TP1 PIC32MX
PWM et A/D

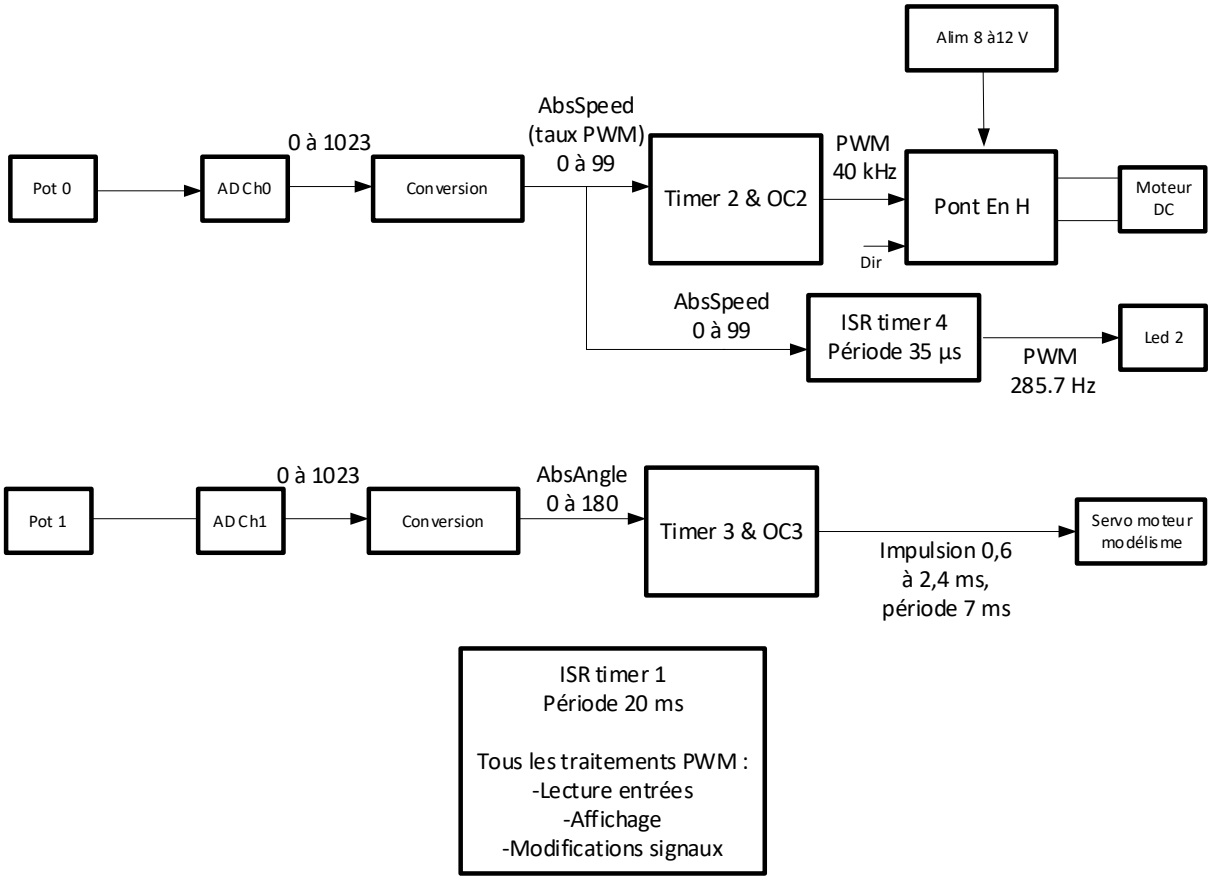
OBJECTIFS

Réalisation de signaux PWM sur la base de 3 approches différentes :

- Utilisation du module OC2 en mode PWM avec le timer 2 pour commander la rotation d'un moteur DC via le pont en H.
- Utilisation du module OC3 en mode génération d'impulsion avec le timer 3 pour générer un 2ème signal PWM adapté pour un servomoteur de modélisme.
- Génération sur base d'une interruption cyclique (timer 4) d'un 3ème signal PWM.

