Université Mohammed Premier
Ecole Nationale des sciences appliquées
- Oujda –

Génie des Systèmes Electronique
Informatique et Réseaux
GSEIR

TP 4 – Informatique embarquée

Gestion des TIMERS et 7seg display

Encadré par :

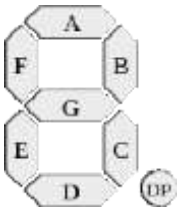
EMHARRAF Mohamed

1. Objectifs

Manipuler les 4 afficheurs 7 segments de la carte EasyPic7 sous contrôle de Timer. Dans un premier temps, vous mettrez en œuvre un seul afficheur afin de maitriser uniquement la gestion des segments et l'utilisation d'un fichier de définition (.h). Ensuite, vous devrez multiplexer deux afficheurs selon 2 approches différentes, sans et avec gestion de timer.

2. Les afficheurs 7 segments de la carte Easypic7

Un afficheur est constitué de 7 segments identifiés par les lettres de A à G et du point décimal DP. Chaque segment est rattaché à l'une des broches du µC. Pour les activer, vous écrirez dans le registre LATx correspondant au port x, une valeur hexadécimale (mais on pourrait également donner une valeur binaire ou octale). En vous aidant du schéma de l'afficheur, complétez le tableau suivant :

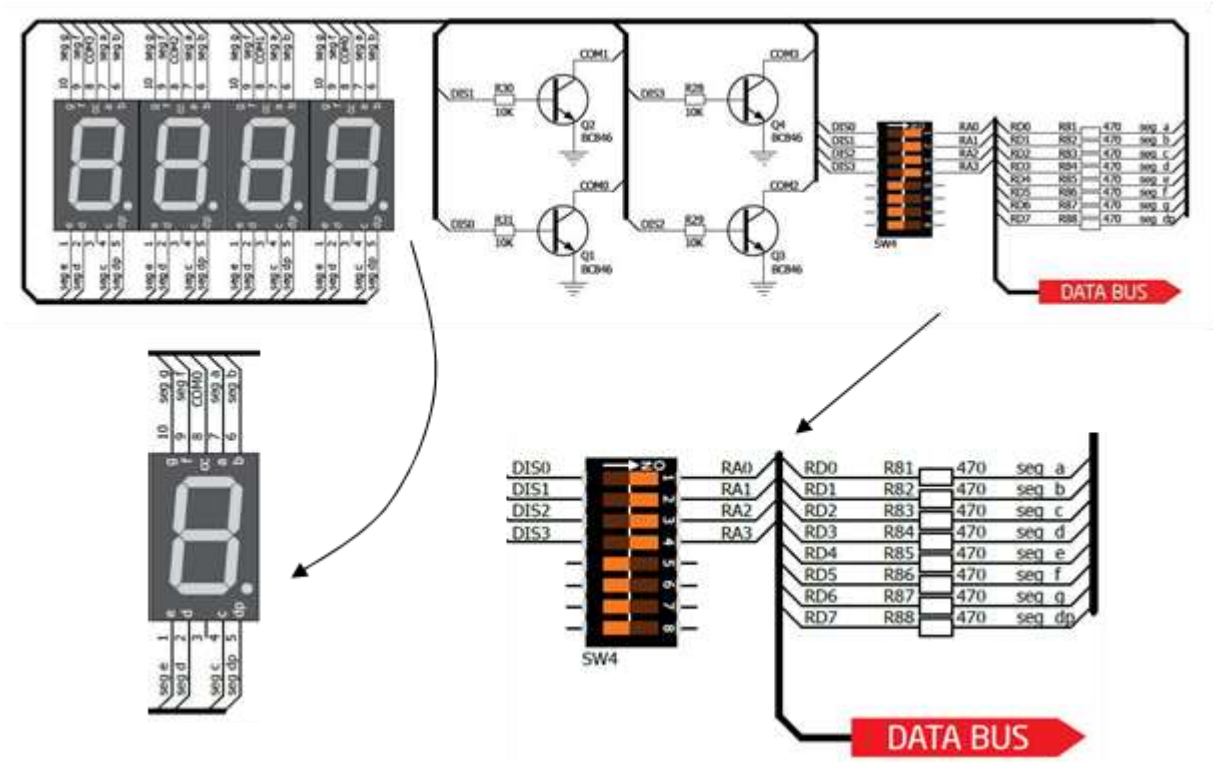


Affichage	bit7 DP	bit 6	bit 5	bit 4	bit 3	bit 2	bit1 B	bit 0	Hexa
0	0	0	1	1	1	1	1	1	**0x3F**
1	0								
2	0								
3	0								
4	0								
5	0								
6	0								
7	0								
8	0								
9	**0**								

