DUT Génie Électrique – Département GEII2

Génie Electrique – Semestre 1

II1 - Travaux pratiques

Informatique Industrielle – Utilisation de ports E/S

Auteur(s) : Y. Guinand, H. Discours, F. Wilmot
Date – Version : Novembre 2023

LISTE DES OBJECTIFS
Mettre en œuvre une carte microcontrôleur à base de PIC18F4550.
Utiliser les outils de développement associés, avec programmation en langage C.
Utiliser les ports d'E/S du microcontrôleur. Se servir de la notion de masque pour accéder indépendamment aux différents bits d'un port.
Réaliser le câblage de broches d'entrée TOR du microcontrôleur (configuration pull-up ou pull-down).
Réaliser le câblage de broches de sortie TOR du microcontrôleur (configuration source ou sink).
LOGICIEL ET MATERIEL UTILISES
Carte microcontrôleur « Gamel Trophy » IUT de Cachan à base de microcontrôleur PIC18F4550
Logiciels MPLABX IDE de Microchip et compilateur XC8 de Microchip
Programmateur PicKit3
Carte d'extension IUT Cachan avec broches E/S et LEDs
CONSIGNES
La partie PREPARATION est à traiter avant la séance de TP. Elle sera évaluée par le professeur avant de commencer le TP.
Un compte rendu MANUSCRIT par personne est à rédiger pendant la séance. Ce compte rendu sera à remettre au professeur en fin de séance.

1. Préparation

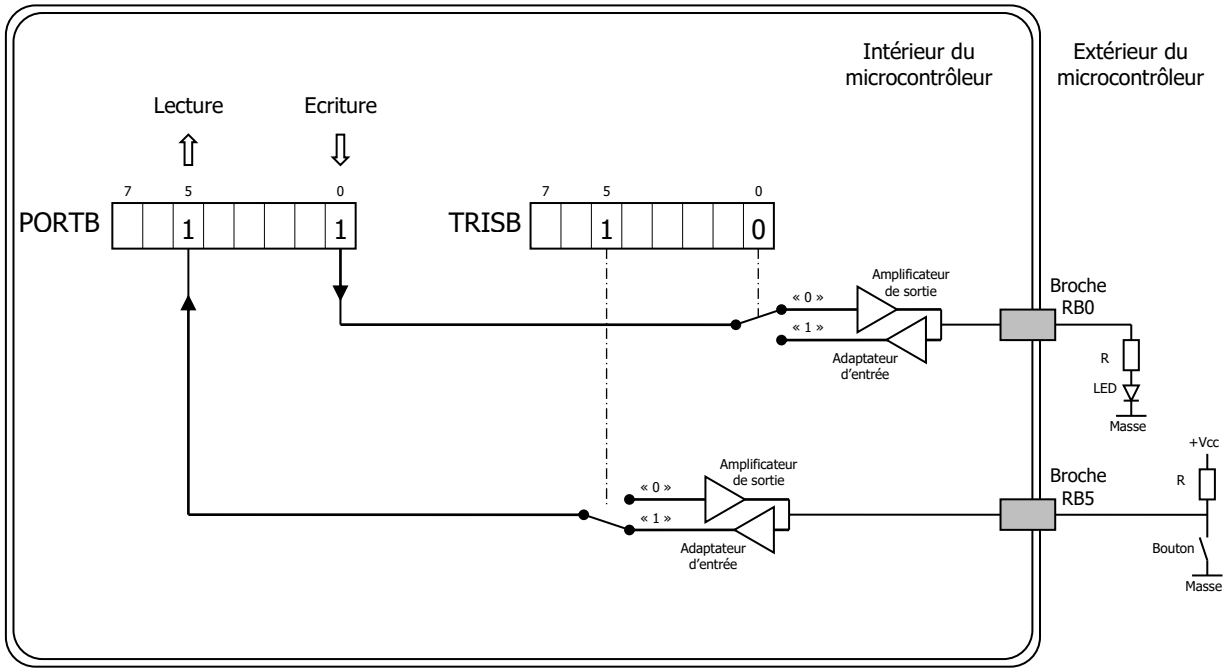
Q1. - Un microcontrôleur réalise-t-il la partie commande ou la partie opérative d'un système ?
- Les entrées d'un microcontrôleur servent-elles à recueillir des informations ou à transmettre des commandes ?
- Les sorties d'un microcontrôleur servent-elles à recueillir des informations ou à transmettre des commandes ?

