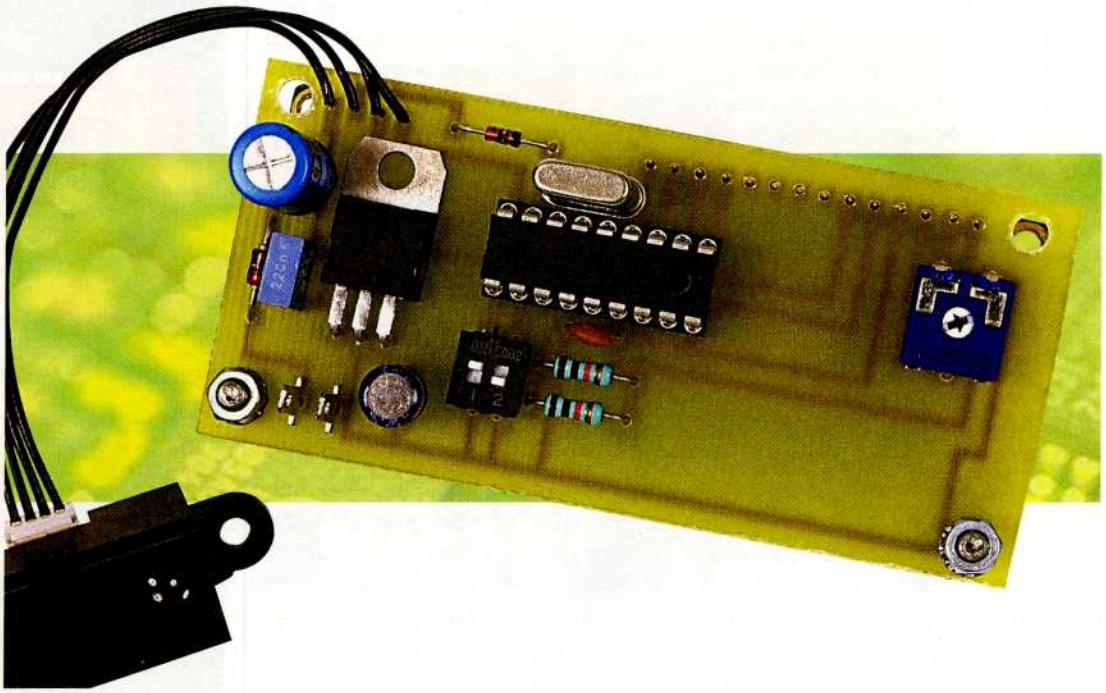
À la découverte des microcontrôleurs PIC

(Deuxième partie)



Nous voici comme promis dans notre dernier cours au coeur d'un microcontrôleur PIC, l'architecture présentée figure 6 est celle d'un microcontrôleur PIC 16 F 84 sur lequel notre étude portera.

Avant d'étudier l'architecture interne d'un PIC, nous allons faire un bref rappel sur les bascules, pour nous amener jusqu'au "circuit registre", constituant essentiel dans l'architecture d'un PIC.

Il existe en logique combinatoire différentes bascules ayant chacune des caractéristiques et un mode de fonctionnement propre. Nous allons étudier ici le fonctionnement de l'une d'elles : la bascule D.

Une bascule D peut être construite à l'aide de 4 portes NAND et d'un inverseur tel que représenté sur la **figure 1**.

