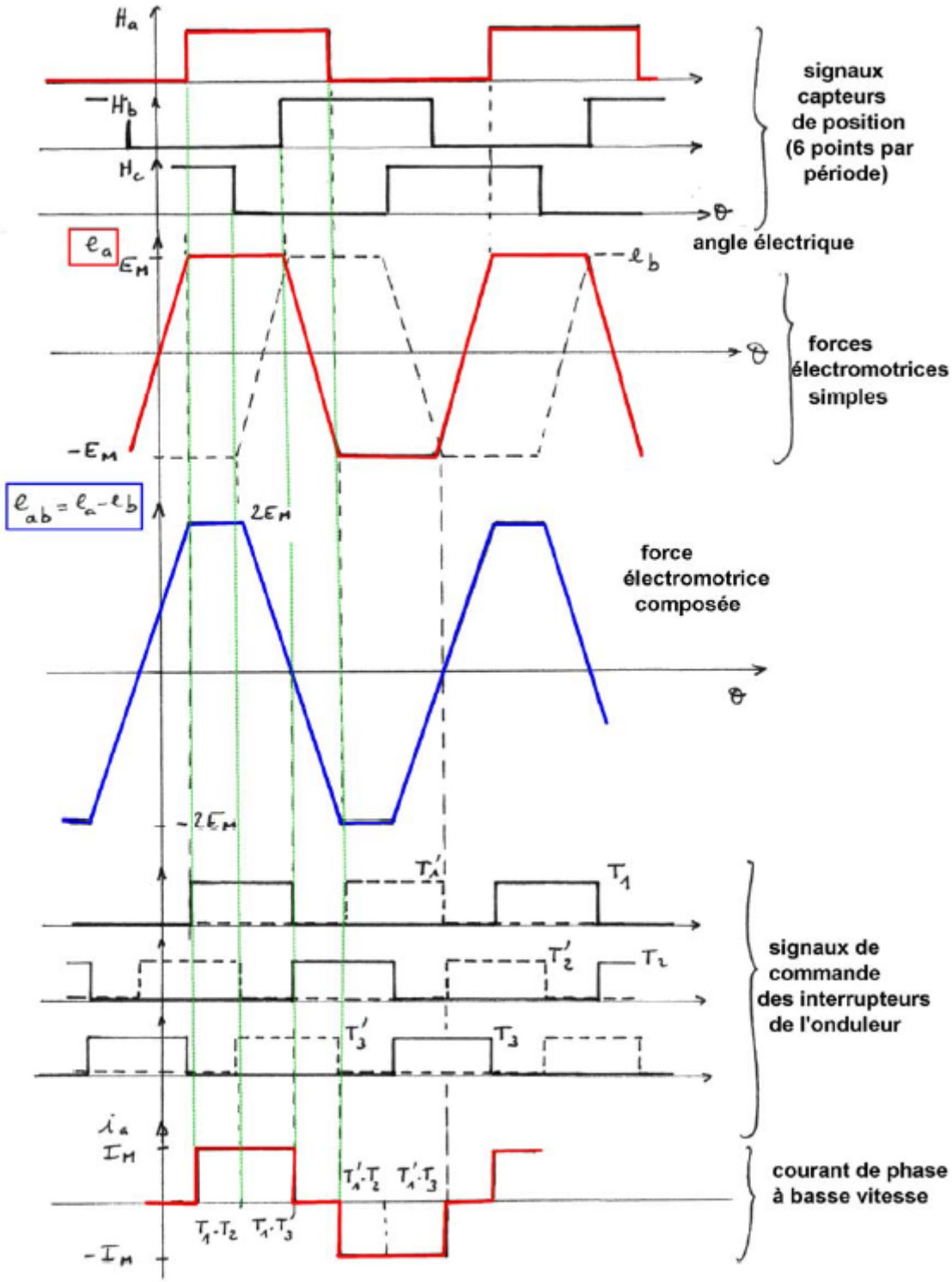


Figure 2-9 :formes d'ondes capteur/FEM interne/formes de courant/ commandes de transistor onduleur

Cette figure 2.9 est importante car elle va nous permettre de réaliser la logique combinatoire permettant de faire l'autopilotage de la machine brushless à partir des capteurs à effet Hall.

3. Travail à réaliser :

Le travail de projet sera décomposé en 2 étapes de conception sur la carte fournie qui comporte un PLD Altera et un uC Beck qui peuvent communiquer entre eux:

1) configuration en VHDL d'un circuit logique programmable qui va assurer le tache de collecteur numérique, de filtrage des capteurs à effet hall et de générateur de PWM.
2) Programmation en C d'un microcontrôleur Beck qui fournira la consigne de PWM en fonction de la consigne de pédalage. Il assurera également le rôle de superviseur et pilotera un écran LCD de supervision.