Q2. Un bouton poussoir peut-il être raccordé sur une entrée ou sur une sortie d'un microcontrôleur ?
Une LED peut-elle être raccordée sur une entrée ou sur une sortie d'un microcontrôleur ?

Q3. Le registre TRISB du microcontrôleur permet-il de :
- Définir la direction des broches du port B (entrée ou sortie)
- Connaître l'état des broches du port B
- Forcer l'état des broches du port B

Le registre PORTB du microcontrôleur permet-il de :
- Définir la direction des broches du port B (entrée ou sortie)
- Connaître l'état des broches du port B
- Forcer l'état des broches du port B

La figure donnée à la page suivante montre une représentation simplifiée du fonctionnement d'un port d'entrée/sortie du microcontrôleur. On peut voir sur cette figure un exemple de broche utilisée en entrée et une autre en sortie.



Q4. Sur la figure ci-dessus, quelle broche est utilisée en entrée ? Justifier à l'aide de TRISB
Quelle broche est utilisée en sortie ? Justifier à l'aide de TRISB

Q5. A quoi sert l'instruction suivante : `TRISB = TRISB | 0x20 ; // (0b00100000)`
Quelle broche du microcontrôleur est concernée ? Quel sera son sens d'utilisation (entrée ou sortie) ? Expliquer comment fonctionne le masquage.

Q6. A quoi sert l'instruction suivante : `TRISB = TRISB & 0xFE ; // (0b11111110)`
Quelle broche du microcontrôleur est concernée ? Quel sera son sens d'utilisation (entrée ou sortie) ? Expliquer comment fonctionne le masquage.

Q7. A quoi sert l'instruction suivante : `PORTB = PORTB | 0x01 ; // (0b00000001)`
Dans ce cas, au niveau de la broche RB0, le potentiel est-il à +Vcc ou à la masse ? le courant sort-il du microcontrôleur ? rentre-t-il ? ou est-il nul ? En déduire l'état de la LED.

Q8. A quoi sert l'instruction suivante : `PORTB = PORTB & 0xFE ; // (0b11111110)`
Dans ce cas, au niveau de la broche RB0, le potentiel est-il à +Vcc ou à la masse ? le courant sort-il du microcontrôleur ? rentre-t-il ? ou est-il nul ? En déduire l'état de la LED.
Que se serait-il passé, si l'on avait utilisé l'instruction `PORTB = PORTB & 0xF0 ;`

En règle générale pour la programmation de microcontrôleur il est usuel d'utiliser le format hexadécimal. Si vous avez des difficultés avec ce format vérifiez vos valeurs avec le site web suivant : https://sebastienguillon.com/test/javascript/convertisseur.html

0b0000 ↔ 0x00	0b0100 ↔ 0x04	0b1000 ↔ 0x08	0b1100 ↔ 0x0C
0b0001 ↔ 0x01	0b0101 ↔ 0x05	0b1001 ↔ 0x09	0b1101 ↔ 0x0D
0b0010 ↔ 0x02	0b0110 ↔ 0x06	0b1010 ↔ 0x0A	0b1110 ↔ 0x0E
0b0011 ↔ 0x03	0b0111 ↔ 0x07	0b1011 ↔ 0x0B	0b1111 ↔ 0x0F

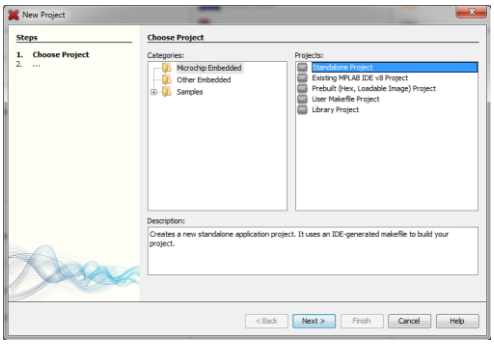
Fin de la partie préparation, mais il n'est pas interdit de lire la suite avant la séance de TP…

2. Expérimentation

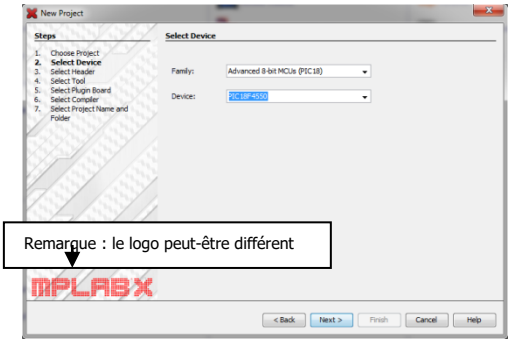
2.1 Création d'un premier projet

Pour éviter les aléas liés au réseau, il est conseillé de travailler en local sur le poste de travail. Sélectionnez le dossier ***C:\etudiant*** et créez un sous-dossier à votre nom (***toto*** pour l'exemple ci-dessous).