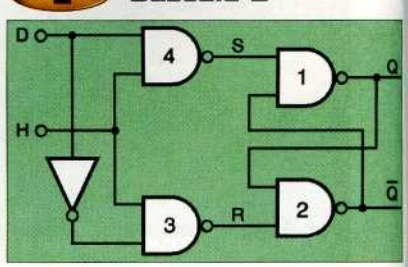
Mode de fonctionnement

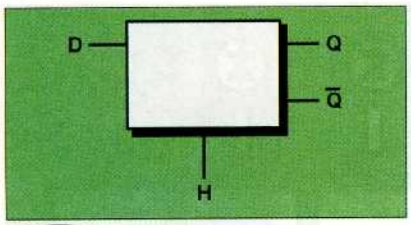
Tant que l'entrée de validation H est au niveau logique bas (0 V) la sortie des portes NAND (3 et 4) est au niveau logique haut. En effet un 0 V appliqué sur une des entrées d'une porte NAND provoque la mise au niveau logique "1" de la sortie de cette porte, on dit que le "0" est l 'élément absorbant sur une NAND (voir la table de vérité **figure 3**). La sortie des portes 3 et 4 restera "bloquée" au niveau logique "1" tant que l'entrée H est à 0, ce qui signifie que le signal présent sur l'entrée D (data) ne modifiera pas les sorties des portes 3 et 4 (sortie R et sortie S). La sortie Q de la bascule D restera donc dans l'état précédent, on peut dire que la bascule D a mémorisé l'état antérieur (voir la table de vérité).

Passons maintenant l'entrée H au niveau logique "1". Pour une porte NAND le niveau logique "1" sur l'une de ses entrées représente "l'élément neutre", c'est à dire que la sortie de la porte ne dépendra alors que de l'état de la deuxième entrée logique (**figure 3**). Si l'entrée D est au niveau logique 1, la sortie de la porte 4 (S) passe au niveau logique "0", ce qui provoque une mise au niveau logique "1" de la sortie de la porte 1 appelée "Q".

Un inverseur est inséré entre le signal d'entrée D et la porte 3. De

1 Bascule D





2 Symbole

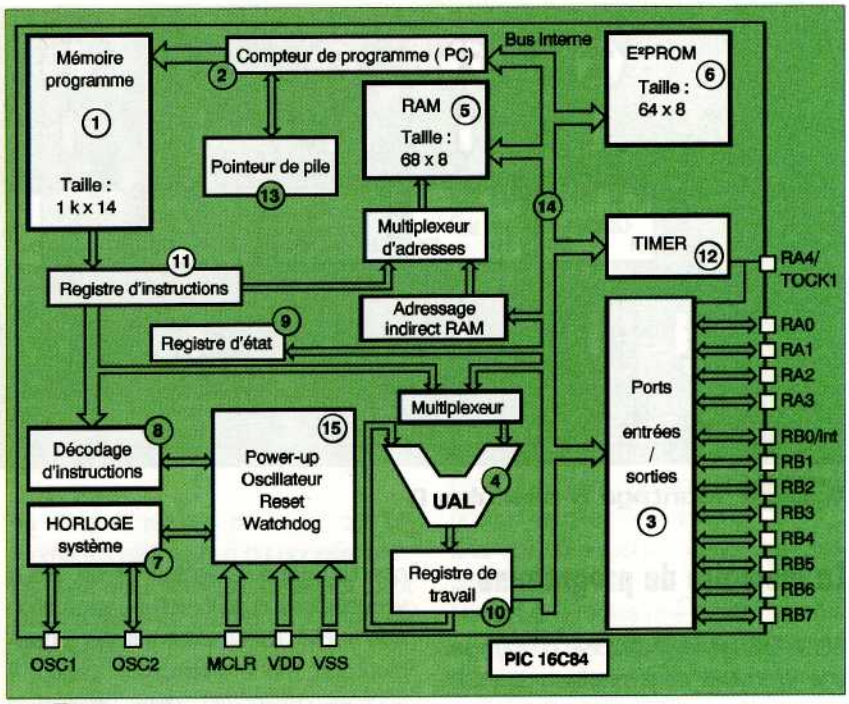
A	B	S
0	0	1
0	1	1
1	0	1
1	1	0

3 Table de vérité

ce fait, sachant que nous avons mis précédemment D à 1 cela signifie que la sortie de la porte 3 est au niveau logique "1". La deuxième entrée de la porte 2 est connectée sur la sortie de la porte 1 qui est au niveau logique "1", de ce fait la sortie Q barre de la porte 2 est donc au niveau logique "0".
Conclusion : On peut dire que la sortie Q d'une bascule D recopie l'état de l'entrée D tant que le signal de validation H est au niveau logique "1". Si le signal H est au niveau logique "0", alors la bascule peut être assimilée à une cellule mémoire de 1 bit, car elle mémorise l'état antérieur de la sortie Q.
Comment utiliser des bascules D pour former une mémoire statique ?

