

Nom, prénom : _____ Groupe : _____

TP 6 : Opérations sur les bits

Préparation

Durant ce TP, nous allons utiliser une maquette d'entrées / sorties sur le bus USB. Cette maquette contient 8 leds et 8 interrupteurs. Pour utiliser la maquette, la librairie inoutUSB.h vous est fournie. Vous devrez utiliser les 4 fonctions suivantes :

- unsigned char OpenPort(void) : cette fonction « ouvre » le canal de communication avec la maquette. Si tout se passe bien (boîtier présent, absence de problèmes de communication ...) la fonction renvoie 1 (valeur logique vraie) sinon la fonction renvoie 0 (Valeur logique faux).
- void ClosePort(void) : cette fonction « ferme » le canal de communication. Lorsque la maquette n'est plus utilisée par le programme (à la fin par exemple), il est nécessaire d'appeler cette fonction.
- unsigned char InPortUSB(void) : cette fonction renvoie l'état des huit interrupteurs.
- void OutPortUSB (unsigned char) : cette fonction allume ou éteint les leds conformément à la valeur passée en paramètre.

Noter que l'appel des fonctions OpenPort et ClosePort ne se fait généralement qu'une seule fois (respectivement au début et à la fin du programme).

Dessiner l'organigramme du programme suivant :

```
unsigned char Val=0;

if (!OpenPort())      // Ouverture du port d'entrées/sorties
{
    // Erreur lors de l'ouverture
    printf ("Erreur 1 : Il y a un problème avec la maquette");
    exit (1);          // Arrêt du programme
}

while (!kbhit())      // Jusqu'à ce qu'une touche soit appuyée
{
    Val=InPortUSB();   // Val <- Etat des interrupteurs
    OutPortUSB (Val);  // Recopie l'état des interrupteurs (Val) sur les led
}

OutPortUSB (0);       // Eteint toutes les leds
ClosePort();          // Ferme le port
```

Organigramme :

Travail pratique

Exercice 1 : Créer un nouveau projet. Installer le pilote du boîtier d'entrées/sorties :

Au moyen du menu **Projet/Ajouter au Projet** rechercher le fichier de librairie **DriverUSB.obj** qui se trouve dans le répertoire suivant :

C:\ProgramFiles\Borland\Builder4\Imports

Saisir et tester le programme de la préparation en conservant les lignes insérées par l'environnement.

Effectuer d'éventuelles modifications dans le programme afin de déterminer le poids (1,2,4,8,16, ...) de chaque interrupteur et de chaque led. Compléter directement sur le schéma du boîtier.

Effectuer d'éventuelles modifications dans le programme afin de déterminer si les leds sont actives à l'état haut ou à l'état bas. De même, déterminer l'état renvoyé par un interrupteur lorsqu'il est basculé vers le haut et vers le bas.

Exercice 2 : On souhaite dorénavant que les leds soient allumées lorsque les interrupteurs sont basculés vers le bas et éteintes sinon. Modifier le programme.

Exercice 3 : On souhaite que les 3 leds de droite soient toujours allumées et que les 5 autres soient commandées par les interrupteurs comme à l'exercice 1.

Exercice 4 : On souhaite que les 3 leds de gauche et la led de droite soient toujours éteintes et que les 4 autres soient commandées par les interrupteurs comme à l'exercice 2.

Exercice 5 : On souhaite que :

- les deux leds de gauche soient commandées comme à l'exercice 1
- les deux suivantes soient toujours allumées
- les deux suivantes soient commandées comme à l'exercice 2
- les deux dernière soient toujours éteintes.

Exercice 6 : Saisir le programme suivant :

```
void main(void)
{
    unsigned char Val=0x80;    // 0x80 = Première led allumée
    char ch;                  // Caractère lu au clavier
    if (!OpenPort())          // Ouverture du port d'entrées/sorties
    {
        // Erreur lors de l'ouverture
        printf ("Erreur 1 : Il y a un problème avec la maquette");
        return;              // Arrêt du programme
    }
    // Petit message à l'utilisateur
    printf ("Appuyer sur une touche pour décaler, Echap pour quitter");

    do
    {
        OutPortUSB (Val);    // Ecriture sur les led de Val
        ch=getch();          // Attend un appui sur une touche
        Val=Val>>1;          // Val est décalé à droite de 1 bit

    }
    while (ch!=27);           // Jusqu'à un appui sur Echap

    OutPortUSB (0);          // Eteint toutes les leds
    ClosePort();             // Fermeture du canal d'entrées/sorties
}
```

Au cours d'un décalage à droite, le bit de poids fort est-il remplacé par un 1 ou un 0 ? Remplacer `unsigned char Val=0x80;` par `char Val=0x80;` . Compléter la première ligne du tableau sur le document réponse.