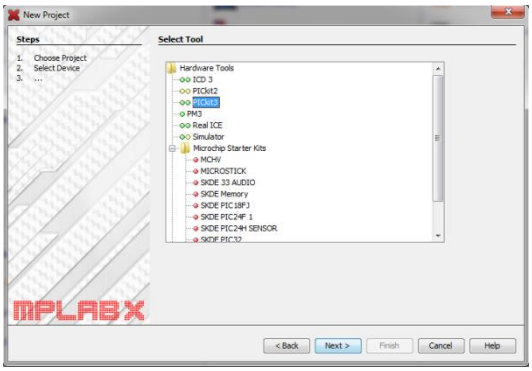
Exécutez MPLABX IDE v3.61 et créez un nouveau projet avec le menu ***File → New project***



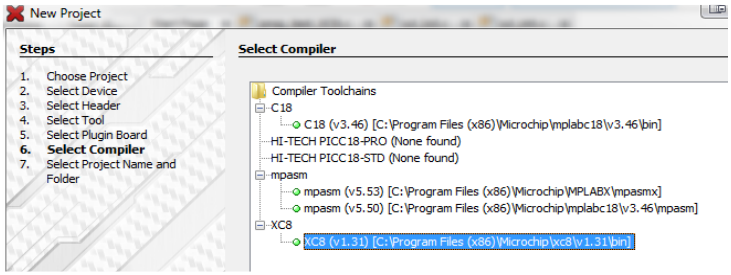
Dans la catégorie ***Microchip Embedded***, choisissez ***Standalone Project*** et cliquez sur ***Next***.



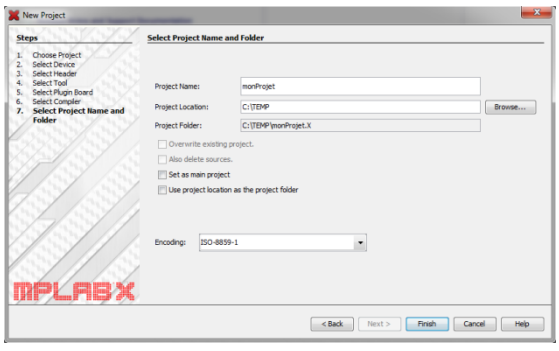
Choisissez la famille ***Advanced 8-bit MCUs (PIC18)*** et sélectionnez le microcontrôleur ***PIC18F4550***, puis cliquez sur ***Next***.



Sélectionnez le programmateur ***PicKit3*** et cliquez sur ***Next***.



Sélectionnez le compilateur ***XC8*** et cliquez sur ***Next***.