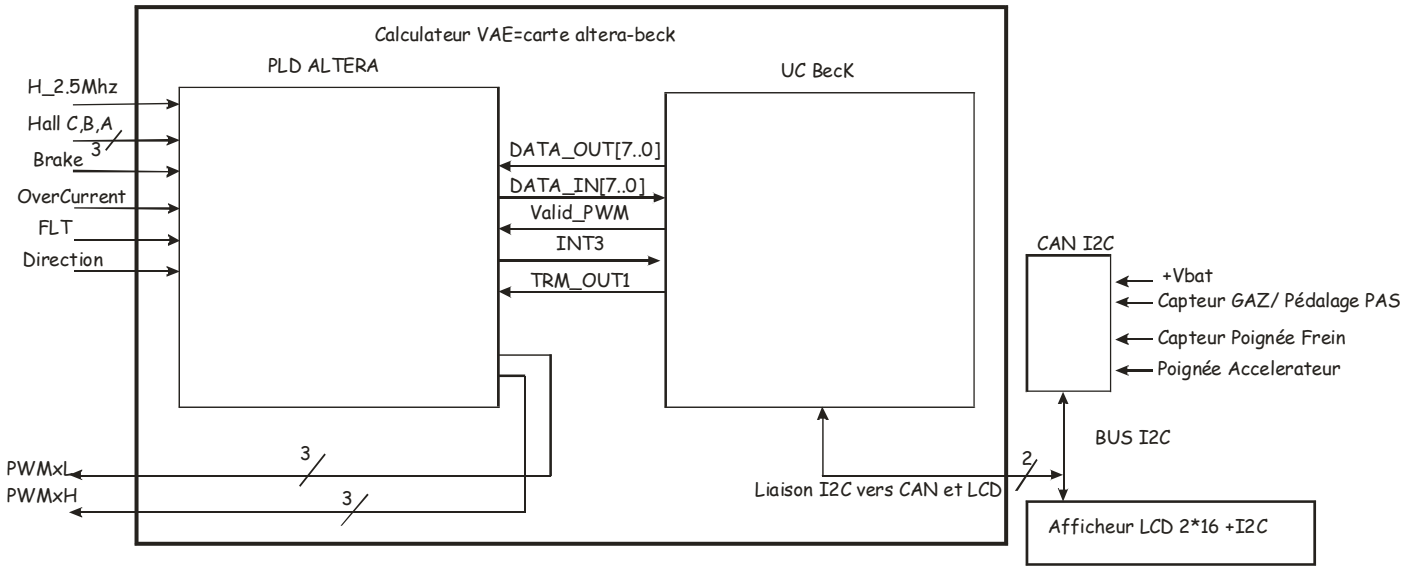
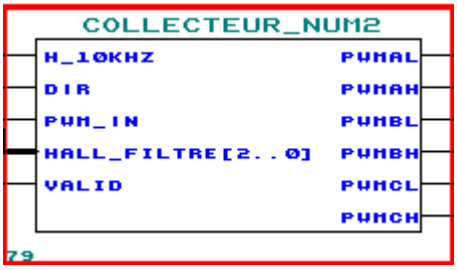


Figure 3-1 : architecture de la carte contrôleur de VAE

3.1. Logique combinatoire d autopilotage du brushless.



Rôle : Ce bloc permet de commander les 6 transistors de l'onduleur triphasé. Les transistor du Bas PWMxL sont commandés en mode pleine onde et réalisent la fonction de collecteur numérique. Les sorties PWMxH distribue la consigne de PWM reçue à l'entrée sur les transistors supérieurs de l'onduleur.

Ressources d entrée :

Hall_FILTRE[2..0]: Ce bloc reçoit l'information sur la valeur des 3 capteurs hall. correspondant à HallC, HallB, HallA. En fonction de cette information, ce bloc pilote les 6 transistors du bas PWMxL et distribue la MLI sur l'un des 3 transistors du Haut MLIxH.

VALID : Le signal valid indique à quel moment les capteurs de halls présentent une valeur stable. (Actif à 1)

DIR : permet de choisir le sens de rotation de la machine. 1=Marche <AV et 0=Marche Arrière.

H_10Khz :horloge de synchronisation générale 10Khz disponible en interne.

PWM_IN : entrée de PWM modulée en largeur d'impulsion suivant la consigne de pédalage ou de gaz.

Ressources de sortie :

PWMxL et PWMxH : commande logique des transistors de l onduleur.

Représentation de fonctionnement attendue

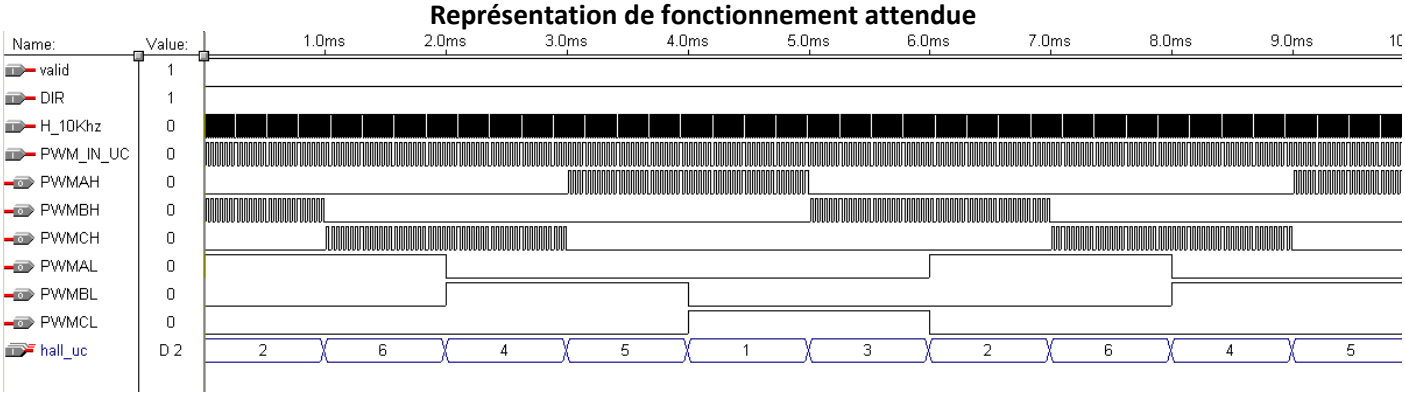


Figure 3-2 : chronogramme de fonctionnement du bloc collecteur_num2 en marche AV

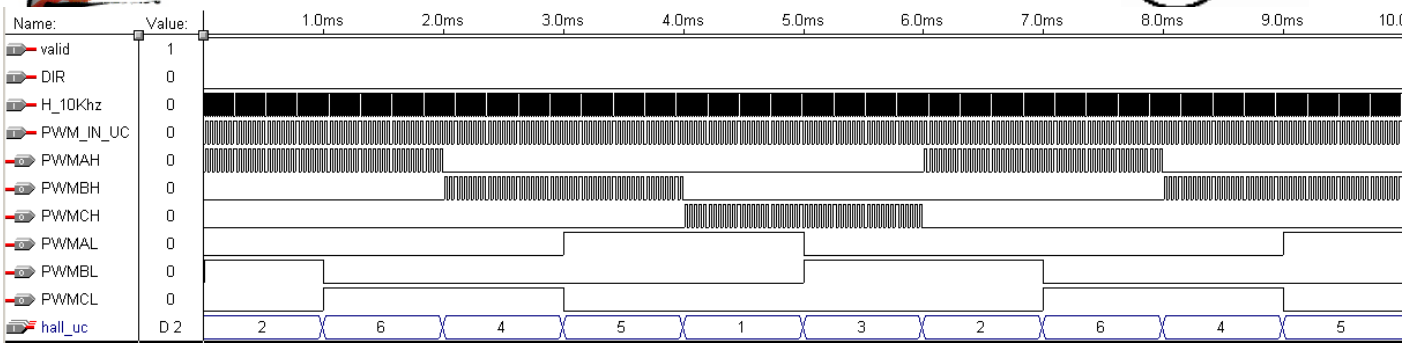
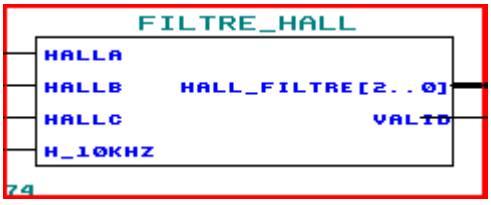