De la bascule à la mémoire ...

Nous venons de voir le fonctionnement d'une bascule D unique, celle-ci peut être assimilée à une cellule mémoire de 1 bit (en effet on ne mémorise qu'une seule information binaire). Pour réaliser par exemple une mémoire 8 bits (1 octet), on pourra juxtaposer 8 bascules D comme représenté sur le schéma de la **figure 5**.
La donnée à mémoriser est présentée, via les interrupteurs, sur les entrées Data des bascules D (D0 à D7), puis lorsque l'on veut mémoriser l'état de ces 8 entrées, il suffit d'appliquer une impulsion sur la broche de validation H (remarquez que toutes les entrées de validation H sont reliées ensembles). Les



6 Architecture interne simplifiée du PIC 16 F 84

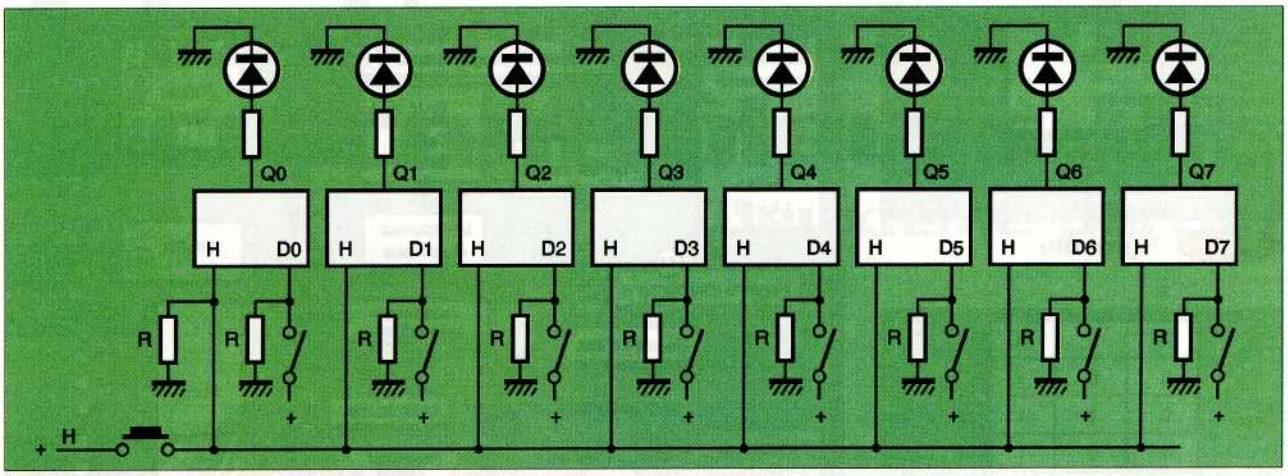
1 - Mémoire programme
2 - Registre compteur de programme
3 - Port A et Port B d'entrées - sorties
4 - Unité Arithmétique et logique
5 - RAM
6 - E2PROM
7 - Horloge système
8 - Registre de décodage des instructions
9 - Registre d'état
10 - Registre de travail
11 - Registre d'instruction
12 - Timer
13 - Pointeur de pile
14 - Bus internes
15 - Reset ; Watch dog ; Alimentation

8 leds connectées sur les sorties Q donnent l'état du contenu de chaque bascule. Le montage ci-après peut se réaliser à l'aide d'un circuit TLL comportant 8 bascules D tel que le 74374.
Cette mémoire 8 bits peut également s'appeler registre 8 bits. Dans l'architecture interne d'un microcontrôleur PIC nous retrouverons de nombreux registres semblables à celui-ci.
Le PIC 16F84 possède 15 registres ayant chacun une fonction bien définie.

