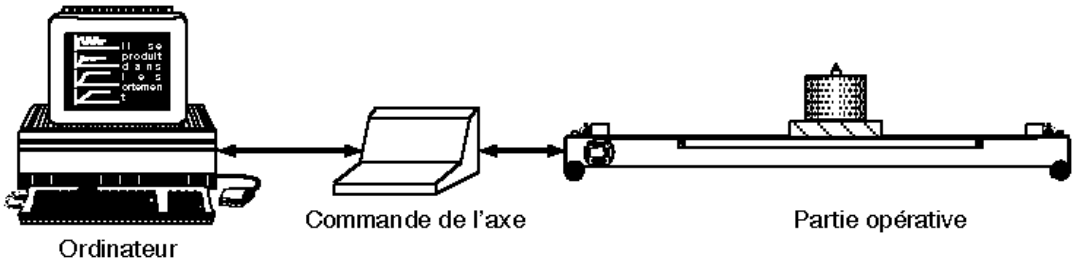
# IF40 Final automne 2017

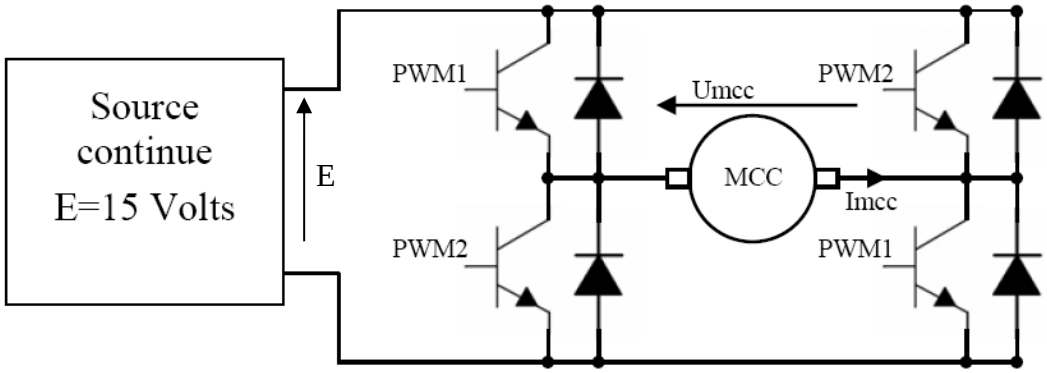
Calculatrices non autorisées. Fascicule DSP Contrôleur TMS320LF2407 non autorisé.
Traducteurs électroniques non autorisés. Cours et TD et tous documents non autorisés.

## 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 TMS320LF24LF07 cadencé à **40 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 -15V à 15V en ajustant $\alpha$, le rapport cyclique des signaux PWM1 et 2. La fréquence de ces signaux est de **20 kHz**. Les transistors requièrent un délai de sécurité (ou temps mort) de **4 µs**. La sortie PWM1 est active au niveau haut.

1.1) **Écrire** en langage C la fonction INITEVA réalisant l'initialisation du module EVA et des broches PWM utilisées, et l'initialisation des signaux PWM possédant une fréquence 20 kHz 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 30A. Cette sonde est reliée à l'entrée ADCIN1 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 2,7V. Elle est reliée à l'entrée ADCIN9 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 3,3V est appliquée à l'entrée ADCIN10 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 double séquenceur pour permettre la conversion de la voie 1 ou de la voie 9 et de la voie 10. La conversion sera déclenchée logiciellement. La calibration et le test du module ADC ne sont pas effectués.

1.3) **Écrire** en langage C la fonction MESURE_COURANT qui réalise la conversion analogique numérique de la voie 1 et renvoie le résultat de la conversion au programme principal.

1.4) **Écrire** en langage C la fonction MOUVEMENT qui réalise la conversion analogique numérique des voies 9 et 10 et renvoie ces deux valeurs au programme principal. Vous devrez utiliser des pointeurs car une fonction ne peut pas renvoyer deux valeurs. Les variables globales sont interdites.

Le moteur est mis en fonctionnement par la fermeture d'un contact de relais piloté par la broche IOPA1. 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 IOPA2 et le capteur de fin de course gauche est relié à la broche IOPA3. Un bouton départ de cycle est relié à l'entrée IOPA4. On doit configurer IOPA1 en sortie, IOPA2 IOPA3 et IOPA4 en entrée.

- IOPA1 au niveau logique 1 → le moteur fonctionne.
- IOPA1 au niveau logique 0 → le moteur est à l'arrêt.
- Fin de course gauche détectée → IOPA3 = 1 sinon IOPA3 = 0
- Fin de course droite détectée → IOPA2 = 1 sinon IOPA2 = 0
- ↑ (front montant) sur IOPA4 → départ de cycle.

1.5) **Écrire** en langage C la fonction INITPA initialisant le port A sachant que le moteur doit être à l'arrêt à la fin de l'initialisation.

Le cycle de fonctionnement est le suivant : après un front montant sur le bouton départ de cycle (↑ sur IOPA4) et si l'axe est à gauche (IOPA3=1), il se déplace vers la droite à une vitesse réglée par le potentiomètre. Lorsque l'axe arrive en fin de course à droite (IOPA2=1), il se déplace vers la gauche (toujours à la même vitesse) et revient en position initiale (fin de course gauche IOPA3=1). IOPA1 doit être égale à 1 pendant le déplacement. IOPA1 repasse à 0 lorsque l'axe est de retour à gauche

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 $\alpha$ de la tension Umcc variant entre 0,5 et 0,8. 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 $tr.min^{-1}$. La loi Umcc=f($\alpha$) est une fonction affine.