Figure 3-3 : chronogramme de fonctionnement du bloc collecteur_num2 en marche AR

3.2. Logique de filtrage des capteurs à effet hall :



Rôle : ce bloc permet de filtrer numériquement les signaux issus des capteurs hall et qui peuvent être bruités par les signaux de puissance au moment de leur changement d'état : L'idée du filtrage est de comparer le signal hall sur 2 périodes d'horloge de base consécutives et de ne valider le signal de sortie (Valid) que lorsque le signal hall d'entrée est resté constant pendant au moins 2 périodes d horloges consécutives.

Ressources d'entrée :
HallA,B et HallC sont les 3 capteurs halls de position.
H_10Khz, horloge de base interne de 10Khz.

Ressources de sortie :
Hall_FILTRE[2..0]. Regroupement de HallC,B et A après filtrage.
Valid : indicateur de signal stable de l'état des capteurs hall (actif à 1).

Représentation de fonctionnement attendue

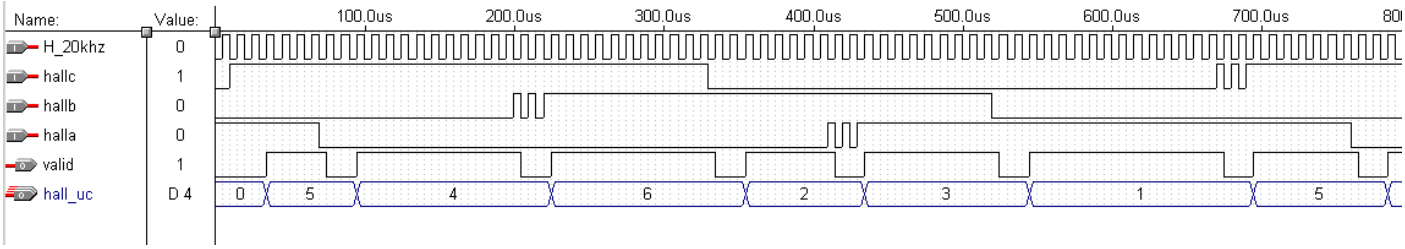
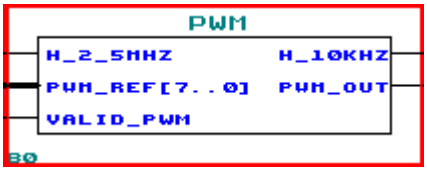


Figure 3-4 : chronogramme de fonctionnement du filtre de capteur hall.

3.3. Bloc générateur PWM



Rôle : Ce bloc reçoit la consigne de PWM sur 8 bits en // en provenance du micro contrôleur et fournit un signal de PWM de (fréquence de découpage de 10Khz) sortie unique qui sera plus tard distribuée par le bloc collecteur numérique.

Ressources d'entrée :
H2.5Mhz : horloge rapide 2.5Mhz externe.
Valid_PWM : signal de validation global de la PWM (actif à 1) permettant de prévoir un arrêt d'urgence ou une limitation en courant par détection de sur intensité. Si Valid_PWM=0, la sortie PWM_OUT=0
PWM_REF[7..0] : consigne PWM d entrée fournie par le uC beck.

Ressources de sortie :
H_10Khz : sortie horloge 10Khz pour usage interne dans le PLD.
PWM_OUT : sortie PWM modulée en fonction de PWM_REF[7..0]

Représentation de fonctionnement attendue

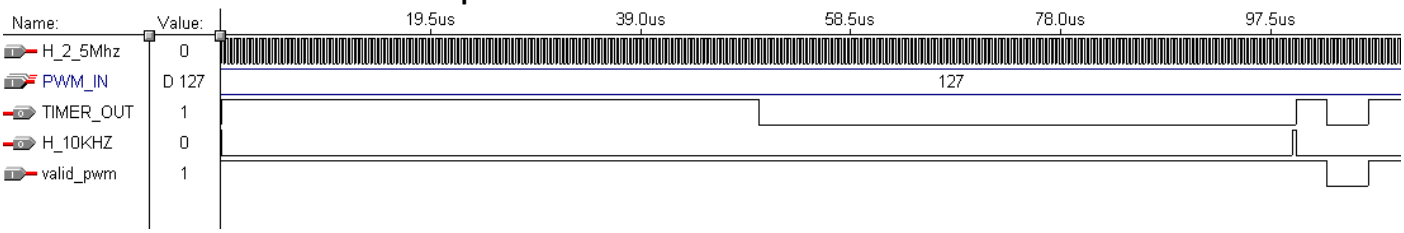
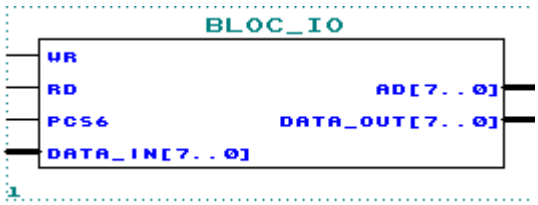


Figure 3-5 : chronogramme de fonctionnement du filtre de capteur hall.

3.4. Bloc de communication PLD/uC Beck



Rôle : ce bloc (qui est fourni en code source VHDL) permet l'échange bidirectionnel de mots de 8 bits // entre le PLD Altera et le uC Beck.

En effet le PLD Altera est implanté sur le bus de données du Beck et donc est localisé dans son plan adressable entre l'adresse 0x600 et l'adresse 0x6FF (décodage par /PCS6).

