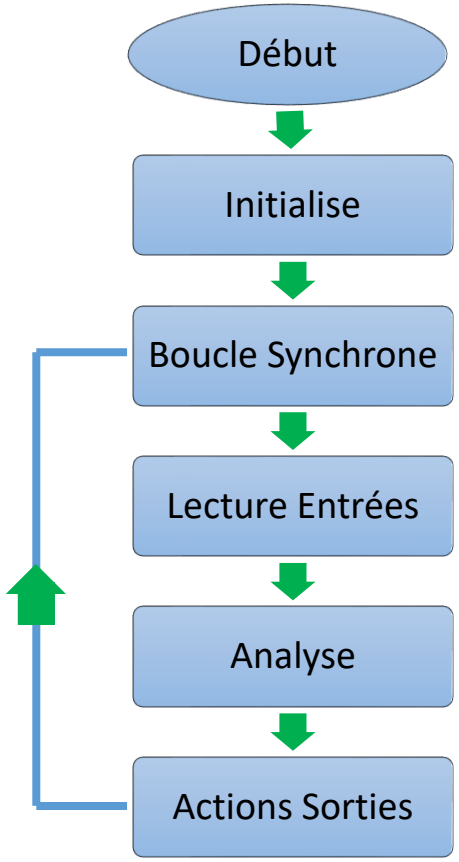
Electronique Numérique

Systèmes à microprocesseur

S6 : Techniques de Programmation

Frédéric Giamarchi
IUT GEII Nîmes
Centre Spatial Universitaire (Montpellier Nîmes)
Université de Montpellier
frederic.giamarchi@umontpellier.fr

Structure d'un programme



Chaque bloc représente une fonction organisée par la fonction main().

- Initialise : initialisation du µC, de ses périphériques utilisés et de ses variables.
- Boucle Synchrone : Boucle qui exécute le programme principal, à intervalle de temps régulier.
- Lecture Entrées : enregistre les informations externes et internes (registres)
- Analyse : Traite les informations reçues et prend des décisions.
- Actions Sorties : Met à jour les lignes en sorties du µC et les registres internes.

Utilisation de Fonctions

- Le but des fonctions est de rendre plus lisible un programme et de pouvoir réutiliser ces fonctions.
- Une fonction est un petit programme séparé du programme principal. Il peut être appelé à la demande.

- Il existe 4 types de fonctions :
  - void Ma_Fonction (void) — exécuter le contenu de la fonction
  - void Ma_Fonction (int data0, int data1) — exécute la fonction avec un ou plusieurs paramètres
  - int reponse Ma_Fonction (void) — exécute la fonction et renvoie un résultat
  - Int reponse Ma_Fonction (int data0, data1) — exécute la fonction avec un ou plusieurs paramètres et renvoie un résultat

Utilisation de Fonctions

- Le but des fonctions est de rendre plus lisible un programme et de pouvoir réutiliser ces fonctions.
- Une fonction est un petit programme séparé du programme principal.
- Il peut être appelé à la demande.

- La 1ère fonction d'un programme en C est la fonction :
  int main() { } c'est le point d'entrée de tous les programmes en C.

- Forme d'une fonction :

```
Type NomFonction (arguments)
{
    // Instructions effectuées par la fonction
}
```

La fonction : void Ma_Fonction (void)

- C’est une fonction qui exécute une liste actions, lorsqu'elle est appelée.
- Exemple de fonction qui définit des états en début de programme
  - Les noms employés doivent avoir été créés préalablement.

```
void Initialisations (void)
 {
     etat = 0;                  // La variable état est mise à zéro.
     LED_Rouge = 0;             // La led rouge est éteinte.
     MOTEUR = 0;                // Le moteur est stoppé.
}
```

La fonction : type Ma_Fonction (type data)