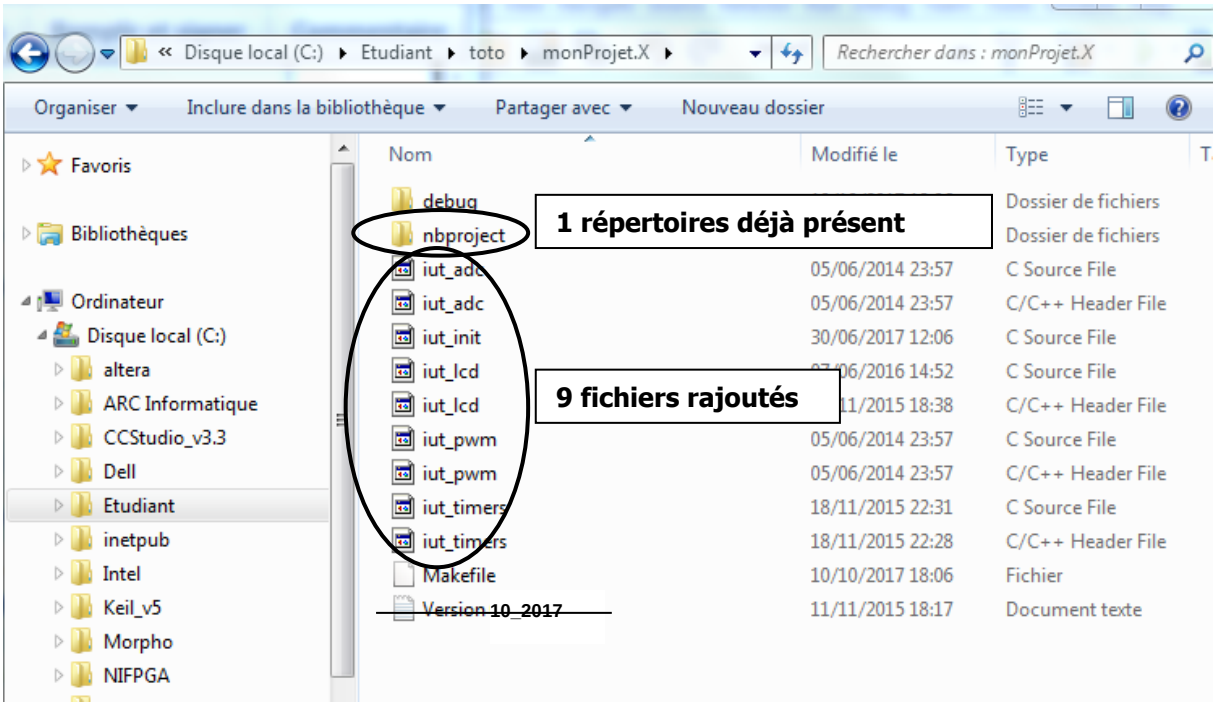
Choisissez un nom de projet (***Project Name***), par exemple ***monProjet***. Évitez les caractères spéciaux, les accents et les espaces.
Comme emplacement de sauvegarde de votre projet (***Project Location***), sélectionnez votre répertoire de travail (***C:\etudiant\toto*** pour l'exemple).

Cliquez sur *Finish*, la création du projet est terminée.

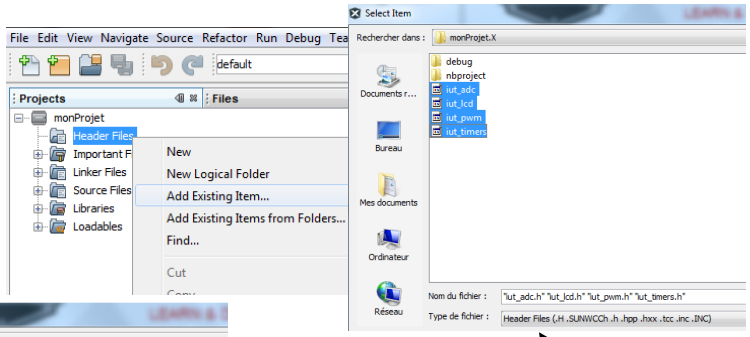
Ouvrir l'explorateur Windows, puis :
Copiez dans le répertoire du projet, ici ***C:\etudiant\toto\monProjet.X***, le fichier zip des bibliothèques de l'IUT pour la carte microcontrôleur, que vous trouverez dans le répertoire :

https://profge2.iut-cachan.u-psud.fr/foadIE/@II1S1/telechargements.html

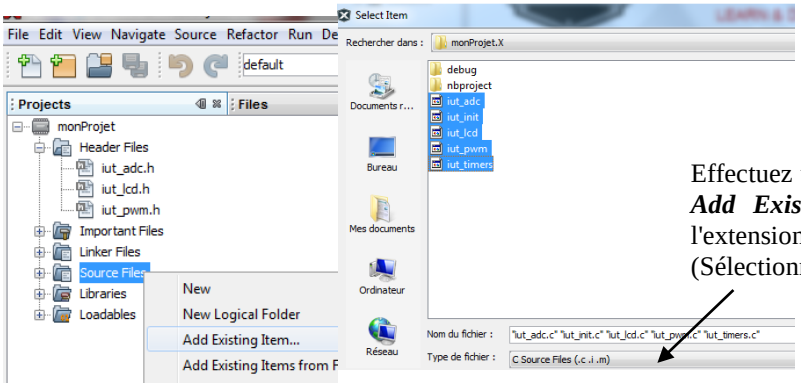
Décompressez le fichier zip et placer les fichiers obtenus dans votre répertoire de travail. Vous devez obtenir ceci :



Dans ***MplabX***, effectuez un clic droit sur ***Header Files*** puis cliquez sur ***Add Existing Item***.
Ajoutez les quatre fichiers avec l'extension ***.h***.

