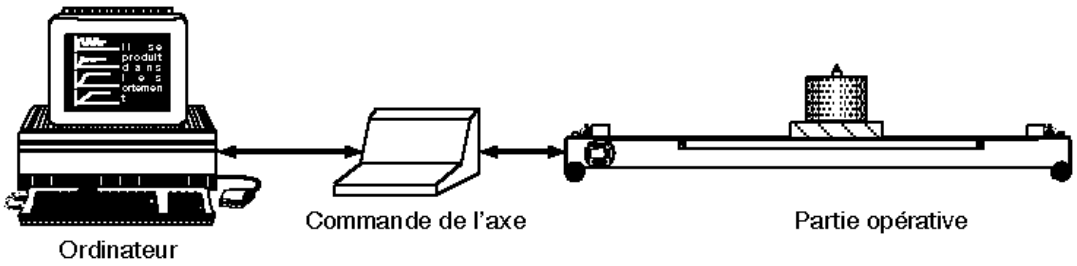
IF40. Final automne 2018

Documents non autorisés. Calculatrice autorisée.

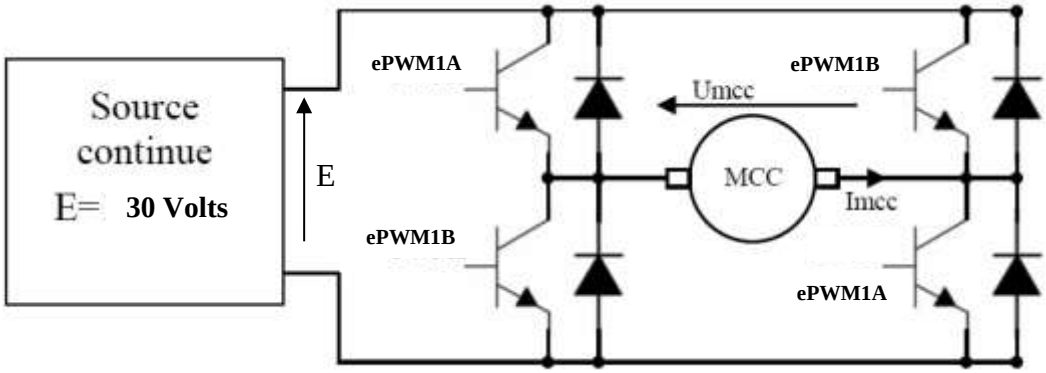
Les programmes que vous écrirez doivent être commentés.

Exercice 1 : axe horizontal

Dans une salle de TP, on met en place un axe de déplacement horizontal pour le modéliser. Le système est piloté par un DSP contrôleur TMS320C28335 cadencé à **150 MHz**.



L'axe est motorisé par un moteur à courant continu alimenté par un hacheur à partir d'une source continue (E). On donne ci-dessous le schéma de puissance :



Les transistors permettent de piloter la tension Umcc de -30V à 30V en ajustant α, le rapport cyclique des signaux ePWM1A et ePWM1B. La fréquence de ces signaux est de **10 kHz**. Les transistors requièrent un délai de sécurité (ou temps mort) de **1 µs**. La sortie ePWM1A est active au niveau bas.

1.1) **Écrire** en langage C la fonction initPWM réalisant l'initialisation du module ePWM1, des broches PWM utilisées, et l'initialisation des signaux PWM en mode asymétrique avec les délais de sécurité requis. Après l'initialisation, la valeur moyenne de Umcc doit être nulle.

Un capteur à effet Hall mesure l'intensité du courant Imcc. Cette sonde de courant délivre une tension proportionnelle à Imcc et variant de 0 à 3V pour un courant allant de 0 à 20A. Cette sonde est reliée à l'entrée ADCINA0 du convertisseur analogique numérique du DSP.

Un montage potentiométrique délivre une tension Ucom qui permet de piloter la vitesse du moteur à courant continu. Ucom donne la consigne de vitesse du moteur. Cette tension varie entre 0 et 2,5V. Elle est reliée à l'entrée ADCINA5 du convertisseur analogique numérique du DSP.

Pour mettre en place la régulation de vitesse, celle-ci doit être mesurée. On installe sur l'arbre moteur une dynamo tachymétrique associée à un montage à amplificateur opérationnel qui délivre une tension

continue U_T. La tension continue U_T qui varie entre 0 et 3V est appliquée à l'entrée ADCINA6 du convertisseur analogique numérique du DSP.

1.2) **Écrire** en langage C la fonction INITADC réalisant l'initialisation du module ADC en mode start/stop et séquenceur cascadé pour permettre la conversion de la voie 0, puis de la voie 5, puis de la voie 6, respectivement. La conversion sera déclenchée logiciellement.

1.3) **Écrire** en langage C la fonction MESURES qui réalise la conversion analogique numérique des voies 0,5 et 6 stocke les résultats dans les variables Imcc, Ucom et Ut respectivement (on considèrera ces variables déjà déclarées comme variables globales de type unsigned int).

Le moteur est mis en fonctionnement par la fermeture d'un contact de relais piloté par la broche GPIO0. Deux capteurs de fin de course sont positionnés de chaque côté de l'axe. Le capteur de fin de course droit est relié à la broche GPIO1 et le capteur de fin de course gauche est relié à la broche GPIO2. Un bouton départ de cycle est relié à l'entrée GPIO3. On doit configurer GPIO0 en sortie, GPIO1 GPIO2 et GPIO3 en entrée.