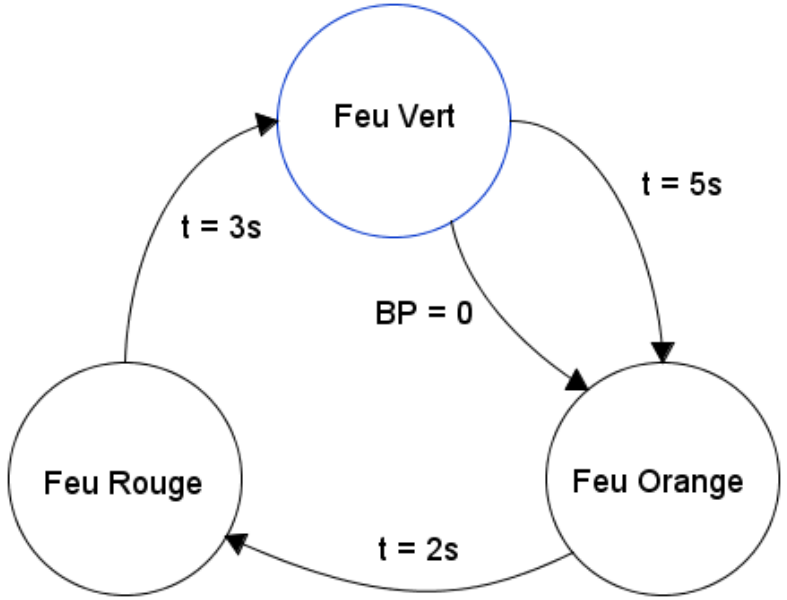
- Exemple de fonction pour calculer la valeur moyenne de 2 nombres

```
float Moyenne(float a, float b)
{
    float resultat;              // On crée un variable en mémoire locale
    resultat = (a+b)/2;          // On réalise l'opération

    return resultat;             // On indique la valeur qui sort de la fonction
}
```

Machine d'états

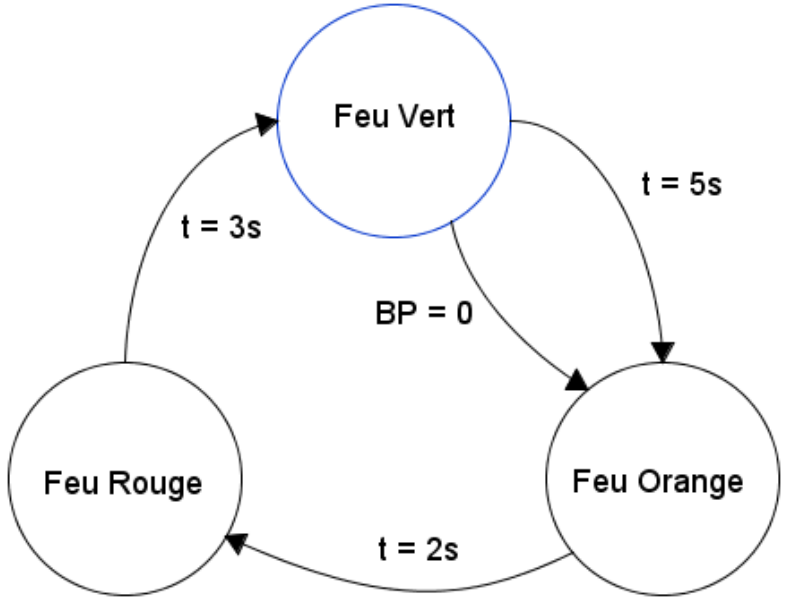
- Exemple : Feu de circulation
  - 3 états (actions)
  - 4 transitions (évènements)




```
switch (etat)
{
        case 0:
                "Feu vert allumé"
                break;
        case 1:
                "Feu orange allumé"
                break;
        case 2:
                "Feu rouge allumé"
                 break;
}
```

Machine d'états

- Exemple : Feu de circulation
  - 3 états (actions)
  - 4 transitions (évènements)




```
switch (etat)
{
        case 0:          "Feu vert allumé"
                si (t=5s ou appui BP)
                    alors etat = 1
                break;
        case 1:          "Feu orange allumé"
                si (t=2s)
                    alors etat = 2
                break;
        case 2:          "Feu rouge allumé"
                si (t=3s)
                    alors etat = 0
                 break;
}
```

Synchronisation des actions

- Les temporisations sont des boucles d'attente.
  - Aucune autre action n'est possible.
  - Elles réduisent le temps de traitement.
  - Elles influencent la périodicité des actions.
- Les temporisations sont remplacées par des compteurs.
  - Il y a autant de compteurs qu'il y a d'actions indépendantes.
  - On peut aussi utiliser des « Ticker » pour la gestion des évènements périodiques.