Modifier le programme de façon à ce que l'interrupteur de gauche commande le sens de décalage (droite ou gauche). Initialiser Val à 0x08.

Compléter le reste du tableau et expliquer ce phénomène.

Exercice 7 :

Programmer un chenillard qui change de sens selon l'état du 5^{ème} interrupteur.

Exercice supplémentaire :

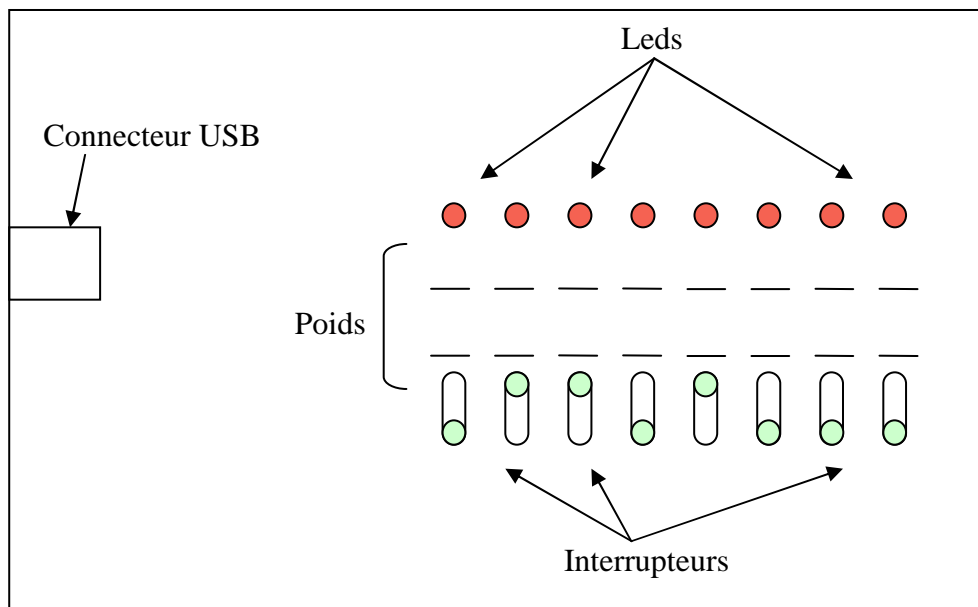
Ecrire un programme où :

- la led de gauche est commandée par l'interrupteur de droite.
- la seconde led est commandée par l'avant-dernier interrupteur.
- la 3^{ème} led est commandée ...
- ...
- la led de droite est commandée par l'interrupteur de gauche.

Il n'existe pas de fonction ou d'instruction permettant de le faire directement, utiliser une boucle.

Compte-rendu à compléter

Exercice 1 :



● Etat logique : _____ ○ Etat logique : _____

○ Etat logique : _____ ● Etat logique : _____

Exercice 2 : A compléter :

```
while (!kbhit())      //Jusqu'à ce qu'une touche soit appuyée
{

}

}
```

Exercice 3 : A compléter :

```
while (!kbhit())      //Jusqu'à ce qu'une touche soit appuyée
{

}

}
```

Exercice 4 : A compléter :

```
while (!kbhit())      //Jusqu'à ce qu'une touche soit appuyée
{

}

}
```

Exercice 5 : A compléter :

```
while (!kbhit())      //Jusqu'à ce qu'une touche soit appuyée
{

}

}
```

Exercice 6 :

Décalage à droite

	unsigned char	char
Bit de poids fort à 1 avant le décalage	Le bit de poids fort est remplacé par un : -----	Le bit de poids fort est remplacé par un : -----
Bit de poids fort à 0 avant le décalage	Le bit de poids fort est remplacé par un : -----	Le bit de poids fort est remplacé par un : -----

Décalage à gauche

	unsigned char	char
Bit de poids faible à 1 avant le décalage	Le bit de poids faible est remplacé par un : -----	Le bit de poids faible est remplacé par un : -----
Bit de poids faible à 0 avant le décalage	Le bit de poids faible est remplacé par un : -----	Le bit de poids faible est remplacé par un : -----

Explication :

Exercice 7 et exercice supplémentaire : A la fin de chaque exercice ; commenter, indenter le programme et appeler l'enseignant afin qu'il valide votre code source.

Tableau à compléter par l'enseignant :

Exercice 7 :	Exercice supplémentaire :