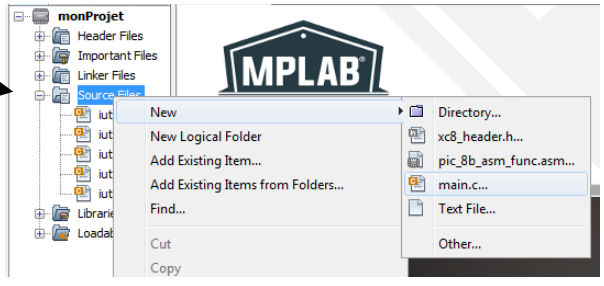


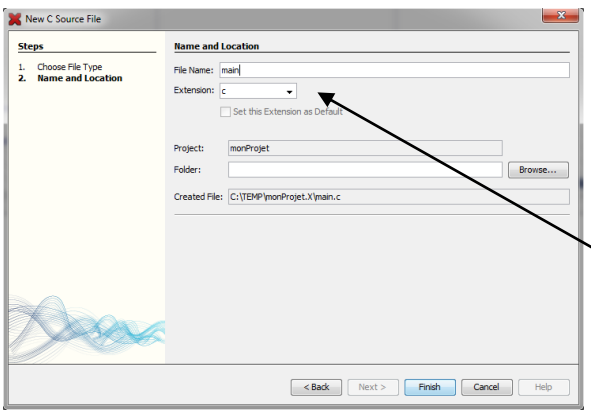
Choisir Header Files



Effectuez un clic droit sur ***Source Files*** puis cliquez sur ***Add Existing Item***. Ajoutez les cinq fichiers avec l'extension ***.c***.
(Sélectionner C Sources Files)

Effectuez un nouveau clic droit sur ***Source Files*** puis cliquez sur ***New → main.c…***





Choisissez un nom de fichier, par exemple ***main***.

Vous pouvez maintenant taper votre premier programme :

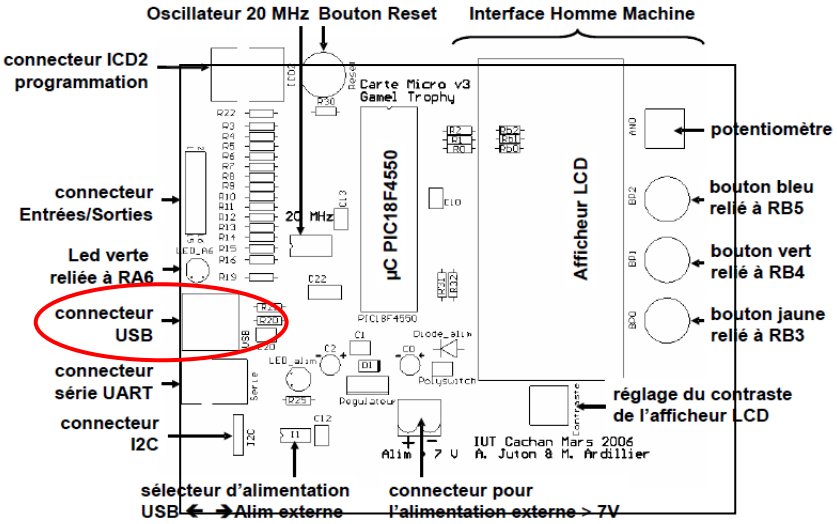
```
#include <xc.h>

void main(void)
     {
     while (1)
        {
        }
     }
```

Ce programme ne fait strictement rien. Il attend indéfiniment...

Mais après avoir cliqué sur ***Build Project*** 

Vous devez vérifier que vous avez créé correctement votre projet.

La carte que nous allons utiliser est constituée des éléments ci-contre :



Dans ce TP, nous alimenterons la carte microcontrôleur par le connecteur USB de l'ordinateur. Réaliser la connexion, et vérifier que la carte soit correctement alimentée. Si ce n'est pas le cas, changer la position du sélecteur d'alimentation.