Utilisation du convertisseur A/D intégré du PIC dans le but de générer des signaux dont le rapport cyclique varie en fonction des tensions mesurées.
L'afficheur LCD sera utilisé pour afficher la valeur du rapport cyclique en % et l'angle en degré pour le servomoteur. Le signal fourni par OC2 est prévu pour varier la vitesse d'un moteur DC, tandis que le signal issu de OC3 doit varier l'angle du servomoteur.

VUE D'ENSEMBLE DE LA REALISATION



AFFICHAGE INITIAL

Au démarrage il est demandé d'afficher durant 3 secondes :

```
TP1 PWM 2016-2017
Nom 1
Nom 2
```

PRINCIPE DE LA REALISATION

Le projet mobilise 4 timers. Adaptation aux broches imposées par le Kit PIC32MX795F512L avec utilisation du pont en H pour commander un moteur DC.

ROLE DU TIMER 1

Le timer 1 doit être programmé de manière à générer une interruption cyclique avec une période de 20 [ms]. Niveau de priorité 4.

Dans la routine de réponse à l'interruption du timer 1, on effectue l'ensemble des traitements, en appelant une séquence de fonctions (implémentées dans le fichier GestPWM.c)**.**
☹ Remarque : cette approche n'est pas une solution à appliquer pour obtenir un système optimal (à cause de la durée de l'affichage), mais elle permet de mettre en évidence le besoin de gérer les priorités d'interruptions.

Séquence d'appel des fonctions :

- GPWM_GetSettings() Obtention vitesse et angle
- GPWM_DispSettings() Affichage
- GPWM_ExecPWM() Execution PWM et gestion moteur.

Actions de la fonction GPWM_GetSettings() :
Traitement pour le 1er AD :

- Lecture valeur canal 0 de l'AD.
- Moyenne glissante sur les 10 dernières valeurs afin de limiter la fluctuation de la consigne.
- Calcul d'une valeur variant de 0 à 198.
- Obtention à partir de cette valeur de l'information speed variant de -99% à 99% et de absSpeed variant de 0 à +99

	Pot. au minimum	Pot. au milieu	Pot. au max
Valeur brute A/D	0	(511) 512	1023
Valeur 0 à 198	0	99	198
Vitesse signée	-99	0	99
Vitesse ABS ou taux PWM	99	0	99

Traitement pour le 2ème AD :

- Lecture valeur canal 1 de l'AD.
- Moyenne glissante sur les 10 dernières valeurs afin de limiter la fluctuation de la consigne.
- Calcul d'une valeur variant de 0 à 180 (champ absAngle).
- Obtention à partir de cette valeur de l'angle signé de -90% à 90%.

	Pot. au minimum	Pot. au milieu	Pot. au max
Valeur brute A/D	0	(511) 512	1023
Angle pour calculs	0	90	180
Angle signé (affichage)	-90	0	+90

Actions de la fonction GPWM_DispSettings() :

- Affichage du champ speed (vitesse signée de -99 à +99 sur la 2ème ligne de l'afficheur LCD.
- Affichage du champ absSpeed (vitesse absolue de 0 à +99 sur la 3ème ligne de l'afficheur LCD.
- Affichage du champ angle (angle signé de -90 à +90 sur la 4ème ligne de l'afficheur LCD.

```
TP1 PWM 202X-202Y
SpeedSetting SSS
absSpeed      NN
Angle        AAA
```

Actions de la fonction GPWM_ExecPWM() :

- Gestion de la direction du pont en H en fonction du signe de la vitesse.
- Calcul de la valeur du nombre d'impulsions pour OC2 (à partir de absSpeed) afin d'obtenir le rapport cyclique voulu, valeur à fournir à la fonction PLIB_OC_PulseWidth16BitSet().
- Calcul de la valeur du nombre d'impulsions pour OC3 (à partir de absAngle) afin de générer une impulsion dont la largeur est proportionnelle à l'angle en utilisant la fonction PLIB_OC_PulseWidth16BitSet().

ROLE DU TIMER 2 (PWM "HARDWARE")

Le module OC2 doit être configuré pour travailler en PWM, en générant un signal d'une fréquence de 40 kHz. Le timer 2 sert à établir la période du signal PWM. Il n'y a pas besoin d'une interruption.