Si le uC tente une lecture d'une de ces adresses (via la bibliothèque port.c et port.h et la fonction PIO_Read()) le PLD lui retourne un vecteur de 8 bits DATA_IN[7..0].
De la même façon, si le uC tente une écriture dans cette plage d'adresse (PIO_Write()), la valeur écrite se retrouve disponible dans le PLD sur le vecteur DATA_OUT[7..0].
Les deux vecteurs DATA_IN[7..0] et DATA_OUT[7..0] seront affectés selon le tableau suivant :

3.5. Tableau et affectation générale des signaux échangés entre le Beck et le PLD Altera

Pin sur PLD Altera.	**Broche connecteur sur la carte Beck**	Localisation du les vecteurs DATA_IN et DATA_OUT dans PLD	**Désignation du signal**
		DATA_OUT[7..0]	PWM_IN[7..0]
39	**13**	DATA_IN0	HALLA
40	**14**	DATA_IN1	HALLB
41	**15**	DATA_IN2	HALLC
36	**12**	DATA_IN3	DIRECTION (1=AV 0=AR)
37	**11**	DATA_IN6	OVER_CURRENT* (actif=0)
6	**18**	DATA_IN4	FLTA (actif=1)
5	**17**	DATA_IN5	BRAKE* (actif =0)
33	**9**	PIO13/TIMER_OUT0	VALID_PWM (actif1)
34	**10**	PIO3/TIMER_OUT1	--
31	**--**	INT3/PIO12	Valid_Hall,Brake,Over_Current
43	**25**	INT4/PIO5	H_2.5MHZ
16	**24**		PWMAL
14	**23**		PWMAH
12	**22**		PWMBL
11	**21**		PWMBH
9	**20**		PWMCL
8	**19**		PWMCH
27	**--**	WR	WR
29	**--**	RD	RD
28	**--**	PCS6	PCS6
	4		*V GAZ/PEDALAGE*
	5		*VBAT*
	6		*VTemp*
	7		*Vmes*

Figure 3-6 : tableau générale d'affection des signaux entre le PLD et le uC Beck

4. Planning de progression du travail pour le semestre 3

seance1:
Présentation du projet. Réalisation du bloc collecteur numérique /simulations VHDL

seance2:
Réalisation du bloc filtre hall/simulation
Réalisation du bloc PWM /simulation
Assemblage des blocs
Réalisation d'un petit programme de test permettant de saisir une consigne de MLI

Séance 3:
Premiers essais pratiques en rotation du moteur. Mesure du nombre de paires de pôles.
Programme de mesure de la vitesse de rotation du moteur en fonction des capteurs hall de positon.
Mise ne place de la limitation de vitesse a 25Km/H

Séance 4:
Mise en place du blocage de l'assistance en cas de freinage.
Mise en place d'une limitation de courant en cas de sur intensité.

Présentation de l'I2C
Test de l i2c sur un 8574A.

seance5:
Pilotage du CAN I2C : affichage des paramètres instantanés dans un menu console.
Commande de la vitesse de la roue à partir de la poignée de gaz.
Affichage console des paramètres température, tension batterie, courant mesuré.

Séance 6 :
Principe de commande du moteur en courant avec une régulation/limitation de courant sommaire.
Présentation du principe de réalisation d'une page WEB et du mécanisme CGI
Mise en place d'une page web dynamique pour afficher les paramètres instantanés.

Séance 7:
Suite page web et conclusion sur cette première étape

DS écrit de synthèse

La note d'ERS3 sera basée sur 3 critères

1) **notes de validation des étapes demandées en séances**
2) **Note d'efficacité/comportement en séance**
3) **Note de tenue du cahier**
4) **Note finale du DS de synthèse.**

5. Séance 2 : Réalisation d'un programme de test pour la saisie d'une consigne MLI en mode console

L'objectif du programme qui sera implanté dans le uC Beck est de permettre, via un petit menu console en liaison telnet, de pouvoir saisir ou visualiser des paramètres pour notre contrôleur de VAE.

5.1. Mise en place d'un menu console à choix multiples :

Voici à quoi peut ressembler ce menu console :