Pour les premiers essais, nous allons faire afficher un texte, sur l'écran LCD, avec le programme suivant :

```
#include <xc.h>
#include "iut_lcd.h"        // Entêtes des fonctions gérant l'afficheur

void main(void)
     {
     lcd_init ();           // Appel de la fonction initialisant l'afficheur
     lcd_position (0, 2);   // Appel de la fonction de positionnement du curseur
     lcd_printf ("TP II1"); // Appel de la fonction d'affichage

     while (1)   // Une fois l'affichage terminé le programme boucle sans fin
        {}
     }
```

- Saisir le programme et compiler. Brancher le programmateur ***PicKit3*** à la carte microcontrôleur.

- Cliquer sur ***Make and Program Device*** 

Si tout va bien, le message doit s'afficher sur l'écran LCD, sinon appelez à l'aide !
Débranchez le programmateur, éteignez et rallumez la carte microcontrôleur. Remarquez que le fonctionnement est autonome et ne dépend pas du programmateur ou du PC.

Q9. L'écran LCD est un afficheur de caractères à 2 lignes et 16 colonnes.
La fonction `lcd_position` permet le positionnement du curseur à l'écran.
Modifier la position de l'affichage et expliquer comment fonctionne le système de coordonnées.

2.2 Lecture des entrées TOR

Outre le bouton rouge de Reset, la carte dispose de 3 boutons poussoirs, connectés sur le port B :

- Bouton poussoir jaune BP0 sur RB3
- Bouton poussoir vert BP1 sur RB4
- Bouton poussoir bleu BP2 sur RB5

Commençons par lire le bouton BP0, avec le programme suivant :

```
#include <xc.h>
#include "iut_lcd.h"

void main(void)
       {
       char bp0;

       lcd_init();

       TRISB = TRISB | 0x38; //(0b00111000) uniquement les Pins RB5, RB4, RB3 du port B
                             // définies en entrée grâce au masquage

       while (1)
              {
              bp0 = PORTB & 0x08; //(0b00001000) lecture et masquage de l'entrée RB3
              lcd_position(0, 0); // place le curseur en haut à gauche de l'écran LCD
              lcd_printf("BP0: %08B %02X", (int)bp0, (int)bp0);
              } // affiche la valeur de BP0 en binaire et en hexa
       }
```

Remarque pour le printf : Le format "%08B" permet d'afficher un **int** en binaire sur 8 caractères en écrivant les 0 à gauche non significatifs. L'expression **(int)** qui suit transforme la valeur de l'octet qui est un **char** en **int** pour obtenir un affichage correct.

Q10. Saisir et exécuter le programme sur la carte microcontrôleur. Justifier la valeur de BP0 affichée, en fonction de l'état appuyé/relâché.
Le câblage des boutons est-il réalisé en « pull-up » ou « pull-down » ? Expliquer en dessinant un schéma électrique avec les 2 positions du bouton poussoir.

Q11. Modifier le programme pour afficher l'état du bouton BP1, justifier la valeur du masquage utilisé.

Le programme suivant permet de lire l'état des 3 boutons en même temps :

```
#include <xc.h>
#include "iut_lcd.h"

void main(void)
    {
    char bp0,bp1,bp2,bpTous ;

    lcd_init();

    TRISB = TRISB | 0x38; //(0b00111000) uniquement les Pins RB5, RB4, RB3 du port B

    while (1)
         {
         bpTous = PORTB & 0x38 ; //(0b00111000) lecture du port B avec masque

         bp0 = bpTous & 0x08; //(0b00001000) nouveau masque pour garder uniquement le bit 4
         bp1 = bpTous & 0x10; //(0b00010000) nouveau masque pour garder uniquement le bit 5
         // bp2 = bpTous & ????;

         lcd_position(0, 0);
         lcd_printf("BP0: %02X BP1: %02X", (int)bp0, (int)bp1);
         }
    }
```

Q12. Ajouter les instructions pour lire et afficher BP2. Justifier la valeur du masque utilisé.
Tester le programme et **faire valider par le professeur.**

Q13. Quelles sont les valeurs possibles en hexa et en binaire pour `bpTous`, quand les boutons sont appuyés et relâchés. Compléter le document réponse DR1.

La carte microcontrôleur dispose d'un connecteur qui peut être utilisé pour raccorder des entrées et sorties supplémentaires, sur une carte d'extension (à demander au professeur).

