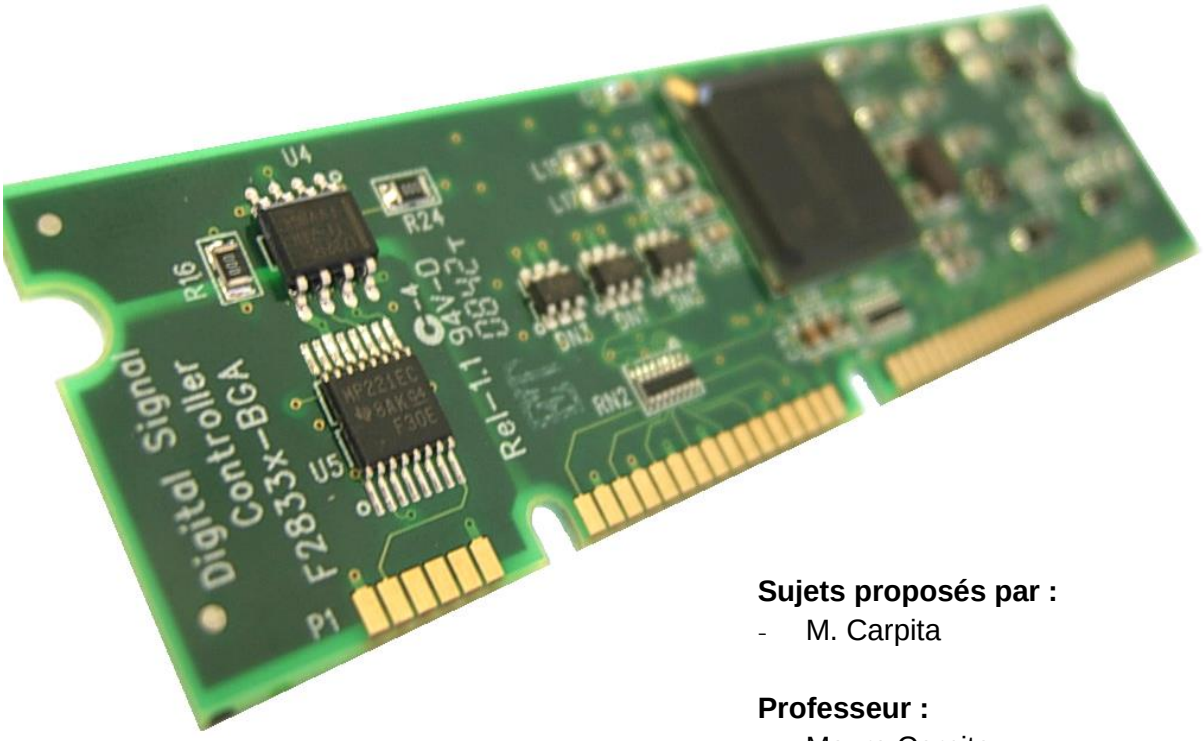
MA_POWELSYS

Onduleur triphasé en boucle ouverte



Sujets proposés par :
- M. Carpita

Professeur :
- Mauro Carpita

DONNÉE SÉANCE 2 : ONDULEUR TRIPHASÉ EN BOUCLE OUVERTE

1.1 Schéma de principe

Le schéma de principe de l'onduleur triphasé à étudier dans ce laboratoire est représenté à la figure 1.

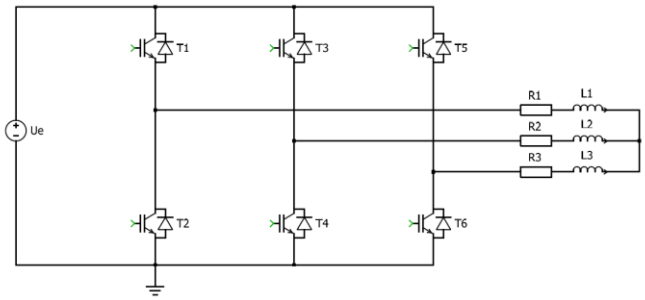


Figure 1. Onduleur triphasé sur charge ohmique/inductive

La charge est constituée d'une inductance de **30 mH** et d'une résistance de **20 Ω**. La tension d'alimentation vaut **110 V**.
Il va s'agir de faire circuler un courant sinusoïdal d'une valeur crête égale à **1.5 A** à une fréquence de **50 Hz**. La fréquence de commutation des IGBT en commande PWM est de **16 kHz.**

1.2 Programmation

Le but étant de compléter le code manquant, à partir de celui de la première séance, afin de faire fonctionner le système.

Les trois fichiers à compléter sont :

- main.c
- EPwm.c
- Gpio.c

Les instructions pour la modification du code sont dans la suite de la donnée. Le manuel technique de référence avec explication des modules EPwm et GPIO est disponible dans la documentation fournie.

1.3 Mesures

Les mesures sur l'installation didactique vont permettre de tester le code et de pouvoir le corriger si besoin est.

Temps à disposition :	Programmation :	1.5 heures
	Mesures :	1.0 heures

Programmation

Chargez le projet de la première séance dans Code Composer. Faites ensuite une copie de celui-ci dans le « Project Explorer » afin de pouvoir changer son nom. Fermez la version de la séance 1 en faisant un clic droit puis « Close Project ».