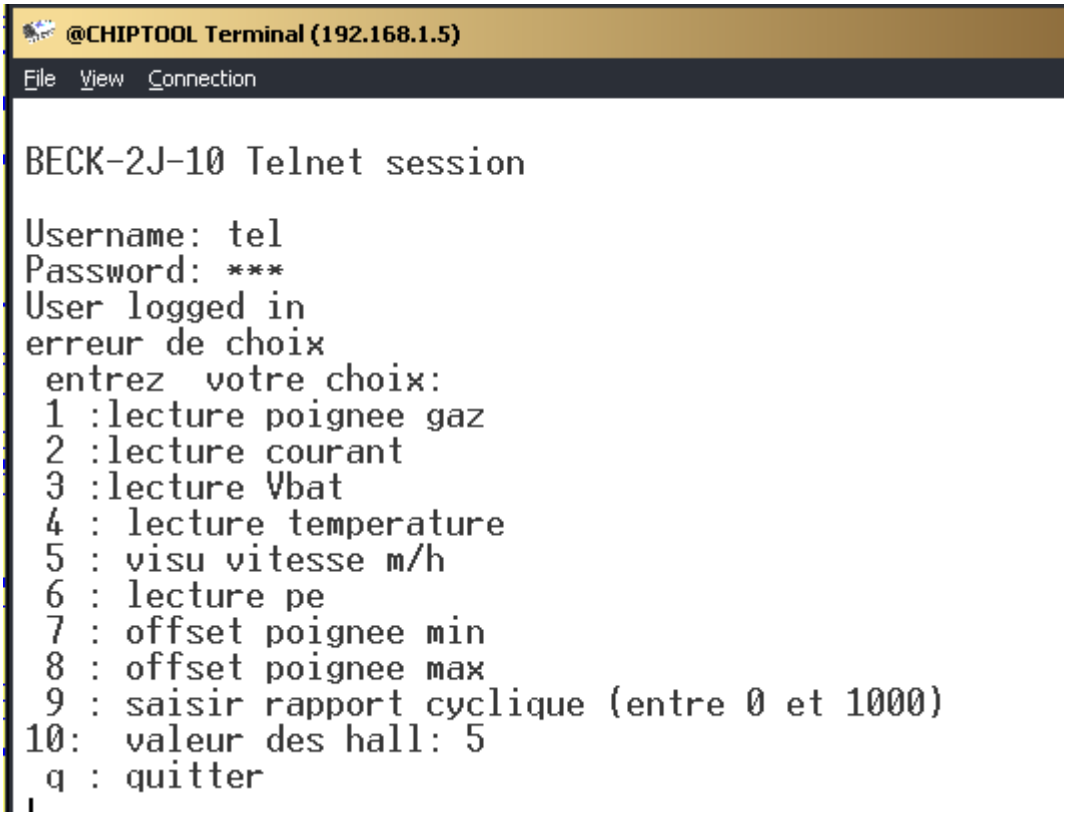


Figure 5-1 : menu console permettant la commande et la visualisation des paramètres du VAE

Ce menu sera géré en tache de fond. On pourra en sortir en tapant la lettre 'q' (quit) ce qui provoquer la mise à 0 de la MLI et sa dévalidation générale.

Dans un premier on ne demande de gérer que deux options :

- la possibilité de saisir un rapport cyclique
- la possibilité de visualiser la valeur du port d'entrée Data_IN[7..0] affin d'afficher la valeur de brake, direction, overcurrent
- la possibilité d'afficher la combinaison des capteurs à effet hall.

5.2. Travail à effectuer

1) Réaliser ce programme de menu console et vérifier que la saisie d'une consigne de PWM (entre 0 et 255) provoque bien la génération d'une MLI dont le rapport cyclique sera variable entre de 0 à 1 (cette manipulation suppose que le PLD ait été complètement programmé).
Ce rapport cyclique de commande sera observable sur l'un des transistors supérieurs de l'onduleur (PWMxH) suivant la position du moteur roue

2) réaliser un premier test de rotation du moteur en montant progressivement la tension de la partie puissance. On doit observer que la vitesse de rotation du moteur est pilotée par cette consigne de PWM saisie en mode console.

6. Séance 3 : Premiers essais pratiques du moteur en rotation et mesure de la vitesse

6.1. Mesure du nombre de paires de pôles du moteur roue.

Ce moteur synchrone possède plusieurs paires de pôles magnétiques P par phase. Cela signifie qu'il faut réaliser P périodes électrique d'alimentation pour aboutir à un tour mécanique complet.
La mesure du nombre de paires de pôles se fait aisément en comptant le nombre de périodes complètes d'un signal de capteur hall pour une rotation de 1 tour mécanique du moteur.

Pour cela faire un repère au feutre sur la roue (en concordance avec la fourche du support) lors d'un front montant d'un des capteurs hall. Compter le nombre de périodes de ce signal capteur pour une rotation de 360° mécanique.

6.2. Mesure de la vitesse de rotation du moteur

Une fois le nombre P obtenu, proposez une structure de programme faisant intervenir des interruptions externes INT3 et celle liée au Timer1 pour permettre le comptage du nombre de période capteur hall sur une période de 0.1s.
On modifiera le programme du PLD pour que celui-ci active un front montant sur l'interruption INT3 à chaque nouvel état stable des capteurs hall.
L'interruption INT3 dans le uC Beck permettra de compter le nombre de changement d'état des capteurs hall pendant qu'une seconde s'est écoulée (mesure de la seconde réalisée par une interruption timer1).
Mesurez le diamètre de la roue (pneu compris) et proposez une méthode de calcul de la vitesse tenant compte de P, du diamètre et du nombre de périodes hall comptées pendant un seconde.

Rajouter au menu console l'affichage de la vitesse en km/h et vérifier avec un tachymètre optique pour différentes valeurs de consignes de PWM.

6.3. Mise ne place de la limitation de vitesse a 25Km/H

Proposez une solution pour que la consigne PWM appliquée au moteur soit diminuée progressivement si la vitesse de rotation du moteur dépasse 25km/H.
Si la vitesse repasse sous la limite légale alors la consigne de PWM sera à nouveau ré-augmentée pour rejoindre la valeur de consigne initiale.

7. Séance 4 : mise en place de la gestion freinage/overcurrent et présentation I2C

7.1. Blocage de l'assistance en cas de commande freinage.

Proposez à présent une solution logicielle permettant de stopper l'assistance électrique si l'utilisateur actionne la poignée de frein (bouton poussoir sur le contrôleur).
Si le frein est relâché ,il faut pouvoir rétablir la consigne à sa valeur précédente soit de manière directe soit de manière progressive (par pas de 1 toutes les 5ms par exemple jusqu'à retrouver la consigne précédente) .
La gestion du frein sera réalisé par une interruption cyclique qui analysera la valeur du port DATA_IN[7..0].
En profiter pour rajouter l'affichage de la valeur du frein dans le menu console.

7.2. Limitation de courant en cas de sur intensité (optionnel).

Vous avez pu observer que tout changement brutal de la consigne de PWM provoquait un appel de courant important sur l'alimentation. Tentez de le justifier.
Le contrôleur fournit un signal (/Overcurrent actif à 0) . Proposer une solution permettant de remettre la PWM à 0 des que ce signal devient actif sans le uC contrôleur ait à intervenir. Cela constituera une protection rapprochée du contrôleur.
Modifier en conséquence le programme du PLD pour intégrer cette protection rapprochée .
Modifier également le code C du uC pour que celui-ci interprète également toute sur intensité comme une commande de réduction de la consigne MLI (décrémentation de la consigne toute les 5ms) . Lorsque la surintensité disparait la consigne PWM pourra à nouveau se ré incrémenter jusqu' ' à sa valeur de consigne.