Q14. Dessiner sur le document DR1 un interrupteur en mode pull-up (en utilisant une résistance), l'interrupteur, noté BPA, est à brancher sur la pin RB2 du microcontrôleur.
Ajouter au programme précédent la lecture et l'affichage de l'état de l'interrupteur sur l'écran.
Expliquer les instructions et les masquages utilisés.
Câbler l'interrupteur et tester le fonctionnement. **Faire valider par le professeur**.

Q15. Câbler maintenant l'interrupteur en mode pull-down (en utilisant une résistance), toujours sur la pin RB2. Sans changer le programme, tester le fonctionnement. Justifier les valeurs affichées sur l'écran, selon l'état de l'interrupteur.
Imprimer le programme complet (à joindre au compte-rendu).

2.3 Ecriture d'une sortie TOR et réalisation de fonctions logiques combinatoires

Une led verte est connectée sur la broche 6 du port A.
- Si RA6=1 alors la sortie est à 5V. La led sera allumée
- Si RA6=0 alors la sortie sera à 0V. La led sera éteinte

Nous allons allumer cette led si l'on appuie sur BP0 (et l'éteindre sinon), en utilisant le programme donné ci-dessous.

Q16. Expliquer ce que réalise chacune des instructions dans la boucle while(1). Justifier notamment dans quel cas la led est allumée et dans quel cas elle est éteinte.

```
#include <xc.h>
#include "iut_lcd.h"

void main(void)
     {
     char bp0;
     TRISB = TRISB & 0x38;  // (0b00111000) Broches RB5, RB4, RB3 définies en entrée
     TRISA = TRISA & 0xBF;  // (0b10111111) La broche RA6 est définie en sortie
     lcd_init ();
     while (1)
        {
        bp0 = PORTB & 0x08; // (0b00001000)
        lcd_position(0, 5); lcd_printf("BP0 : %02X", (int) bp0);
        if (bp0 == 0x00)
               {
               PORTA = PORTA | 0x40;      // (0b01000000)
               }
        else {
               PORTA = PORTA & 0xBF;      // (0b10111111)
               }
        }
     }
```

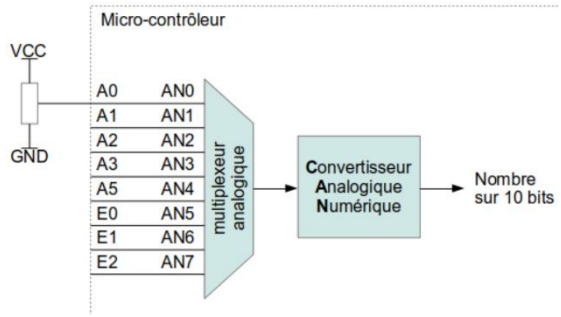
Q17. Saisir et exécuter le programme.
Proposer 2 solutions de modification du programme afin d'obtenir le fonctionnement inverse, c'est-à-dire la led allumée au repos et éteinte si bouton appuyé.

Q18. Sur la carte d'extension relier la led D1 sur le port RB0. Dessiner le branchement sur DR1.
Compléter le programme, pour que BP1 allume D1 (si appuyé et led éteinte sinon).
Faire valider par le professeur.

Q19. Sur la carte d'extension relier la led D5 sur le port RB1. Dessiner le branchement sur DR1.
Compléter le programme, pour que BP2 allume D5 (si appuyé et led éteinte sinon).
Faire valider par le professeur et imprimer le programme complet (à joindre au compte-rendu).

2.4 Entrées analogiques

Le PIC18F4550 est muni en interne d'un convertisseur analogique numérique 10 bits. Par multiplexage, celui-ci peut acquérir jusqu'à 8 tensions sur les canaux nommés AN0 à AN7.