Ce tableau représente une opération de masquage qui sera mise en œuvre dans un fichier de définition (.h). Autrement dit, chaque symbole (0, 1, 2 etc.) est remplacé par un mot binaire. Par exemple, le symbole 8 (0000 1000b) est remplacé par le mot binaire 0111 1111b. Mais on pourrait s'y prendre autrement et utiliser un tableau de 10 cases pour mémoriser les valeurs hexadécimales.

Vous pouvez facilement contrôler les valeurs hexadécimales obtenues grâce à l'un des utilitaires de mikroC. Pour l'afficher : Tools > Seven Segment Editor.

Le schéma des connections des afficheurs est rappelé ci-dessous :



3. Exercice 1 : Gestion de l'afficheur DIS0 seul

Votre projet sera ici constitué de 2 fichiers, un fichier principal (.c) et un fichier de définition opérant le masquage (.h). Afin que le compilateur puisse lier le fichier .h au fichier .c, la directive #include sera utilisée.

A. Grâce au tableau que vous avez rempli en page 2, compléter le fichier suivant (fonction mask):

```
// Fichier Display_utils.h
unsigned short mask(unsigned short num) {
  switch (num) {
  case 0 : return 0x3F;
  case 1 : return.....;
  case 2 : return ... ;
  case 3 : return ... ;
  case 4 : return ... ;
  case 5 : return ... ;
  case 6 : return ... ;
```

```
    case 7 : return ... ;
    case 8 : return ... ;
    case 9 : return ... ;
    }
  }
```

Remarques :

- Utilisation du format unsigned short, sur 8 bits (de 0 à 255), ce qui est largement suffisant ici
- il est également possible de n'indiquer que le prototype de la fonction dans le fichier .h (càd. seulement la ligne " unsigned short mask(unsigned short num); ") et de coder la fonction dans un autre fichier .c.

B. Remplacer les lettres **x** et **y** par les bons ports dans fichier principal suivant :

```
// Fichier principal (DIS0.c)
#include "Display_utils.h"
void main () {
  unsigned short i;
  RCON|= 0x80; // Disable priority levels on interrupts
  INTCON = 0; // Disable all interrupts
  TRISx = 0; // Configure Portx as output
  LATx = 0; // Turn off all 7seg displays
  TRISy = 0; // Configure Porty as output
  LATy =0;  // Clear port y
  while (1) {
     for (i = 0 ; i <= 9 ; i++) {
      LATx = 0; // Turn off all 7seg displays
      LATy= mask(i); // bring appropriate value to PORTy
      LATx = 1; // turn on appropriate 7seg. display
      Delay_ms (1000);
    }
  }
}
```

Avant d'aller plus loin, analyser le code. Selon vous, que va faire l'afficheur DIS0?

1- Configurer le switch SW4 pour permettre l'activation de l'afficheur DIS0 (SW4.1 on).
2- Créer un nouveau projet (TP4/DIS0), fréquence 8 MHz
3- Recopier le contenu du fichier DIS0.c et enregistrer
4- Ouvrir une page blanche : File > New > New Unit

5- Recopiez le contenu du fichier Display_utils.h
6- Sauvegarder ce fichier : File > Save As puis sélectionner le type Header File (*.h)
7- Allumer la carte EasyPic7
8- Compiler et programmer (built+Program) le µC
9- Vérifier le bon fonctionnement de votre code

Modifier le programme afin d'utiliser l'afficheur DIS1, puis DIS2 et finalement DIS3.