ROLE DU TIMER 3 (PWM PAR IMPULSION)

Le module OC3 doit être configuré pour travailler en modulation de largeur d'impulsion en relation avec le timer 3, en générant des impulsions d'une largeur modulable entre 0,6 ms et 2,4 ms. Cette impulsion doit être déclenchée toutes les 7 ms (choix pour garantir le cycle maximum de 20 ms et avoir une bonne résolution).

Le timer 3 sert de base de temps pour la génération de l'impulsion et de sa répétition. Avec une période de répétition de 7 ms, on n'utilise qu'une partie de la période pour l'impulsion. Il n'y a pas besoin d'une interruption.

ROLE DU TIMER 4 (PWM SOFTWARE)

Le timer 4 doit être configuré pour générer une interruption toutes les 35 µs. Priorité d'interruption 4 au départ.

On appelle la fonction GPWM_ExecPWMSoft() dans la réponse à l'interruption.

La fonction GPWM_ExecPWMSoft() doit générer par comptage un signal PWM d'une période de 3.5 ms (100 cycles de 35 µs) donc une fréquence de 285.7 Hz. Le rapport cyclique sera lui aussi réalisé par comptage des interruptions. La variation du rapport cyclique se fait ainsi par pas de 1% = 35 µs.

La valeur absSpeed est directement utilisée pour le rapport cyclique. Cependant, c'est dans la réponse à l'interruption du timer 4, que **la nouvelle valeur du rapport cyclique sera prise en compte et ceci uniquement à la fin de la période**.

Le signal est généré sur un port standard, nous utiliserons LED2.

REGLAGE PRIORITES DES INTERRUPTIONS

- Dans une 1ère phase, on configure le même niveau de priorité pour le timer 1 et le timer 4. On prendra le niveau 4 pour les 2 avec la même sous-priorité de 0.
  En utilisant les marqueurs (détails en p.6), on observera, la relation entre les 2 interruptions.

- Dans une 2ème phase on configure des niveaux de priorités différents pour le timer 1 et le timer 4. On conserve le niveau 4 pour le timer 1, par contre on prend le niveau 7 pour le timer 4.
  En utilisant les marqueurs (détails en p.6), on observera, la relation entre les 2 interruptions.

DETAILS DE LA REALISATION

La réalisation se fait en 2 étapes :

1) Mise en place du projet avec le MHC.
2) Ajout et adaptations nécessaires pour obtenir le comportement souhaité.

MISE EN PLACE DU PROJET AVEC LE MHC

On utilise le même principe que pour le TP0 en se référant aux chapitres 2, 4 et 5 du cours MINF T.P.

Utilisation du BSP pic32mx_skes

Au niveau de la configuration du projet, il faut introduire les 4 instances statiques de drivers de timer avec et sans interruption. Il faut ajouter les 2 OC Driver.

Nom du projet : TP1_TimerPWM à placer sous C:\microchip\harmony\v<n>\apps\MINF\TP.

Les canevas des fichiers GestPWM.h et GestPWM.c sont sous : ...\Maitres-Eleves\SLO\Modules\SL229_MINF\TP\TP1_TimerPWM\Fichiers_TP1.

ADAPTATION DU PROJET

L'essentiel de l'adaptation est réalisée au niveau des fichiers app.c, GestPWM.h et GestPWM.c. Il y aura également d'autres modifications.

MODIFICATION DU FICHIER APP.H

Il faut ajouter APP_STATE_WAIT=1 dans le type énuméré APP_STATES.

Ainsi que le prototype de la fonction :

```
void APP_UpdateState ( APP_STATES NewState ) ;
```

MODIFICATION DU FICHIER SYSTEM_INTERRUPT.C

Il est demandé d'ajouter dans la routine de réponse à l'interruption du timer 1 (cycle 20 ms) un mécanisme qui établit APP_STATE_SERVICE_TASKS après 3 secondes (150 cycles) puis par la suite effectue à chaque cycle la séquence d'appel des fonctions :

- GPWM_GetSettings() Obtention vitesse et angle
- GPWM_DispSettings() Affichage
- GPWM_ExecPWMs() Exécution PWM et gestion moteur.