- GPIO0 au niveau logique 1 → le moteur peut fonctionner.
- GPIO0 au niveau logique 0 → le moteur est à l'arrêt.
- Fin de course gauche détectée → GPIO2 = 1 sinon GPIO2= 0
- Fin de course droite détectée → GPIO1 = 1 sinon GPIO1 = 0
- ↑ sur GPIO3 → départ de cycle.

1.4) **Écrire** en langage C la fonction INITPORT initialisant les GPIOs sachant que le moteur doit être à l'arrêt à la fin de l'initialisation.

Le cycle de fonctionnement est le suivant :Si GPIO0 =0 si l'axe est à gauche (GPIO2=1), après un front montant sur le bouton départ de cycle (↑ sur GPIO3) il se déplace vers la droite à une vitesse réglée par le potentiomètre. Lorsque l'axe arrive en fin de course à droite (GPIO1=1), il se déplace vers la gauche (toujours à la même vitesse) et revient en position initiale (fin de course gauche GPIO2=1).

Pour le déplacement vers la droite, la vitesse du moteur peut varier entre 0 et 3000 $tr.min^{-1}$ ce qui correspond à un rapport cyclique α de la tension Umcc variant entre 0,5 et 0,9. La valeur moyenne de Umcc est positive.

Pour le déplacement vers la gauche, la valeur moyenne de Umcc doit être négative. La vitesse du moteur peut varier entre -3000 $tr.min^{-1}$ et 0 $tr.min^{-1}$, ce qui correspond à un rapport cyclique α de la tension Umcc variant entre 0,5 et 0,1.

1.5) **Donner** le nom du registre permettant de faire varier le rapport cyclique α. **Calculer** ses valeurs correspondant aux vitesses de -3000 $tr.min^{-1}$, 0 $tr.min^{-1}$ et +3000 $tr.min^{-1}$. On tronquera les résultats si besoin.

La tension Ucom donne la consigne de vitesse **<u>en valeur absolue</u>**. Pour Ucom =0V, la valeur absolue de la vitesse demandée est de 300 $tr.min^{-1}$ (vitesse minimale) et pour Ucom = 2,5V, la vitesse demandée est de 3000 $tr.min^{-1}$. La fonction $|vitesse|$ en fonction de Ucom est une fonction affine proportionnelle: vitesse = f(Ucom).

1.6) **Calculer** les résultats N_0 et N_1 de la conversion analogique numérique correspondant respectivement aux valeurs de Ucom de 0V et 2,7V. On tronquera les résultats si besoin.

Une variable globale sens (type int) est utilisée pour calculer la valeur du rapport cyclique. Si sens=0, l'axe se déplace vers la gauche, sinon la valeur de sens=1 et l'axe se déplace vers la droite.

1.7) **Écrire** en langage C la fonction VITESSE qui utilise le résultat de la conversion analogique numérique correspondant à Ucom et la variable sens et qui renvoie la valeur à entrer dans le registre (question 1.6) permettant de faire varier la vitesse.

1.8) **Écrire** en langage C la fonction MOUVEMENT qui réalise le déplacement de l'axe de gauche vers la droite puis de droite vers la gauche décrit précédemment. Si l'une des deux conditions initiales n'est pas respectée, on sort directement de la fonction.

1.9) **Écrire** en langage C le programme principal qui réalise les initialisations et pilote le moteur en fonction de la vitesse. Ne pas oublier la déclaration des variables.

Exercice 2 : Interfaçage et décodage d'adresses

On donne page suivante le schéma partiel d'une carte 6809. Seuls sont représentés les mémoires et le microprocesseur ainsi que le bus d'adresse.

2.1) Pour les mémoires MEM1, MEM2, MEM3, MEM4 et MEM5, **indiquer** la capacité mémoire en bits et l'organisation en nombre de mots et taille des mots (N mots de M bits).

2.2) **Compléter** le tableau ci-dessous (DR1) en justifiant vos résultats, les tailles mémoires seront données en mots sur le document DR2.

Document réponse DR1.

	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
MEM1																
MEM2																
MEM3																
MEM4																
MEM5																

Document réponse DR2.

	Zone occupée		Zone utile (si ≠ zone occupée)		Zone de recouvrement			
	Adresse mini	Adresse maxi	Adresse mini	Adresse maxi	Adresse mini	Adresse maxi	Taille occupée	Taille utilisée (si≠)
MEM1								
MEM2								
MEM3								
MEM4								
MEM5								

Rappel : Table de vérité du 74LS138.

Entrées						Sorties							
Valid			Select										
G_1	$\overline{G_{2A}}$	$\overline{G_{2B}}$	C	B	A	$\overline{Y_0}$	$\overline{Y_1}$	$\overline{Y_2}$	$\overline{Y_3}$	$\overline{Y_4}$	$\overline{Y_5}$	$\overline{Y_6}$	$\overline{Y_7}$
0	X	X	X	X	X	1	1	1	1	1	1	1	1
X	1	X	X	X	X	1	1	1	1	1	1	1	1
X	X	1	X	X	X	1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	0	1	1	0	1	1	1	1	1	1
1	0	0	0	1	0	1	1	0	1	1	1	1	1
1	0	0	0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	0	0	1	1	1	1	0	1	1	1
1	0	0	1	0	1	1	1	1	1	1	0	1	1
1	0	0	1	1	0	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0