4. Exercice 2 : Gestion de 2 afficheurs - Multiplexage

Vous allez maintenant mettre en œuvre 2 afficheurs (DIS0 et DIS1) afin d'afficher une valeur qui s'incrémentera de 00 à 50. Etant donné qu'il n'est pas possible de valider simultanément plusieurs afficheurs (en raison du câblage de la carte EasyPic7), il vous faudra basculer de l'un à l'autre (multiplexage). Si le multiplexage fonctionne bien, le basculement ne doit pas être visible. Dans cet exercice, il sera indispensable de basculer entre les 2 afficheurs en passant par l'algorithme suivant :

Faire plusieurs fois (boucle for : for (j = 1; j <= 50; j++) {)
- activer DIS0
- afficher le chiffre de poids faible
- tempo (10 ms par exemple)
- activer DIS1
- afficher le chiffre de poids fort
- tempo (10 ms par exemple)

1. Autoriser l'utilisation des afficheurs DIS0 et DIS1 (switch SW4.1 et .2 on)
2. Créer un répertoire TP4/DISO1/
3. Créer un nouveau projet (TP4/DIS01), fréquence 8 MHz
4. Dans la seconde fenêtre du New Project Wizard, ajouter le fichier Display_utils.h
5. En s'appuyant sur l'algorithme donné, écrire et tester le programme qui affiche un compteur de 00 à 50 sur les deux afficheurs. Vous utiliserez les opérateurs / (division) et % (modulo – reste de la division) :
   a. Pour extraire le chiffre des dizaines d'un nombre « number » : digit = (number / 10) % 10.
   b. Pour extraire le chiffre des unités : digit = number % 10;
6. Modifier la valeur des temporisations (100 ms)

7. Que constatez-vous ? Donnez une explication.
8. Même si la valeur de la temporisation est faible (10 ms ou moins), cette manière de programmer est-elle viable ?

5. Exercice 3 : Gestion des 4 afficheurs - Interruption

Vous allez maintenant utiliser une autre méthode pour réaliser le multiplexage. Il s'agit ici d'utiliser l'interruption généré par le Timer0. Le programme incomplète suivant (DIS0123.c), permet d'afficher un compteur sur les 4 afficheurs 7 segments de la carte Easypic7.

```
#include "Display_utils.h"

unsigned short shifter;            // For offset display
unsigned short portd_index;     // data table index
unsigned int   digit, number;      // data digits
unsigned short portd_array[4]; // data table

void interrupt() {
  LATA = 0;                              // Turn off all 7seg displays
  LATD = portd_array[portd_index]; // bring appropriate value to
  PORTD LATA = shifter;                // turn on appropriate 7seg display
  // move shifter to next digit
  shifter <<= 1;
  if (shifter > 8)
    shifter = 1;
  // increment portd_index
  portd_index++ ;
  if (portd_index > 3)
    portd_index = 0;
    // reset TIMER0 data register value
    // Clear TMR0IF
}
void main()
        {

  // Configure PORTA as output
  // Clear PORTA
  // Configure PORTD as output
  // Clear PORTD
  // Set TMR0 mode, and assign prescaler to TMR0 to create a delay of 1 ms
  // clear TMROL
  // active Timer0 interruption
```

```
  digit = 0;
  portd_index = 0;
  shifter = 1;
  number = 9950;     // Initial number value
  do {
    // extract thousands from number an put it in digit
    // and store the corresponding mask to PORTD array index 3
    // extract hundreds from number an put it in digit
    // and store the corresponding mask to PORTD array index 2
    // extract tens from number an put it in digit
    // and store the corresponding mask to PORTD array index 1
    // extract ones from number an put it in digit
    // and store the corresponding mask to PORTD array index 0
    delay_ms(1000);
    number++ ;
    if (number > 9999)
       number = 9950;}
    while(1);
}
```

1. Complete le programme (lignes rouge) par les instructions nécessaire.
2. Autoriser l'utilisation des 4 afficheurs (SW4.1, SW4.2, SW4.3, SW4.4 on)
3. Créer un répertoire TP3/DISO123/
4. Coller les fichiers DIS0123.c et Display_utils.h dans ce répertoire
5. Créer un nouveau projet DIS0123, fréquence 8 MHz
6. Dans la seconde fenêtre du New Project Wizard, ajouter les 2 fichiers DIS0123.c et Display_utils.h
7. Compiler et programmer le µC, puis verifier le bon fonctionnement de système.

Exercice 4: Timer 0 Mode compteur

Programme permet de compter le nombre des cliques sur la branche RA4 et affiche le résultat sur PORTD.