Le canal ANalogique 0 (AN0) est relié, en interne, à la broche RA0 du microcontrôleur (noté A0 sur le schéma ci-dessus). Tous les autres canaux, sont également reliés en interne à des broches (le dernier AN7 est lié à la broche RE2, cf schéma ci-dessus).
En externe, sur la carte, à côté des boutons poussoirs, un potentiomètre est branché sur la broche RA0. Il est câblé comme indiqué sur le schéma ci-dessus. L'entrée du microcontrôleur n'absorbant pas de courant, le montage est un simple diviseur de tension. Ainsi, la tension $V_{AN0} = \alpha \times 5V$ avec α la position du curseur comprise entre 0 et 1.
Pour programmer le convertisseur simplement, une bibliothèque regroupe les fonctions nécessaires. Pour utiliser cette bibliothèque, il faut inclure le fichier d'en-tête ***iut_adc.h*** avec l'instruction :

```
#include "iut_adc.h"
```

Prototypes des fonctions disponibles :

`void adc_init(char numero_dernier_canal);`

Cette fonction initialise le convertisseur analogique numérique (nombre de canaux à utiliser, temps d'acquisition, temps de conversion…).
On peut utiliser le nombre d'entrées analogiques que l'on veut, à condition de les prendre successives à partir de AN0, et de donner en paramètre de la fonction, le numéro du dernier canal utilisé. Exemple : une valeur de 2, stipule que l'on veut utiliser AN0, AN1 et AN2.

`int adc_read(char numero_canal);`

Cette fonction déclenche la conversion analogique numérique sur le canal indiqué en paramètre (0 pour AN0, 1, pour AN1…). Elle renvoie ensuite le résultat de conversion sous forme d'un entier sur 10 bits. Le temps de conversion est d'environ 25µs.

Exemple de programme, effectuant la lecture de la valeur du potentiomètre (AN0 relié à RA0) et la stockant dans un entier :

```
#include <xc.h>
#include "iut_adc.h"

void main(void)
       {
       int ValeurADC = 0;

       adc_init(0); // initialisation du convertisseur, canal AN0

       while(1)
           {
           ValeurADC = adc_read(0); // lecture de AN0
           lcd_position(0, 0);
           lcd_printf("ADC= %b", ValeurADC);
           lcd_position(1, 0);
           lcd_printf("ADC= %4d", ValeurADC);

           }
       }
```

Q20. Tester le fonctionnement, et justifier la valeur maximum obtenue (on rappelle que l'on dispose d'un convertisseur 10 bits).

Nous allons maintenant appliquer une tension externe, issue d'un GBF, sur le canal AN1 du convertisseur. Pour éviter de détériorer le composant, régler avant câblage, le GBF pour qu'il délivre une tension continue de 1 Volt.
Brancher ensuite la sortie du GBF, à la broche RA1 de la carte d'extension (et ne pas oublier de relier les masses).
Ajouter au programme précédent, les instructions nécessaires pour faire afficher sur l'écran, en décimal, la valeur issue de la conversion sur AN1 (Ne pas oublier de modifier `adc_init`).

Q21. Tracer la courbe de la valeur numérique de AN1 en fonction de la tension du GBF. Prendre une valeur tous les 0,5V, de 0 jusqu'à la valeur max de 5V (à ne pas dépasser).

Q22. Modifier le programme pour faire afficher la valeur des entrées analogiques en volts sur l'afficheur. La valeur sera un `float`, affiché avec 3 chiffres après la virgule, en utilisant le format %5.3f pour le printf.
Faire valider par le professeur et imprimer le programme complet (à joindre au compte-rendu).

2.5 Le simulateur

Pour pouvoir refaire ce TP sans disposer de la carte microcontrôleur, un simulateur est disponible sur le site d'IUT en ligne : (et il est aussi installé sur les PC de l'IUT sous le nom de PicSimGamel)

https://profge2.iut-cachan.u-psud.fr/foadIE/@II1S1/telechargements.html

Quand vous compilez votre projet avec MPLAB, un fichier avec l'extension hex est généré dans le répertoire :

C:\etudiants\..... \monProjet.X\dist\default\production

Il suffit de charger ce fichier dans le simulateur pour exécuter votre programme.
Vérifiez que vous arrivez à faire fonctionner vos programmes sur le simulateur.

Document Réponse DR1 **Noms :**.....

Question Q13

Etat des boutons			Valeur de bpTous	
BP2 (RB5)	BP1 (RB4)	BP0 (RB3)	Binaire	Hexadécimal
Relâché	Relâché	Relâché		
Relâché	Relâché	Appuyé		
Relâché	Appuyé	Relâché		
Relâché	Appuyé	Appuyé		
Appuyé	Relâché	Relâché		
Appuyé	Relâché	Appuyé		
Appuyé	Appuyé	Relâché		
Appuyé	Appuyé	Appuyé		

Question Q14, Q18 et Q19