Tp – Algorithme Langage C.
Séance N°3

Compétences : C3.10 Réaliser la conception détaillée d'un module matériel et/ou logiciel.
Savoir Associé : S4.2. Algorithmique

Pré-requis : Séance N°2 Validée

Cette troisième séance permet de finir la fiche sur les algorithmes. D'un point de vue matériel, nous aborderons la programmation de l'afficheur 7 Segments.

1. Algorithme - Algorigramme.

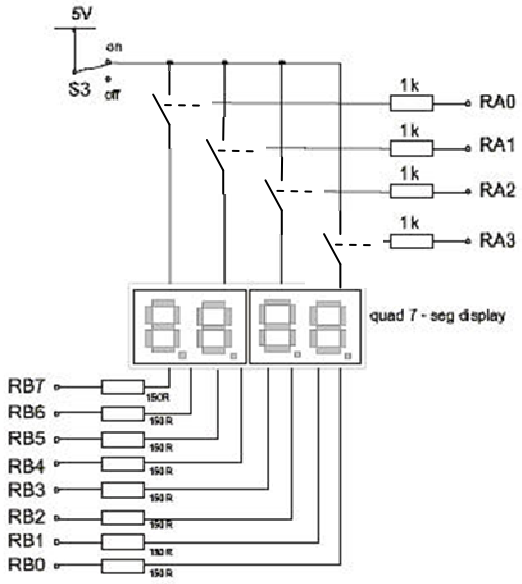
1.1. Compléter les deux dernières fiches :

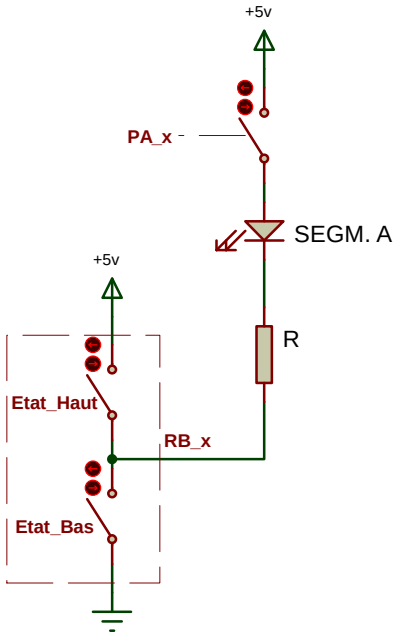
- Boucle Pour : nombre d'itérations connu.

Le fichier doit être complet.

2. Etude structurelle de l'afficheur 7 Segments .

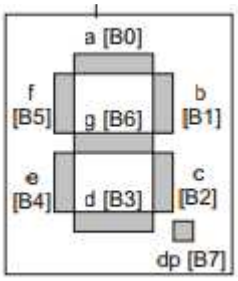
Un afficheur 7 segments comportent 7 led ayant une broche commune et une broche indépendante. Celle-ci permet l'allumage ou non des différents segments.
La carte de développement comporte 4 afficheurs, chacun d'entre eux sera activé par une broche du PORTA. Les segments seront validés par une broche du port B.





Le schéma simplifié à droite montre le principe de la commande d'un segment. A partir des deux schémas préciser :

- L'état (alimenté ou non) de l'afficheur en fonction de la tension RA
- Le niveau électrique (haut ou bas) sur RB_x qui permet d'allumer la led du segment.



L'appellation de chaque « led » de l'afficheur respecte une convention. LE segment a se trouve en haut, puis la rotation s'effectue dans le sens horaire.

Programme N°1 : Je vérifie le fonctionnement des afficheurs 7 segments.

- Proposer un programme simple permettant de valider le fonctionnement des afficheurs :
  - Sélectionner successivement les afficheurs par le PORTA
  - Valider chaque « led » de chaque afficheur avec le portB.

3. Codage de symboles sur afficheur 7 segments à l'aide de tableau.

3.1. Tableau en langage C

Un tableau est un ensemble ordonné de case contenant des données d'un même type. L'accès en écriture ou en lecture se déroule de la même façon que pour une variable à ceci prêt qu'il faut préciser l'indice de la case.

3.2. Exemples :

```
//déclaration
#define NombreDeCases 5
int uneVariable ;
int unTableau[NombreDeCases] ;   // ici les cases contiennent n'importe quoi.
//ou
int unTableau[]={3,10,255,19,3} ; // les cases sont initialisés à la déclaration
```

Représentation concrète du tableau `unTableau[]` et de son contenu initialisé :

Indice	0	1	2	3	4
contenu	3	10	255	19	3

```
//affectation
uneVariable =0;  //met 0 dans la variable
unTableau[0]=15; //remplace le contenu de la première case par la valeur 15
//consultation
uneVariable= unTableau[2]; // le contenu de la case 2 (255) remplace le contenu
de uneVariable
```

Programme N°2 : Codage des symboles sur un afficheur 7 segments à l'aide d'un tableau.

En consultant la page Wikipedia sur les afficheurs 7 segments, proposer un programme qui teste sur un afficheur avec les 10 symboles de la base 10. Vous utiliserez un tableau pour le codage : chaque case contiendra le code à appliquer sur le port B. L'indice et le code seront en cohérence c'est-à-dire : l'Indice 3 correspond au code numérique 3 que l'on souhaite afficher.
Améliorer le programme pour afficher les symboles de la base 16.
Une erreur s'est glissée sur la page wikipédia. Trouvez là.

4. Fonction avec passage de paramètre.

4.1. Déclaration d'une fonction avec paramètres

Il faut déclarer la ou les variables qui seront transmises à la fonction.

4.2. Exemples :

```
//déclaration et définition
Void fonction1(int param1, int param2)
{
// instructions qui peuvent utiliser param1 et/ou param2
}
```

Toute fonction peut être appelée par une fonction. La fonction `main()` (programme principale) est la seul exception.

```
// appel de la fonction
main()
{
Int uneVariable=5 ;
Fonction1(12, uneVariable) ;
}
```

Dans l'exemple ci-dessus, on appelle `fonction1()` en lui transmettant les valeurs **12** et **5** qui initialiseront les variables `param1` et `param2` pendant l'exécution de la fonction.

Programme N°2 : Affichage d'une valeur sur 4 Digits (éléments).
Réaliser une fonction affiche(Noafficheur,code) qui assure l'affichage d'un code sur l'afficheur dont on aura précisé le N° (1-4) et le code à afficher (0-9)

4.3. Sauvegarder le programme et commenter-le.

5. Fonction avec valeur de retour.

5.1. Déclaration d'une fonction avec valeur de retour

Les fonctions peuvent être appelées en proposant une valeur de retour qui pourra :

- modifier le contenu d'une variable par affectation
- faire l'objet d'une expression booléenne.

5.2. Exemple

```
//déclaration
int NomFonction(type 1 param1, type1 param2,...)
{
int unResultat ;
unResultat  = param1+10*param2 ; //calcul bidon pour expliquer

Return unResultat ;
}
```

Le mot clé **return** provoque l'arrêt de la fonction et renvoi la valeur contenu dans la `unResultat` qui doit être du même type que la fonction.

```
//appel de la fonction et récupération du résultat
int uneAutreVariable;

uneAutreVariable= NomFonction(10,2); // affectation du résultat

if( NomFonction(10,5)>10 ) //appel de la fonction dans une expression
{                       // booléenne. Ici, la valeur de retour sera comparée à 10

}
```

Programme N° 3 : Comptage d'évènement et affichage.

5.3. Sauvegarder le programme et commenter-le.