Rôle et description des principaux blocs constituants l'architecture d'un PIC

4 Tableau des états

D	H	Q	$\overline{Q}$	Commentaires
1	0	0	1	État initial
1	1	1	0	La sortie Q prend l'état de D («1»)
1	0	1	0	Mémorisation de l'état antérieur
0	0	1	0	Mémorisation de l'état antérieur
0	1	0	1	La sortie Q prend l'état de D («0»)
0	0	0	1	Mémorisation de l'état antérieur
1	0	0	1	Mémorisation de l'état antérieur
0	1	0	1	Recopie de l'entrée D
1	1	1	0	Recopie de l'entrée D



5 Montage 8 bascules D

La mémoire de programme

Sur le PIC présenté (16F84) on retrouve une mémoire de type flash EPROM ayant une capacité de 1024 instructions (rep : 1). Le constructeur donne environ 1000 cycles d'effacement et d'écriture pour cette mémoire, ce qui nous laisse une marge assez confortable pour mettre au point un programme. Chaque instruction est codée sur 14 bits cela signifie que la mémoire programme du PIC 16F84 à une capacité de 1024x14 bits.

C'est dans cette mémoire que sera stocké votre programme (compilé) qui correspondra aux instructions que devra effectuer le microcontrôleur. Les 1024 (1 k) instructions possibles semblent un peu dérisoires face aux "méga-octets" d'aujourd'hui, mais vous verrez par la suite que cela suffit largement pour une application "grand public". Il ne faut pas oublier également que dans chaque case mémoire on peut stocker 14 bits, ce qui permet d'utiliser des instructions plus "puissantes" que dans une mémoire traditionnelle 8 bits.

A la mise sous tension c'est la case située à l'adresse 0 qui sera lue (c'est ce que l'on appel le vecteur reset). Nous verrons par la suite lors d'une application que le PIC 16F84 peut travailler également en mode interruption (selon 4 sources différentes), dans ce cas la case mémoire qui sera pointée se trouve à l'adresse 4 (c'est le vecteur interruption).

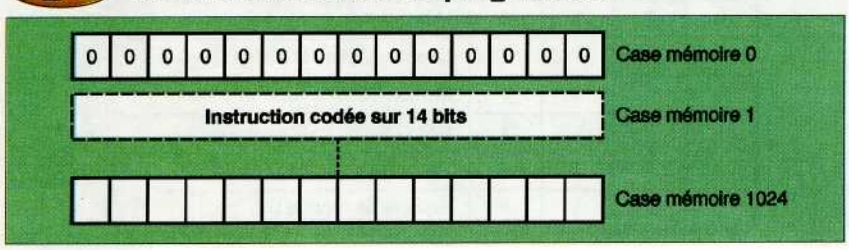
Le compteur de programme

(CP ou PC en anglais pour Programm Counter)

Nous venons de voir que la mémoire programme contient les codes binaires du programme que nous avons défini. Nous avions également vu dans le premier cours que le microcontrôleur exécute une à une les instructions stockées dans la mémoire.

Un registre interne nommé "compteur de programme" (**rep : 2**) va être chargé de pointer (sélectionner) chaque case mémoire une à une, afin que le microcontrôleur puisse exécuter l'instruction correspondante au code binaire stocké dans la case mémoire concernée. Le contenu du registre du pointeur de programme augmentera au grès de l'exécution des instructions, le PC pointe toujours la prochaine instruction à exécuter.

7 Instructions mémoire programme



Dès que le microcontrôleur est alimenté le contenu du registre compteur de programme est remis à zéro ce qui fait que c'est case mémoire située à l'adresse 0 qui va être pointée la première (vecteur RESET).

Le compteur de programme possède une largeur 13 bits, il peut donc adresser une mémoire de 8 k ($2^{13} = 2^{10} \times 2^3$ soit 1 k x 8 = 8 k).

Le compteur de programme s'est incrémenté alors que l'exécution de l'instruction précédente est en cours.

