Microcontroleurs 8 bits : Langage C
Organisation des données, structures et unions

Le langage C permet de 'structurer' les VARIABLES, c'est à dire de les organiser en GROUPES logiques et cohérents.

⇨ *Un programme dont les données sont organisées, structurées, est plus facile à comprendre, à maintenir et a faire évoluer.*

Exemple :

Dans un programme d'horloge il peut etre avantageux de considérer la 'date' comme une variable unique ou 'groupe' dont les éléments (membres) sont les suivants : secondes, minutes, heure, jour, mois, année.

La notion de STRUCTURE permet de définir le groupe et ses membres.

```
struct date {
unsigned char        seconde;
unsigned char        minute;
unsigned char        heure;
unsigned char        jour;
unsigned char        mois;
unsigned int         annee;
};
```

Une telle déclaration crée un nouveau MODELE de variable comportant 6 éléments .

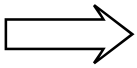
On peut alors CREER une variable (structure) utilisant le modèle 'date' :

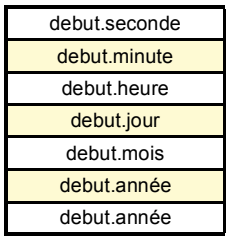
```
struct date debut;
```

La structure 'debut', construite sur le modèle 'date' comporte 6 membres dont les noms individuels sont :

debut.seconde, debut.minute, debut.heure, debut.jour, debut.mois et debut.annee

7 cases mémoires (7 octets) seront alors réservées pour mémoriser l'ensemble de la structure 'début'





Chaque MEMBRE du GROUPE a une place réservée !!

int = 16 bits

On peut aussi rassembler la définition de la structure et la création de la variable :

```
struct date {
unsigned char        seconde;
unsigned char        minute;
unsigned char        heure;
unsigned char        jour;
unsigned char        mois;
unsigned int         annee;
} debut ;
```

OU

```
struct    {
unsigned char        seconde;
unsigned char        minute;
unsigned char        heure;
unsigned char        jour;
unsigned char        mois;
unsigned int         annee;
} debut;
```

Utilisation optimale de la RAM

En langage C classique l'octet (8 bits) est la plus petite taille de variable. ('char' ou 'unsigned char')

Dans les programmes industriels on utilise souvent des 'drapeaux' ou indicateurs dont la valeur vaut, selon les instants, 1 ou 0. Un seul bit suffit à mémoriser l'état d'un drapeau.

Dans le cas des microcontroleurs 8 bits économiques comme les PIC l'espace mémoire en RAM est de petite taille , souvent 256 octets.

On ne peut donc se permettre de gaspiller des cases mémoire de 8 bits pour n'y utiliser qu'un seul bit à chaque fois.

Pour utiliser au mieux la RAM les programmes du Projet 2006 utilisent la technique des STRUCTURES et de l'UNION.

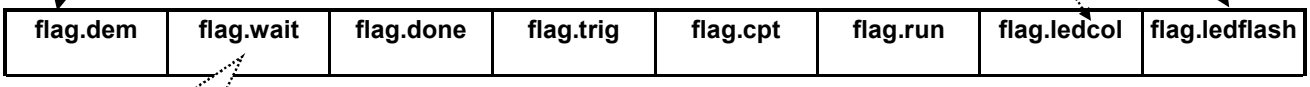
Les différents 'drapeaux' (1 bit) sont regroupés dans des octets (8 bits) tout en conservant la possibilité d'action individuelle sur chacun d'entre eux. On peut également agir collectivement sur les 8 bits.

Exemple pris dans SFR04.H :

```
union flag_srf04                    // divers bits drapeaux pour la gestion du sonar
{
      unsigned char flags;
      struct f_srf04
      {
           unsigned dem:1, wait:1, done:1, trig:1, cpt:1, run:1, ledcol:1, ledflash:1;
      } flag;
};
```

largeur 1 bit

flags = 1 octet = 1 fois 8bits



1 case = 1 bit

flag. = 8 fois 1 bit

⇨ Après cette double définition (individuelle et collective) on peut, selon les besoins, écrire dans le programme :

```
flags = 0                 (pour mettre à zéro les 8 bits simultanément)
```

ou :

```
flag.ledflash = 0         (pour agir exclusivement sur l'un des drapeaux)
```

Le projet 2006 pousse plus loin la structuration des données car ce 'paquet' de 8 bits est lui même intégré dans une structure plus vaste nommée 'sonar' :

```
struct srf04
{
      union flag_srf04      srff;
      int                   cpt;
      int                   timer;
      unsigned char         ptmr0;
      unsigned char         ltmr0;
      long int              distance;
      int                   cptflash;
} sonar;
```

⇨ On accède alors au drapeau 'flag.ledflash' précédent par : `sonar.srff.flag.ledflash = 0;`