Pour ce nouveau laboratoire, vous allez devoir utiliser un nouveau fichier « main.c ». Pour commencer, il faut supprimer le « main.c » actuel. Ce nouveau fichier comporte certains changements tels que des ajouts d'*includes*, de *defines* sous *inputs/outputs,* de variables globales sous *Global vars* et modification des paramètres des fonctions *InitEPwm.* Ces changements sont essentiels pour la commande de l'onduleur. Ce fichier se trouve dans l'onglet des fichiers Teams liés au laboratoire n° 2.

Dans chacun des fichiers à modifier les parties à compléter sont identifiées avec le commentaire suivant :

```
//           *****************
//           *  A compléter  *
//           *****************
```

Les variables à utiliser sont déclarées dans le fichier main.c avant la fonction main dans la section :

```
/*-----------------------------------------------------------------------**
**                              Global vars                              **
**-----------------------------------------------------------------------*/
```

1. Modifier le fichier Main.c

1) Complétez la boucle *while* :

1.1) Sécurité :

Fréquence de sortie	0.0 < f_sin <100
Profondeur de modulation	0.0 < ma < 1.0
Fréquence de la porteuse	500.0 < f_porteuse < 25000.0
Temps d'anti-chevauchement	2.0 < deadband < 5.0
Déphasage entre les porteuses	0.0 < phase < 180.0

Attention : ne JAMAIS descendre le temps d'anti-chevauchement en dessous de 2µs, sinon l'IPM sera détruit !!!

1.2) Modification PWM (voir documentation sur les modules ePWM).

L'objectif est que le signal PWM se modifie lorsque ses paramètres (fréquence et temps d'anti-chevauchement) sont changés.

2) Complétez la routine d'interruption de l'ADC pour générer le signal de commande Ucm et générer les signaux PWM adéquats :

Pour créer le signal PWM, il faut comparer le signal sinusoïdal Ucm au signal triangulaire de la porteuse Uh. La porteuse est générée automatiquement par le DSP, par contre le signal sinusoïdal doit être échantillonné et introduit dans le registre de comparaison par itération. L'échantillonnage d'Ucm est donc fait au cours de l'interruption de l'ADC, laquelle est synchronisée avec la porteuse.

Travail à faire : Programmez la routine de l'ADC à l'aide du flowchart de la Figure 2.