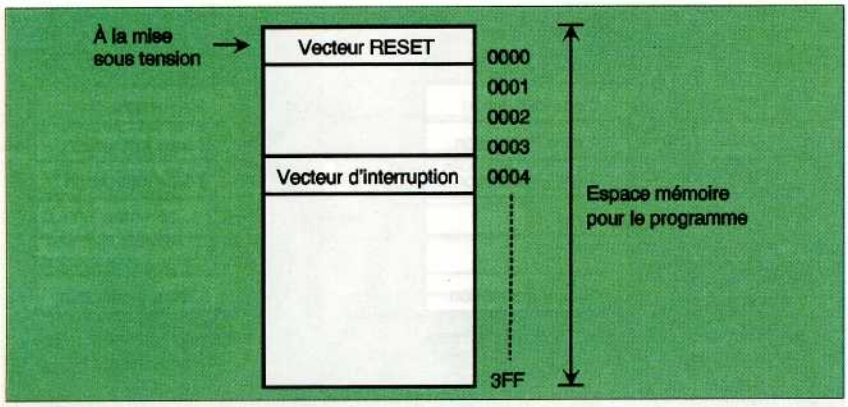
Cheminement d'une instruction

Le registre de contrôle et décodage des instructions

Le programme binaire correspondant à votre source est maintenant dans la mémoire programme du microcontrôleur PIC, le compteur de programme pointe l 'instruction à exécuter, cette instruction est analysée par un registre de contrôle et de décodage des instructions, véritable analyseur logique, qui est chargé de définir ce que le microcontrôleur devra effectuer comme opération. Le contrôleur et décodeur d'instruction définit la stratégie des actions à accomplir en interne pour effectuer l'instruction demandée.

L'unité arithmétique et logique

L'unité arithmétique et logique est considérée souvent comme étant le coeur de



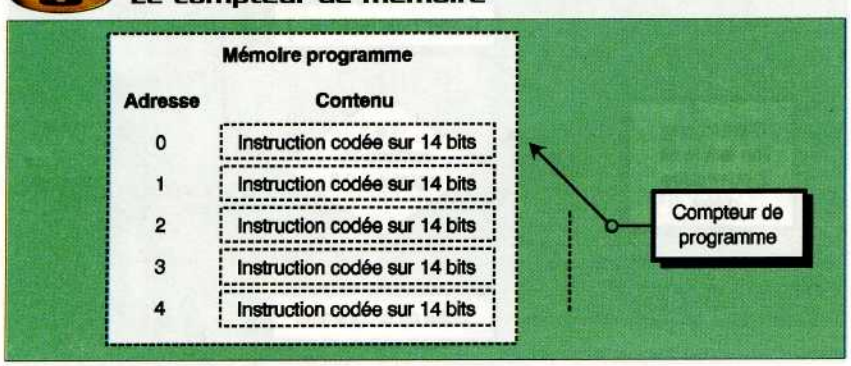
8 Espace mémoire

l'unité centrale (UC). En effet c'est elle qui va être chargée d 'effectuer toutes les opérations de type arithmétique (addition, soustraction etc...) ou bien de type logique (rotation, décalage, complément etc...).
Selon l'opération à effectuer le contrôleur et décodeur d'instruction enverra les signaux nécessaires à l'unité arithmétique et logique pour pouvoir accomplir l'opération demandée.
Exemple : Nous voulons soustraire deux nombres, comment l'unité centrale va-t-elle procéder ?
- Dans un premier temps la première valeur à soustraire va être stockée dans le registre de travail W, remarquez que le registre de travail est relié sur une des entrées de l'Unité Arithmétique et Logique.
- Puis la deuxième valeur à soustraire est dirigée vers une autre entrée de l'Unité Arithmétique et Logique. Ensuite un code indiquant qu'une soustraction doit être effectuée est envoyé vers l'Unité Arithmétique et Logique qui exécute cette instruction.
- Le résultat de la soustraction est stocké dans le registre de travail qui lui-même est relié au bus de donnée interne, cela veut dire que le résultat peut être transféré en interne vers n'importe quel registre.

