Introduction aux DSPIC

But de la manipulation

L'objectif de cette manipulation est d'introduire un microcontrôleur d'une famille couramment utilisée : les PIC de Microchip. Quelques systèmes simples basés sur des entrées-sorties seront réalisés.

Par la suite, nous verrons comment réaliser un programme régi par le temps et introduirons la notion d'interruption

Prérequis

Avant d'entrer au laboratoire, il est conseillé de lire les chapitres suivants :

- Chapitre 5 : Entrées-sorties numériques
- Chapitre 4 : Timers

Il est encore conseillé de lire les sections 1 à 5 du complément « Programmation d'une carte à microcontrôleur »

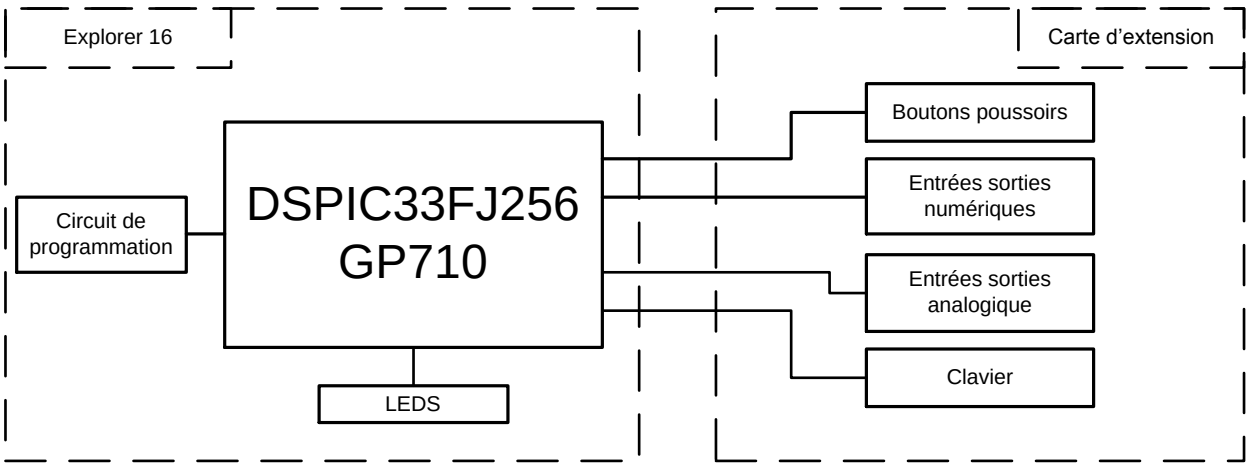
Objectifs

A la fin de ce laboratoire vous devez être capables :

- De réaliser un programme simple pour microcontrôleur
- D'expliquer la notion d'entrées-sorties ainsi que la sortance
- De faire intervenir des éléments asynchrones dans votre programme par le biais de *timers* et d'interruptions.

1. Introduction

Au long des six laboratoires qui illustrent le cours, vous serez amenés à utiliser une carte à microcontrôleur. En plus de la programmer, vous devrez comprendre le fonctionnement de ses différents périphériques et l'interfacer avec le monde extérieur.



Ce premier laboratoire a pour but de vous faire prendre en main la carte et d'utiliser quelques périphériques de base. Vous apprendrez à interagir de manière simple avec le microcontrôleur de la carte.

Le fonctionnement de la carte ainsi que du dsPIC est expliqué dans le complément « Programmation d'une carte à microcontrôleur ».
Pour les deux premiers labos, il vous est conseillé d'utiliser en particulier la section 5 « Première prise en main »

2. Programme basique : utilisation des entrées-sorties

Votre premier programme consistera en l'interfaçage de deux types d'entrées-sorties : les boutons poussoirs et les LEDS. Le cahier des charges est simple : on désire que l'une des 8 LEDS présentes sur la carte s'allume lorsque l'on appuie sur un bouton au choix.

- Localisez les différents périphériques présents sur la carte fournie et cités dans le complément « Programmation d'une carte à microcontrôleur » (LEDs, boutons poussoirs, potentiomètre, sortie analogique connectée à l'ampli de puissance,…).
- Réalisez un schéma bloc de votre système. En particulier, indiquez quels éléments sont numériques, lesquels sont analogiques et lesquels servent de traducteurs entre les deux milieux. Quels périphériques sont utilisés ?
- A l'aide du complément réalisez un programme répondant au cahier des charges. Pour ce faire, vous serez amenés à configurer les registres TRIS des pattes d'entrée-sortie. Un fichier contenant un canevas de programmation vous sera fourni.