Ces 3 fonctions ont pour paramètre un pointeur sur la structure S_pwmSettings (définition fournie dans GestPWM.h).

Il est demandé d'ajouter dans la routine de réponse à l'interruption du timer 4 (cycle 35 µs) l'appel la fonction GPWM_ExecPWMSoft() à chaque cycle.

MODIFICATION DU FICHIER APP.C

Il faut ajouter l'implémentation de la fonction APP_UpdateState.

Au niveau de la fonction APP_Tasks, dans le *switch (appData.state)*, on introduira les traitements suivants :

Au niveau du **case APP_STATE_INIT**:

- Initialisation du LCD
- Affichage du message initial.
- Initialisation de l'AD
- Eteindre toutes les leds
- Appel de la fonction GPWM_Initialize(), qui starte les timers et les OC puis appelle encore la fonction BSP_EnableHbrige().
- Etat suivant = APP_STATE_WAIT

Au niveau du case **APP_STATE_WAIT**:

- aucune action !

Au niveau du case **APP_STATE_SERVICE_TASKS:**

- Etat suivant = APP_STATE_WAIT

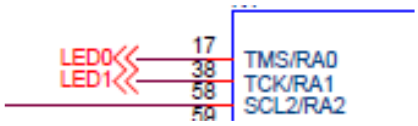
☝ Permet une évolution, afin d'appeler des fonctions d'exécution dans le futur.

OBSERVATION DES DUREES D'INTERRUPTIONS

En utilisant une sortie de libre, il est possible en mettant à 1 la sortie tout au début de la réponse et à 0 tout à la fin, d'obtenir une mesure à l'oscilloscope de la durée de l'interruption. Pour observer les durées des interruptions du timer 1 et du timer 4, il faut utiliser les sorties suivantes du kit PIC32MX :

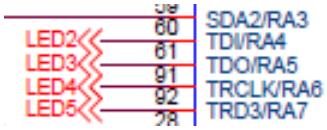
BROCHES POUR LES MARQUEURS

On utilisera LED0 et LED1.

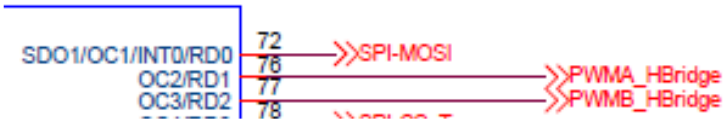


BROCHES DES SORTIES OCX ET PWM SOFT

Pour la sortie PWM soft on utilisera LED2.