Le registre de travail
(registre W rep : 10)

L'Unité Arithmétique et Logique est en étroite relation avec un registre nommé W (work register), c'est un registre de travail qui correspond aux anciens "accumulateurs" sur les microprocesseurs et par lequel vont transiter un bon nombre d'informations que ce soit une donnée à traiter (pour réaliser par exemple une addition, une soustraction etc...) ou bien pour stocker le résultat d'une opération ou d'un traitement.
Nous verrons par la suite lorsque nous réaliserons des programmes que ce registre est très important. En effet l'accès à certains registres du PIC ne peut se faire directement, nous sommes obligés de "passer" la valeur à lire ou à écrire par ce fameux registre de travail. Le fait que le registre de travail soit relié au bus de données interne permet à celui-ci d'être en relation avec le reste de l'architecture du PIC (RAM, E2PROM, TIMER, PORTS A et B, etc...).

9 Le compteur de mémoire



Le registre d'état
(ou registre status)

A chaque fois que l'on devra faire un test au cours d'un programme, nous allons utiliser sans le savoir un registre interne appelé registre d'état qui est en relation avec le résultat de la dernière opération demandée au microcontrôleur PIC. C'est un registre qui contient 8 bits ayant chacun un rôle bien particulier.
Exemple d'un test au cours d'un programme :
Nous avons réalisé une temporisation et nous devons tester si celle-ci est terminée pour passer à la suite du programme, comment le microcontrôleur va-t-il gérer ce programme ?
- Pour réaliser une temporisation, nous allons "charger" une valeur dans un registre du PIC, puis nous allons décrémenter cette valeur jusqu'à atteindre la valeur 0 ce qui définira la fin de notre temporisation. Pour pouvoir dire que le registre que nous avons utilisé est bien à 0, nous allons utiliser une instruction de test qui va nous avertir quand le contenu du registre sera égal à 0. Cette instruction de test contrôle l'état d'un bit du registre d'état, le bit Z (comme zéro) qui passera à 1 lorsque le résultat de la dernière opération effectuée vaudra 0.
Cela peut paraître assez difficile au premier abord mais avec l'habitude.. on se fait à tout. (**figure 13**)

Chronogramme de la temporisation

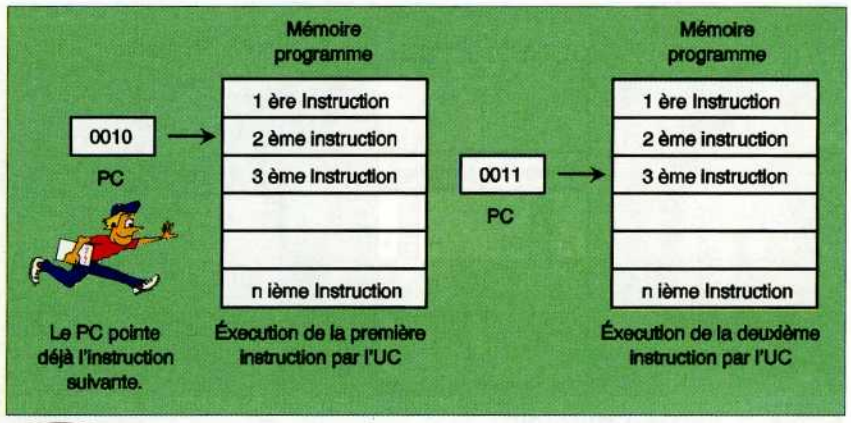
Les bits du registre d 'état

Bit C (carry) : Ce bit du registre d'état va passer à "1" lorsque le résultat de la dernière opération arithmétique a provoqué une retenue.
Bit DC (digit carry) : Ce bit du registre d'état va passer à "1" lorsque le résultat de la dernière opération arithmétique a provoqué une retenue sur les quatre premiers bits, ce bit (ou bien flag... pour drapeau) sera utilisé lorsque l'on travaillera en BCD (binaire codé décimal).