Après coup, on décide de plutôt utiliser une LED ultrabrillante consommant 20mA sous 3.2V. Celle-ci n'étant pas présente sur la carte, vous devrez la connecter à une des bornes de la carte d'extension (voir annexe pour plus de détail).

- Pourquoi une connexion directe entre le dsPIC et la LED ne fonctionne-t-elle pas ? Faites le lien avec la notion de sortance statique.
- Expliquez en quoi l'utilisation d'un circuit du type *buffer* résout le problème. Tracez le nouveau schéma-bloc de votre système.

Nous allons utiliser un circuit buffer 74ACT244 dont la notice est fournie en annexe.

- Vérifiez que ce circuit respecte bien les normes TTL 5V
- Réalisez un montage sur protoboard permettant d'allumer la LED. N'oubliez pas de dimensionner la résistance de limitation du courant dans la diode. Alimentez le buffer en 0V/5V
- Modifiez votre programme pour qu'une pression sur le bouton allume désormais la LED ultrabrillante externe au lieu de celles présentes sur la carte.

3. Utilisation du Timer

On vous demande de faire clignoter des LEDs à une fréquence donnée.
Pour ce faire, vous aurez besoin d'utiliser un *timer*. Dans un premier temps, vous allez écrire un programme permettant de faire commuter la sortie RC1 à une fréquence donnée de 10kHz

- Quelle valeur doit contenir le registre de période PR du timer pour que ce dernier déborde à la bonne fréquence ? Quelle est la période maximale du timer ? Pour rappel, le processeur exécute des instructions au rythme de 40MHz
- Vérifiez que l'appel à la fonction clav2LCD est bien mis en commentaire
- Ajoutez à votre code les lignes nécessaires à la configuration et au lancement d'un timer (au choix)
- Dans la boucle principale de votre programme,
    - vérifiez par *polling* si votre timer est arrivé à son terme
    - Ecrivez une routine permettant de faire commuter la patte d'I/O RC1 à la fréquence de débordement. Cette routine ne peut pas être bloquante : on doit par exemple pouvoir continuer à allumer la LED ultrabrillante à tout moment en appuyant sur un bouton
    - N'oubliez pas de remettre à zéro le bit TxIF dans votre routine
- Vérifiez à l'oscilloscope que la fréquence est bien celle attendue
- Modifiez le programme de sorte à ce que la période soit maintenant de 500ms, et programmez le clignotement des LEDs

4. Utilisation d'une interruption

Jusque maintenant, nous avons à chaque fois considéré que la boucle principale du main était courte et s'exécutait donc en un temps limité. Nous allons maintenant étudier comment une routine longue peut perturber le fonctionnement du reste du programme

- Dé-commentez l'appel à la fonction clav2LCD dans la boucle principale. Cette fonction scanne le clavier alphanumérique de la carte d'extension et affiche la touche enfoncée sur l'écran LCD. Pour des raisons pratiques expliquées dans le guide programmation, la lecture du clavier prend un temps non négligeable (jusqu'à 20µs)
- Configurez le timer pour qu'il déborde toutes les 100µs, et faites basculer la sortie RC1 à chaque débordement
- Observez la sortie RC1 à l'oscilloscope. En quoi les performances sont-elles dégradées ? Expliquez l'origine du problème

Un moyen simple de supprimer ce problème est de configurer le µC de sorte à ce que le traitement du timer. Ceci est réalisé par le biais d'interruptions : lorsqu'un événement spécifique survient (ici, le débordement du timer), le µP interrompt la tâche en cours et appelle une fonction spécifique nommée routine de service d'interruption (ISR).

L'avantage de cette méthode est double :

- Il n'est plus nécessaire de faire de test programmé dans la boucle infinie principale
- Les événements sont traités en priorités par rapport à la tâche de fond.

Vous allez maintenant modifier votre programme pour que le traitement du timer soit réalisé par une routine d'interruption.

- Lisez la section 5.4 de guide de programmation, consacrée aux interruptions
- Activez l'interruption liée au timer choisi
- Ecrivez la routine d'interruption associée. Elle s'écrit comme une fonction, si ce n'est que son nom est imposé : void _ISR _TxInterrupt(void) ; Le x doit être remplacé par votre timer. Tout le traitement du timer doit maintenant être réalisé dans la routine
- Vérifiez que le problème a bien été résolu en branchant l'oscilloscope sur la borne RC1
- Concluez quant à l'utilisation des interruptions