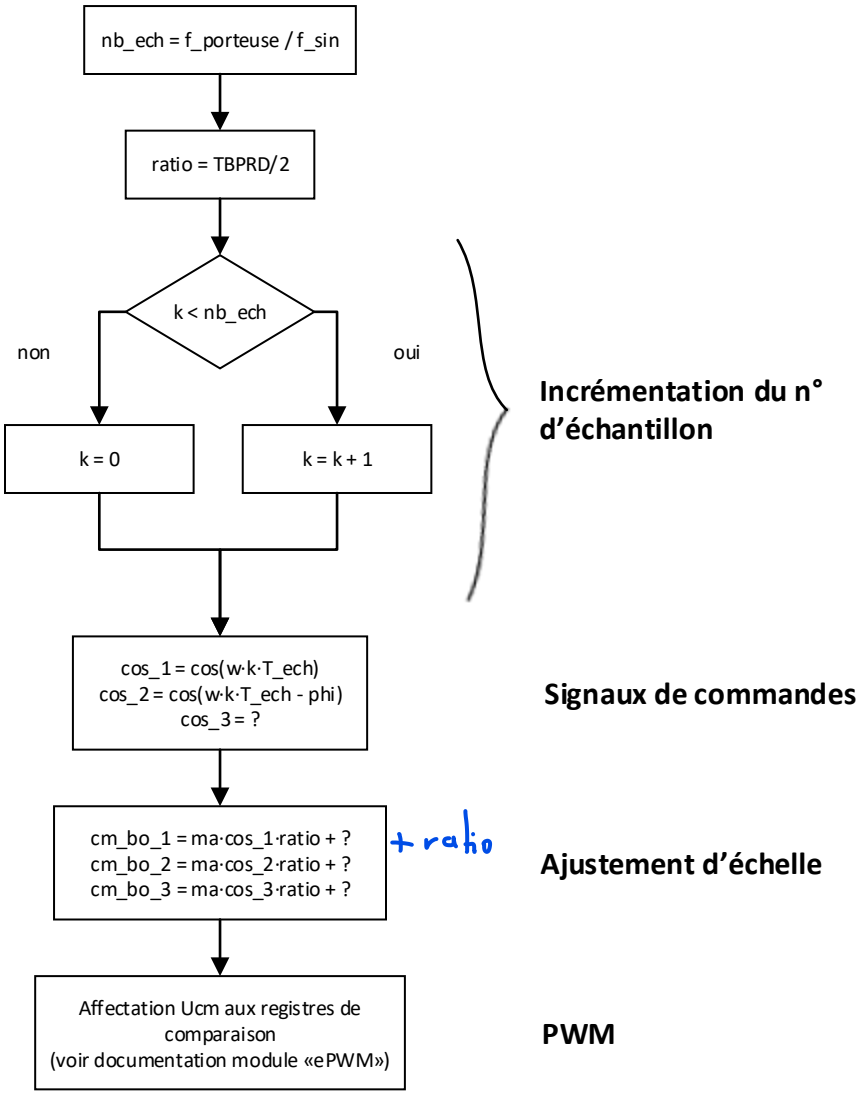


Figure 2. Flowchart des instructions de l'interruption ADC

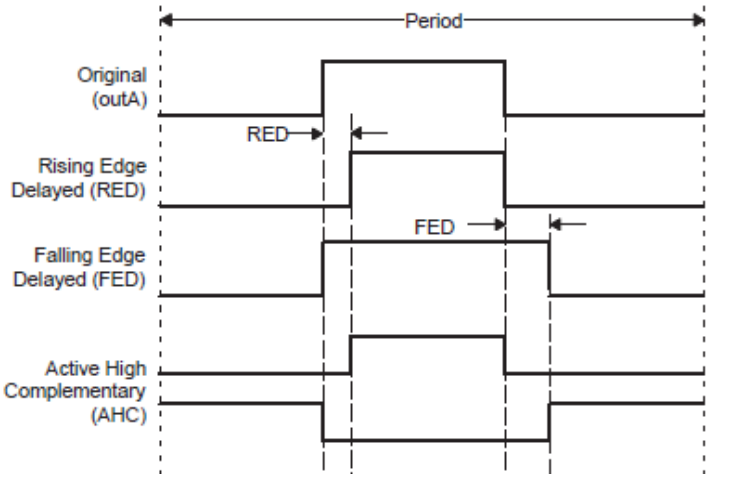
2. EPwm.c

1) Modifiez, pour les PWM1, 2 et 3, le registre TBCTL pour que le compteur s'incrémente positivement (bit 13) et afin de créer un signal triangulaire symétrique (bit 0-1). Pour les PWM 2 et 3, modifiez les bits 2,4 et 5 afin qu'ils se synchronisent sur le PWM1.

2) Modifiez, pour les PWM1, 2 et 3, le registre AQCTLA pour créer un PWM symétrique par rapport à la valeur crête du signal triangulaire (bits 4 à 7).

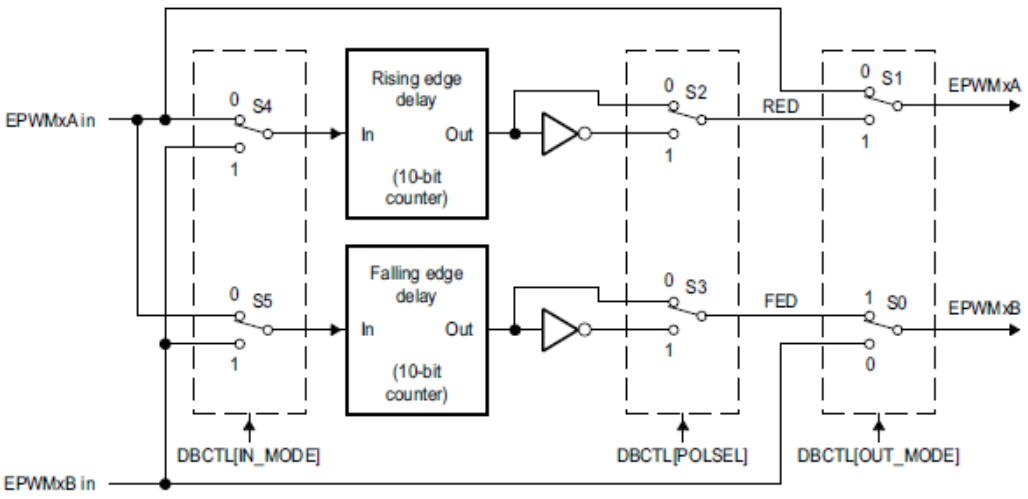
3) Modifiez, pour les PWM1, 2 et 3, le registre DBCTL afin d'obtenir leurs PWM complémentaires avec un temps d'anti-chevauchement (voir la construction du signal AHC à la figure 32 ci-dessous).

Figure 32. Dead-Band Waveforms for Typical Cases (0% < Duty < 100%)



Utilisez les informations fournies par la figure 31 ci-dessous.

Figure 31. Configuration Options for the Dead-Band Submodule



3. Gpio.c

Modifiez les GPIO 0 à 5 pour qu'elles soient utilisées en PWM.

4. Test du système complet

1) Branchez les charges R-L et l'alimentation DC au système CESAR comme montré à la Figure 1.
2) Connectez le PC et la carte de commande avec la sonde JTAG (câble USB) et le câble RS232.
3) Installez les sondes de mesure de tension et courant.
4) Testez l'installation à l'aide de Code composer, puis faire les mesures nécessaires (tension de branche, de phase, courant de charge, …).
5) Faites varier la fréquence de commutation, le temps d'anti-chevauchement (2 < ta < 6 µs), et la profondeur de modulation. Evaluez leur effet sur le courant dans la charge et les tensions.