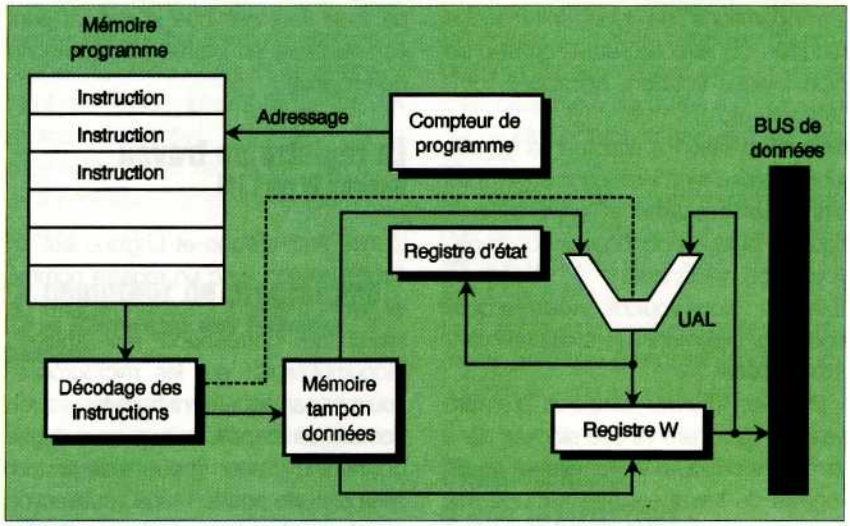
Bit Z (zéro) : Ce bit du registre d'état va passer à "1" lorsque le résultat de la dernière opération est égal à zéro. Dans l'exemple de la temporisation précédente on utilise ce bit zéro.
Bit PD/ (power down) : Ce bit du registre d'état va passer à "0" lorsque le microcontrôleur rencontre l'instruction particulière "Sleep" (mise en sommeil) qui détermine le mode de mise en veille PIC en bloquant les impulsions d'horloge nécessaires au cadencement de tous les échanges, le PIC attend alors un événement pour "repartir".
Bit TO/ (time out) : Ce bit du registre d'état va passer à "0" lorsque le chien de garde interne (nous reviendrons ultérieurement sur son fonctionnement) a atteint la fin de comptage que le programmeur lui a défini. Le chien de garde (ou watch dog) peut être désactivé, il ne servira que lorsque nous voudrons savoir si le programme se déroule correctement.
RP0 et RP1 sont deux bits qui permettent d' accéder à deux zones mémoire RAM différentes (bank 0 et bank 1). Le prochain cours détaille le fonctionnement de ces deux bits
IRP : Bit de réserve pour application future, en ce qui concerne le PIC que nous étudions c'est à dire le PIC 16F84 ce bit ne sert pas.

Le pointeur de pile

Le pointeur de pile (rep : 13) ou SP en anglais (stack pointeur) est un registre pouvant mémoriser huit adresses différentes, on dit que le pointeur de pile est à huit niveaux.
Le rôle du pointeur de pile consiste à mémoriser l'adresse courante lorsque le programme principal est dérouté vers un sous-programme. En effet, lors d'un saut vers un sous-programme, le compteur