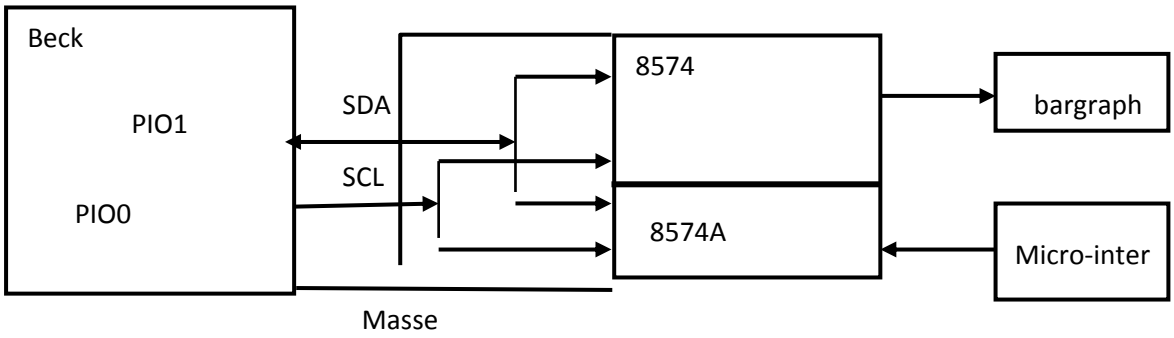
7.3. Présentation du protocole I2C et test sur un port d'E/S déporté PCF8574.

Gestion I2C par microcontrôleur Beck

Cahier des charges :

On désire commander un circuit I2C 8574 qui est un interface d'E/S parallèles (8bits). Pour faciliter la mise en œuvre, on utilise une maquette Test_I2C (voir descriptif ci-joint) avec :

- un 8574 (U1) relié à un bargraph et qui fonctionnera en port de sorties
- un 8574A (U2) relié à des micro interrupteurs (DSW1) et qui fonctionnera en port d'entrées.



Etude logicielle des fonctions fournies (fichier de bibliothèque i2c.c)

Voici ci dessous la liste des fonctions mises à disposition par le jeu d'API Beck pour piloter le bus I2C depuis le processeur SC13.

Control Functions

I2C_init	I2C Master: Initialize the I2C Bus
I2C_scan	I2C Master: Scan for I2C slave devices
I2C_scan_ext	I2C Master: Scan for I2C slave devices (extended address)
I2C_release	I2C Master: Release I2C bus
I2C_restart	I2C Master: Restart I2C
I2C_select_clock_pin	I2C Master: Select I2C Clock Pin
I2C_select_data_pin	I2C Master: Select I2C Data Pin
I2C_set_speed	I2C Master/Slave: Select I2C bus speed

I2C_reset I2C Master/Slave: Reset I2C bus controller

Transmit Functions

I2C_transmit_char I2C Master: Send a single character
I2C_transmit_block I2C Master: Send a block of characters
I2C_transmit_char_ext I2C Master: Send a single character (extended address)
I2C_transmit_block_ext I2C Master: Send a block of characters (extended address)

Receive Functions

I2C_receive_char I2C Master: Receive a single character
I2C_receive_block I2C Master: Receive block of characters
I2C_receive_char_ext I2C Master: Receive a single character (extended address)
I2C_receive_block_ext I2C Master: Receive block of characters (extended address)

Voici un exemple d'utilisation de ces fonction I2C pour réaliser l'initialisation du port I2C (init_i2c()) ainsi que l'écriture de 3 octets dans un esclave dont l'adresse I2C est 0x48 (0x48<<1= 0x90).

C:\...\testi2c.c

```
#include <clib.h>


#define AD_I2C 0x90
#define SDA_PIN 1 // set PIO1 as SDA
#define SCL_PIN 0 // set PIO0 as SCL


void init_i2c(void)  {
I2C_select_data_pin ( SDA_PIN);//select SDA pin
I2C_select_clock_pin ( SCL_PIN);// select SCL pin
I2C_init ( );// init I2C pin and bus
}

char byMess[3]={0x40,0x00,0x0B};//data to send

void main (void) {
init_i2c();// init global ofr I2C bus
I2C_transmit_block ( AD_I2C,byMess,3);// send AD+3 data bytes
I2C_release();// send stop condition to release bus

}
```

Figure 7-1:exemple de code permettant l'envoi de 3 octets sur le bus I2C

7.4. Réalisation d'une fonction communication entre un maître émetteur et un esclave récepteur.

L'esclave récepteur est un 8574 situé sur la maquette Test_I2C et dont l'adresse fixe 0x20 et l'adresse locale déterminée par le jumper JP8 est 0 ou 1 (dans notre cas il vaudra 0).
Créer un projet dans le répertoire z:\er2\seance3 : testI2C.ide.
Créer un répertoire z:\er2\biblio\ qui contiendra plus tard l'ensemble des fichiers de bibliothèques crées au cours des séances.
Pour ce projet créer 3 fichiers texte :

s3_1.c qui sera l'applicatif principal de test (sera stocké dans Z:\er2\seance3
biblio_I2C.c qui contiendra les fonctions de bibliothèque int Write_8574(unsigned char) et Init_I2C()

biblio_I2C.h qui contiendra les #define (pour SDA et SCL) ainsi que les prototypes des fonctions I2C. Ces deux derniers fichiers seront stockés dans z:\er2\biblio

1) Créer les deux fonctions de communication I2C avec le 8574 pour piloter le barreau de led
2) Faire tourner le programme de test qui permet d'envoyer une valeur de comptage comprise entre 0 et 255 vers le barreau de led .

Relever , à l'oscilloscope, le chronogramme de SDA et SCL dans le cas d'une communication sans erreur (esclave présent) puis en forçant une mauvaise communication au niveau de l'esclave (en déplaçant le jumper JP8 pour changer l'adresse locale du 8574).

3) visualiser sur la bargraph l'évolution du comptage. Faire valider votre travail par les enseignants.

7.5. S'il reste du temps : Réalisation d'une communication entre un maître écouteur et un esclave parleur.

L'esclave est un 8574A situé sur la maquette Test_I2C et dont l'adresse fixe est 0x38 Son adresse locale est déterminée par le jumper JP7. Il est en position 1 a priori

Faire une manipulation identique à la précédente, modifier s »_1.c . La donnée transmise par le 8574A est fournie par les micro-interrupteurs (DSW1) situés sur la maquette I2C

.

8. Séance 5 : Réalisation de la commande d'un CAN I2C

Notre onduleur de puissance nous fournit 4 signaux analogiques qui sont connectés sur un convertisseur analogique/numérique PCF8591:
Canal 0 : Consigne de pédalage/Poignée de Gaz.
Canal 1 : Courant mesuré par un capteur ACS754-05
Canal 2 : Tension batterie mesurée par un pont diviseur de tension (rapport à déterminer)
Canal 3 : Température du convertisseur mesurée par un capteur LM335.