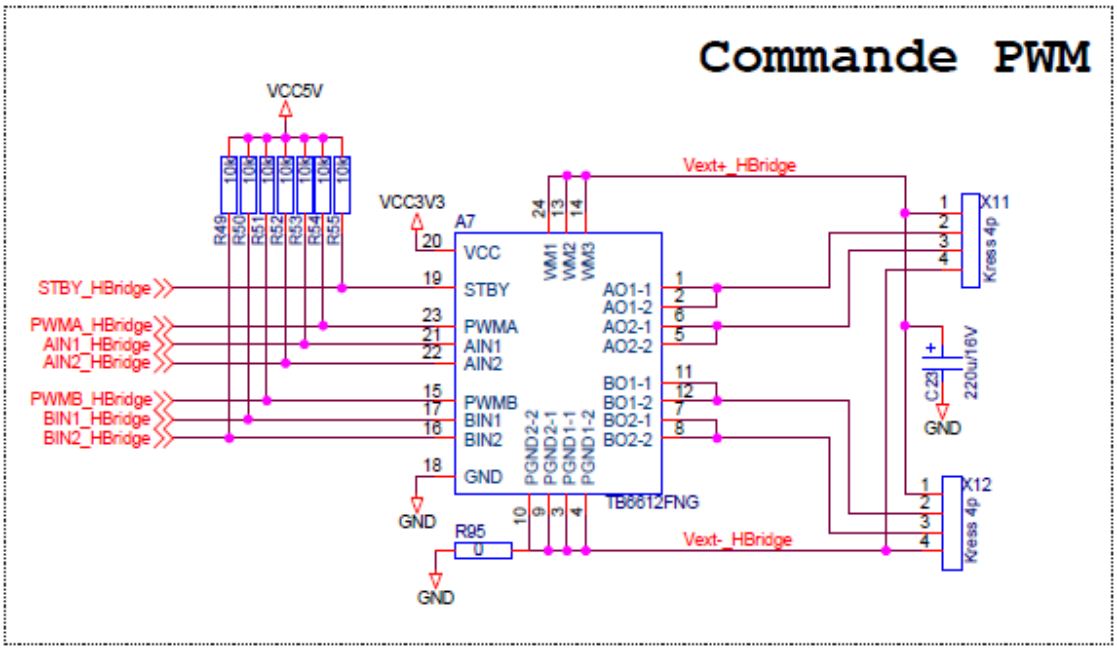


Pour le signal de commande du moteur, on utilise OC2/RD1 broche 76.
Pour le signal de commande du servomoteur, on utilise OC3/RD2 broche 77.



COMMANDE DU MOTEUR AVEC PONT EN H

Le kit PIC32MX dispose d'un double pont en H, dont voici le schéma.



Le signal PWM fourni par OC2 arrive sur le pont en H canal A, nommé PWMA_Hbridge. Pour le sens de rotation il faut se référer au tableau ci-dessous pour activer AIN1_HBridge, AIN2_HBridge et STBY_HBridge.

Input				Output		
IN1	IN2	PWM	STBY	OUT1	OUT2	Mode
H	H	H/L	H	L	L	Short brake
L	H	H	H	L	H	CCW
		L	H	L	L	Short brake
H	L	H	H	H	L	CW
		L	H	L	L	Short brake
L	L	H	H	OFF (High impedance)		Stop
H/L	H/L	H/L	L	OFF (High impedance)		Standby

Pour permettre la rotation du moteur, il faut une alimentation externe à régler entre 10 et 12 V. Les moteurs fournis consomment environ 0.3 A ce qui convient bien au pont en H.

TRAVAIL ET ETABLISSEMENT DU RAPPORT

Le travail s'effectue par groupe de deux en partageant la réalisation, ce qui permet d'acquérir des compétences en collaboration dans la réalisation d'un programme.

Le rapport doit contenir au minimum les éléments suivants :

- Explications complémentaires et détaillées sur les valeurs de configuration des timers et modules OC, valeurs pour l'établissement des rapports cycliques, ainsi que les formules de conversion de la valeur mesurée sur le convertisseur en rapport cyclique (explications sur base d'extraits du programme).
- Contenu des fichiers app.c et GestPWM.c.
- Contenu du fichier system_interrupt.c
- Observations de l'effet sur les marqueurs d'interruptions lorsque les deux interruptions sont au même niveau de priorité, puis lorsque les priorités sont réglées.
- Observations de la durée des réponses des 2 interruptions dans les 2 situations de réglage de priorité.
- Le fonctionnement doit être démontré et des copies d'écran de l'oscilloscope montrant les 2 signaux PWM avec un rapport cyclique de 5%, 50 % et 95% doivent être fournies. La qualité des mesures doit être suffisante pour vérifier avec précision la valeur du rapport cyclique des 2 signaux. On se base sur l'affichage sur le LCD pour l'indication du % du rapport cyclique.
- Montrez la situation, pour les 2 signaux PWM, lorsque le rapport cyclique est de 0. Explications souhaitées du pourquoi du comportement.
- Le fonctionnement doit être démontré avec le servomoteur et avec des copies d'écran de l'oscilloscope montrant les signaux avec un angle de -90 deg, 0 deg et +90 deg. La qualité des mesures doit être suffisante pour vérifier avec précision la durée de l'impulsion.
- Démonstration de la commande du moteur avec le signal PWM rapide avec gestion du signe pour commander l'inversion du sens de rotation.
- Résumé des mesures, précisions des signaux, diverses remarques et conclusion.

Votre travail sera évalué sur la base de :

- Qualité, facilité de réutilisation/modification et taux d'aboutissement du code.
- Fonctionnement et taux d'aboutissement.

DUREE DE LA MANIPULATION

A réaliser en 4 séances.