1.6) **Donner** le nom du registre permettant de faire varier le rapport cyclique $\alpha$. **Calculer** ses valeurs correspondant aux vitesses de -3000 $tr.min^{-1}$, 0 $tr.min^{-1}$ et +3000 $tr.min^{-1}$.

La tension Ucom donne la consigne de vitesse **en valeur absolue**. Pour Ucom =0V, la valeur absolue de la vitesse demandée est de 300 $tr.min^{-1}$ (vitesse minimale) et pour Ucom = 2,7V, la vitesse demandée est de 3000 $tr.min^{-1}$. La fonction $|vitesse|$ en fonction de Ucom est une fonction affine.

1.7) **Calculer** les résultats $N_0$ et $N_1$ de la conversion analogique numérique correspondant respectivement aux valeurs de Ucom (broche ADCIN9) de 0V et 2,7V.

Une variable 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.

1.8) **Écrire** en langage C la fonction VITESSE qui reçoit en paramètre 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 1.5) permettant de faire varier la vitesse.

1.9) **Écrire** en langage C le programme principal. 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.

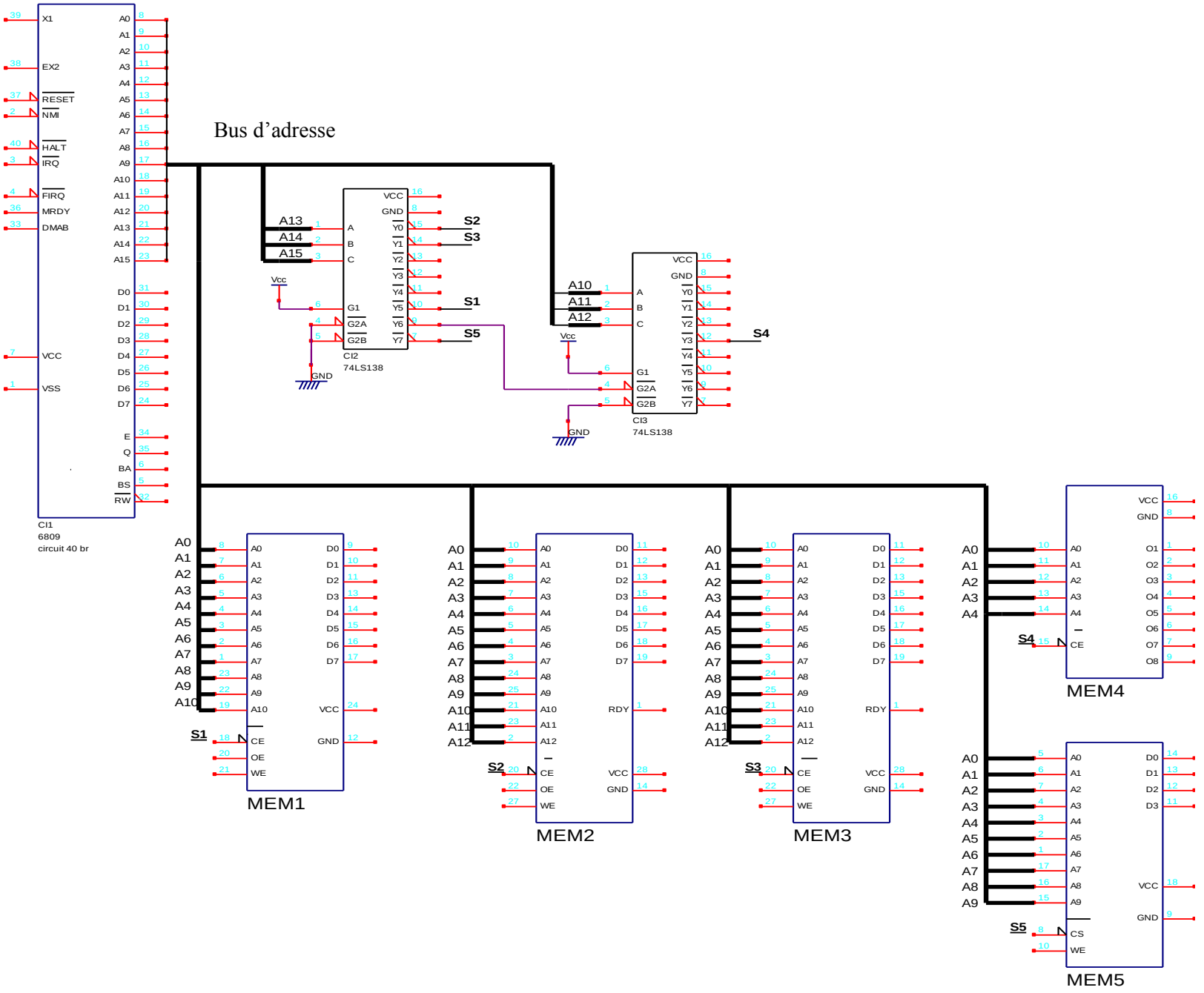
2.1) Pour les mémoires MEM1, MEM2, MEM3, MEM4 et MEM5, **indiquer** la capacité mémoire en octets, et l'organisation en nombre de mots et taille des mots (N mots de M bits).
**Présenter** le résultat sous forme de tableau.

2.2) **Compléter** le tableau ci-dessous en justifiant vos résultats sur votre copie.

| | 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 |


Bus d'adresse
CI1
6809
circuit 40 br
CI2
74LS138
CI3
74LS138
MEM1
MEM2
MEM3
MEM4
MEM5