L objectif de la séance est de réaliser une fonction de lecture des 4 canaux analogique du CAN afin de pouvoir ensuite les exploiter dans le code C du uC beck.

8.1. Analyse de la documentation constructeur du PCF8591

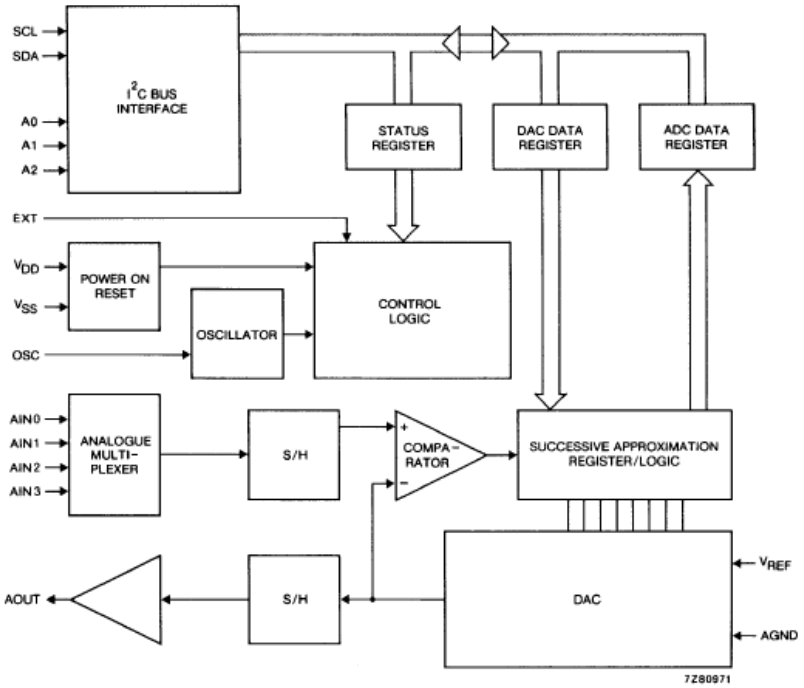


Figure 8-1 :Schéma bloc interne du PCF8591

Le circuit PCF 8591 dispose d'un CAN 4 voies 8 bits ainsi qu'un CNA 8bits compatible avec le bus I2C.
Ce circuit possède une adresse fixe (par construction 0x41 notée sur 7 bits) ainsi qu'une adresse configurable au moyen de 3 pattes présentes sur le boîtier du circuit.

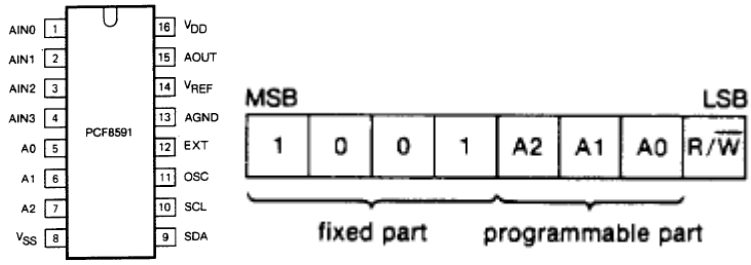
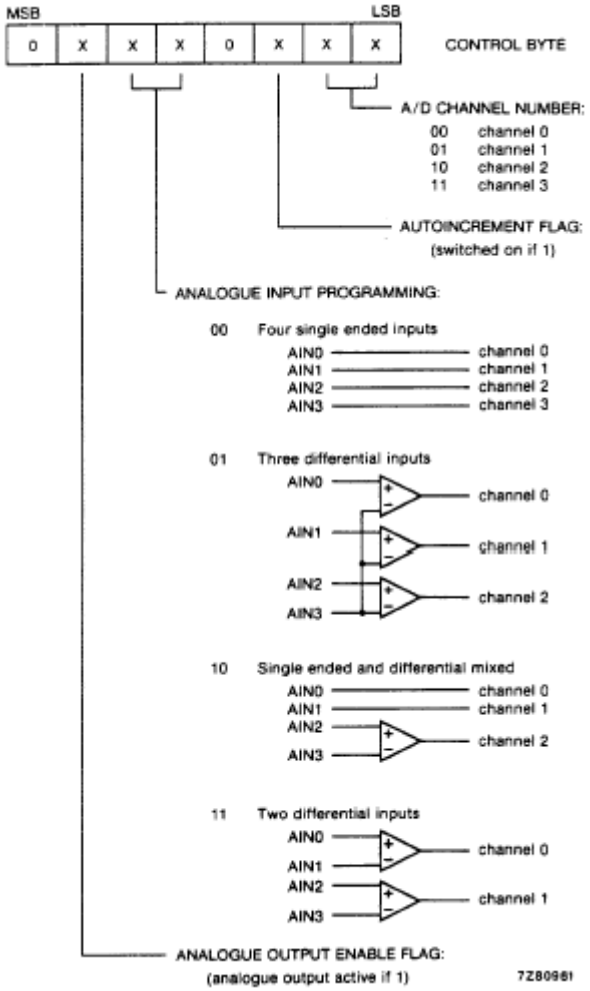


Figure 8-2:adressage du PCF 8591

Le CNA 8 bits permet de générer un signal de sortie analogique variable entre VSS et VDD (ses tensions d'alimentation).
Le CAN possède 4 voies analogiques indépendantes que l'on peut configurer de manière indépendante ou couplées si on veut disposer d'entrées analogiques différentielles (voir le rôle du registre CONTROL BYTE)

Registre de configuration de la partie CAN et CNA. CONTROL BYTE



Ce registre doit être configuré avant d'utiliser le circuit. Cette configuration s'effectue depuis le bus I2C en envoyant un octet du maître vers l'esclave avec le n° d'adresse de l'esclave (octet n°2)

Le bit 6 permet de valider le CNA

Les bits 5 et 4 permettent de configurer le mode des entrées du CNA . En effet, on peut travailler sur 4 voies asymétriques (mode 00) mais aussi différentielles (mode 01), 3 asym. Et 1 diff. (mode 10) et enfin 2 différentielles indépendantes (mode 11).

