Q1 : Charger sous Proteus le fichier tp2-1.dsn et utiliser sous MPLAB le corps du programme suivant :

```
/*****************************************/
#include "iq.h"
#define byte unsigned char

byte *trisb = 0xf93 ;
byte *portb = 0xf81 ;
byte *trisd = 0xf95 ;
byte *portd = 0xf83 ;

/* a compléter */

void affiche_7seg(byte val)
{
byte tab_7seg[]={0b00111111,0b00000110,0b01011011,0b01001111,0b01100110,0b01101101,
                 0b01111101,0b00000111,0b01111111,0b01101111};
*portd=tab_7seg[val];
}

int test_plus()
{
/* A compléter */
}


void main(void)
{
int i=0;
*trisd = 0 ;
*trisb = 0xff;
affiche_7seg(i);
//for(i=0;i<4;i++)
while(1){
        while (test_plus());
        while (! test_plus()); ;

        if (i < 9) i++;
                else i=0;

        affiche_7seg(i);
}

while(1) ;
}

/*********************************************/
```

Compléter la fonction *test_plus()* pour qu'elle retourne 1 lorsque le bouton plus est appuyé et 0 lorsqu'il est relâché. Tester le fonctionnement.

Q2 : Ajouter une fonction *test_moins()* et tester. L'utilisation de test_plus() et de test_moins() simultanément est-il aisé ? Pourquoi ?

Pour résoudre le problème, nous allons utiliser un programme sous interruption.

Q3 : Utiliser sous MPLAB le corps du programme suivant :

```
/*********************************************/

#include "iq.h"
#include "it.h"
#define byte unsigned char

/* variables globale */
byte *trisb = 0xf93 ;
byte *portb = 0xf81 ;
byte *trisd = 0xf95 ;
byte *portd = 0xf83 ;
int i=0;
int nb=0;
void affiche_7seg(byte);
/* interruption sur front montant RB0 */
void it_int0()
{
}

/* interruption sur front montant RB1 */
void it_int1()
{
}

void it_tmr0()
{
}

void affiche_7seg(byte val)
{
byte
tab_7seg[]={0b00111111,0b00000110,0b01011011,0b01001111,0b01100110,0b01101101,0b011111
01,0b00000111,0b01111111,0b01101111};
*portd=tab_7seg[val];
}
```

```
void main(void)
{

*trisd = 0 ;
*trisb = 0xff;
init_it_int0();
init_it_int1();
affiche_7seg(i);

while(1){

        }


}

/***********************************************/
```

Changer de simulateur. Utiliser le debugger PROTEUS VSM. On peut debbuger en interaction avec le matériel.
Mettre un point d'arrêt sur les fonctions it_int0 et it_int1 . Placer les point d'arrêt sur les accolade fermantes ({).
Lancer le programme et expliquer le fonctionnement en appuyant sur les BP plus et moins.

Q4 : Modifier it_int0 et it_int1 pour faire incrémenter et décrémenter l'afficheur. Quels sont les avantages par rapport au fonctionnement en Q1 et Q2 ?

Q5 : Une interruption peut être programmée pour être déclenchée régulièrement par un timer interne au micro-contrôleur. Ajouter la ligne « init_it_tmr0(); » au début du main. Ajouter un point d'arrêt dans la fonction it_tmr0. Lancer et observer. Conclure.

Q6 : Modifier la fonction it_tmr0 pour effectuer un compteur qui évolue toute les secondes.

Q7 : Pour cette question, on utilisera le schéma TP2-2. Le but est de réaliser un chronomètre. Tester et comprendre le fonctionnement du programme suivant :

```
#include "iq.h"
#include "it.h"
#define byte unsigned char

/* variables globale */
byte *trisb = 0xf93 ;
byte *portb = 0xf81 ;
byte *trisd = 0xf95 ;
byte *portd = 0xf83 ;
byte *trisc = 0xf94;
```

```
byte *portc = 0xf82;

int i=0;
int nb=0;
int seconde = 0;
void affiche_7seg(byte);

void affiche_7seg(byte val)
{
byte
tab_7seg[]={0b00111111,0b00000110,0b01011011,0b01001111,0b01100110,0b01101101,0b011111
01,0b00000111,0b01111111,0b01101111};
*portb=tab_7seg[val];

}

void it_int0()
{
}

void it_int1()
{
}

/* interruption timer */
void it_tmr0()
{
i++;
affiche_7seg(seconde % 10);
if ((i % 10) == 0) seconde ++;

}

void main(void)
{
*trisc = 0xff;
*trisd = 0 ;
*trisb = 0;
*portb = 0;
*portd = 0xfe;
init_it_tmr0();


while(1){

        }

}

/*********************************************/
```

Tester le fonctionnement en modifiant la ligne *portd = 0xfe en *portd = 0xfd ;
Conclure sur le fonctionnement de l'afficheur.

Q8 : Modifier la fonction *affiche_7seg(byte val)* en

affiche_7seg(byte val, byte num_affiche) pour pouvoir afficher une valeur sur un afficheur particulier. (une tempo de 1ms est nécessaire pour la prise en compte de l'affichage).

Q9 : réaliser le chronomètre. (le test des BP se fait dans la boucle du programme principal).