Le bit 2 permet de configurer le numéro de la voie du CAN en mode d'auto incrémentation (accès séquentiel aux voies du CAN)

Les bits 0 et 1 permettent de sélectionner le numéro de voie de la prochaine acquisition sur le CAN.

Fonctionnement du CAN/CNA 8 bits.

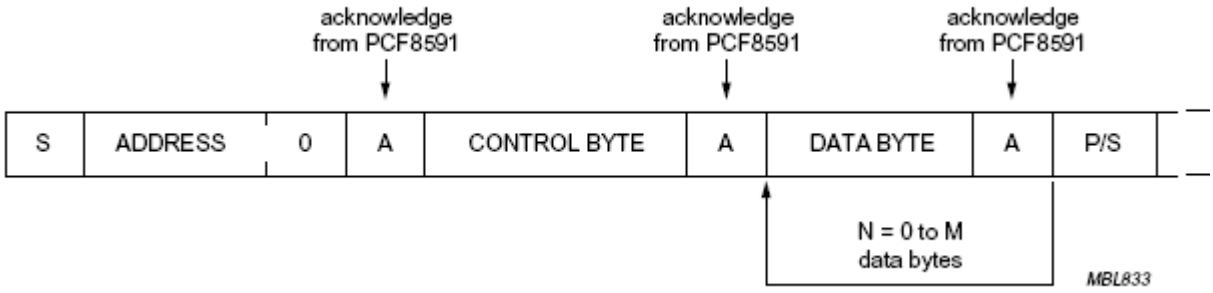


Figure 8-3 : trame de communication en vue de la modification du registre control byte et accès au CNA

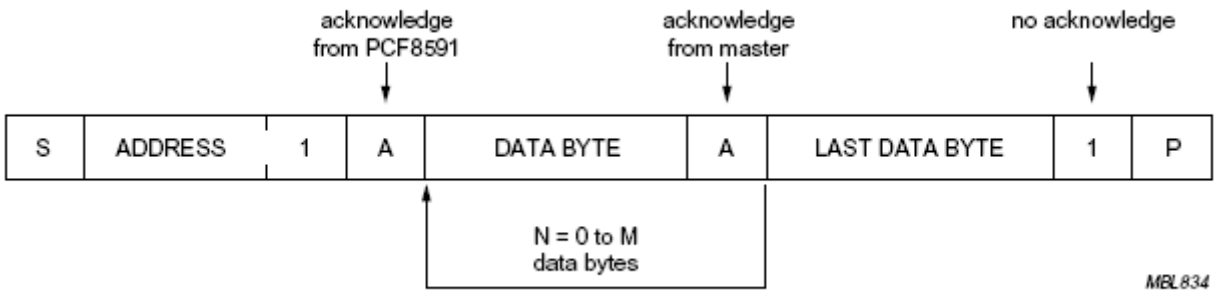


Figure 8-4 : trame de communication en vue d'une conversion A/N (accès au CAN)

Cette trame correspond à 3 octets transmis :

Octect1 : M>E, Adresse + R/W=0
Octect2: E>M, lecture de la précédente conversion A/N et échantillonnage d'une nouvelle donnée.
Octect3 : E>, lecture de la dernière conversion A/N

8.2. Réalisation de la commande d'un CAN I2C PCF8591

Il s'agit cette fois de réaliser une fonction permettant de mettre en œuvre la partie CAN du 8591.

```
Int Read_8591 (unsigned char * byChaine)
```

Acquisition sur les 4 canaux. Le résultat est fourni dans une chaine passée par pointeur

Cette fonction renvoie un code d'erreur concernant la communication I2C.

Pour ce faire, on doit initialiser le CONTROLBYTE pour configurer le CAN en mode :

- entrées indépendantes
- Mode d'auto incrémentation du n° de canal d'acquisition
- Le numéro du canal sera passé à la fonction n°1. Dans le cas de la fonction 2 les 4 canaux seront scannés.

Attention, dans le cas du mode auto incrément, la valeur renvoyée par le CAN correspond à celle du n° de canal précédemment en cours de sélection. Si on veut obtenir la nouvelle donnée convertie par le CAN (correspondant au n° de canal actuellement passé au CAN) il faut faire 2 accès en lecture dont on ne retiendra que le deuxième octet comme correspondant au dernier n° canal passé.

1) Créer la fonction de communication I2C avec le 8591 pour piloter le CAN et la stocker dans la bibliothèque précédente biblio_I2C.C/biblio_I2C.h
2) Faire tourner un programme de test S2_3.c qui permet de lire la valeur disponible sur la CANAL 0 et de l'afficher dans la console TELNET par un printf
3) Afficher dans le menu console les 4 paramètres mesurés et analyser leur valeur physique correspondante.
4) Faire valider votre travail par les enseignants

8.3. Affichage console des paramètres température, tension batterie, courant mesuré.

Testez votre fonction Read_PCF8591_multi() de manière à afficher dans le menu console les 4 paramètres analogiques que peut convertir le CAN de l'onduleur.
Convertir ces paramètres au moment de leur affichage pour afficher la consigne de gaz entre 0 et 100%, le courant entre -5A et +5A (résolution du capteur), Vbat en volts et la température en °c.

8.4. Commande de la vitesse de la roue à partir de la poignée de gaz.

Proposez une modification du code C de manière à ce que la consigne de PWM (entre 0 et 255) soit issue à présent de la positon de la poignée de gaz (entre 0 et 100%).
Il faudra rajouter deux options au menu pour tenir compte des butées minimales et maximales de la poignée.
Ces valeurs seront stockées de manière permanent dans le fichier chip.ini à l'aide des fonctions de l'API Beck :

BIOS_Set_Ini_String	Insert an entry into CHIP.INI.
BIOS_Get_Ini_String	Get an entry from CHIP.INI.

8.5. Commande de la vitesse de la roue à partir de la poignée de gaz avec gestion de la survitesse

Il s agit à présent de limiter la consigne de la poignée de gaz en fonction d'une bride de vitesse qui sera saisie par l'utilisateur et stockée de manière permanente dans le chip.ini du Beck.