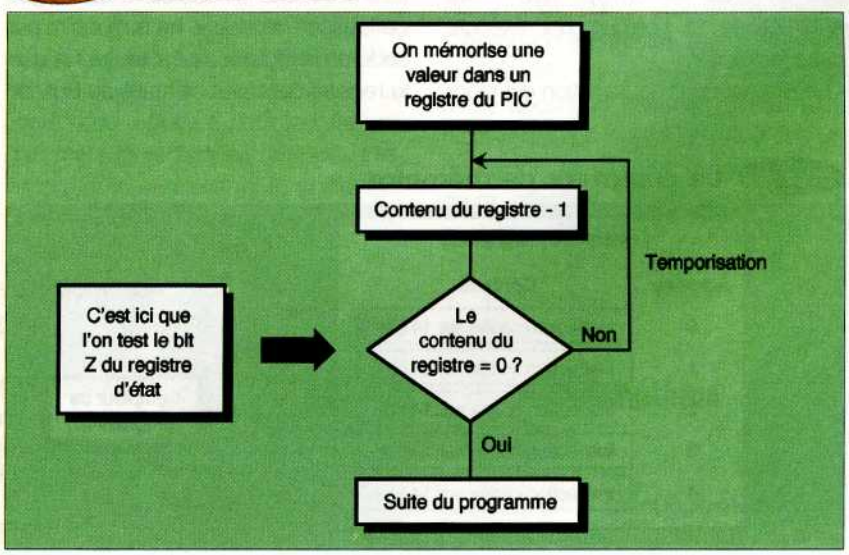
10 Rôle du PC

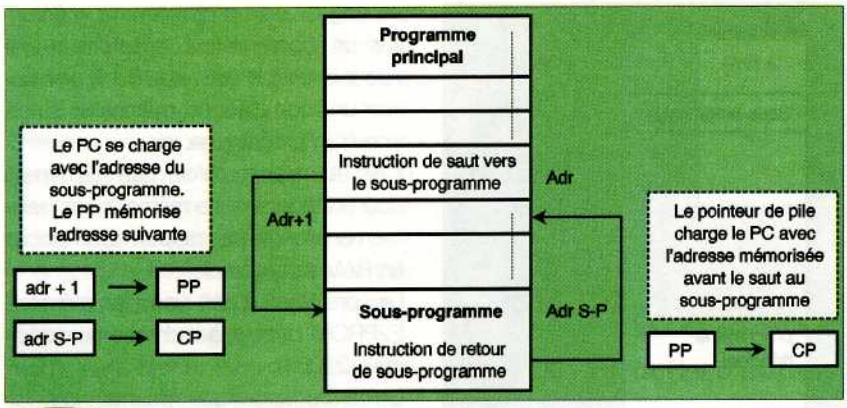


11 Synoptique général

12 Registre de travail W

13 Exemple de test





14 **Rôle du pointeur de pile**

de programme se charge avec l'adresse de celui-ci. Lorsque le sous-programme se termine le compteur de programme doit repointer alors la case mémoire suivant l'instruction de saut (voir figure 1) pour pouvoir reprendre le programme principal ou celui-ci a été dérouté. Le pointeur de pile va alors rechercher automatiquement l'adresse qu'il a mémorisée et il "recharge" le compteur de programme avec cette adresse. Le pointeur de pile sera sollicité dès que dans un programme il y aura un sous-programme. Comme nous le verrons par la suite lors d'un programme fonctionnant avec une interruption, le pointeur de pile aura le même rôle de sauvegarde de l'adresse courante.
Les huit niveaux du pointeur de pile veulent signifier que l'on peut imbriquer huit sous-programmes.
Il est à noter que le pointeur de pile est autonome c'est à dire qu'il gère tout seul la mémorisation et la restitution d'une adresse.

Qu 'est-ce qu 'un sous-programme ?

Un sous-programme est une suite d'instructions correspondant à une fonction bien définie à laquelle votre programme principal fera appel plusieurs fois. En simplifiant, le fait d'écrire un ou plusieurs sous-programmes vous évitera d'écrire plusieurs fois la même chose, d'où une économie de place en mémoire programme et bien sûr une économie de temps.

Prenons un exemple :
Nous devons réaliser un feu tricolore en ayant des temps d'allumage et d'extinction identiques pour chaque lampe, par exemple 3 secondes, la première façon de procéder est la suivante :

- allumage lampe rouge et extinction des autres lampes
- temporisation 3 s
- allumage lampe verte et extinction des autres lampes
- temporisation 3 s
- allumage lampe orange et extinction des autres lampes
- temporisation 3 s
- retour à la première instruction

On voit bien dans cette première façon de réaliser le programme que l'on a écrit trois fois les instructions définissant la temporisation de 3 secondes, ce qui représente en langage assembleur 3 fois 10 instructions soit 30 instructions

La deuxième façon de traiter le problème est de définir un sous-programme de temporisation à qui l'on fera appel autant de fois que nécessaire, ce qui revient à dire que l'on va écrire une seul fois la temporisation de 3 s.

- allumage lampe rouge et extinction des autres lampes
- appel du sous-programme de temporisation 3 s (**1 instruction**)
- allumage lampe verte et extinction des autres lampes
- appel du sous-programme de temporisation 3 s (**1 instruction**)
- allumage lampe orange et extinction des autres lampes
- appel du sous-programme de temporisation 3 s (**1 instruction**)
- retour à la première instruction

Sous programme de temporisation
- temporisation 3 s
- retour de sous-programme

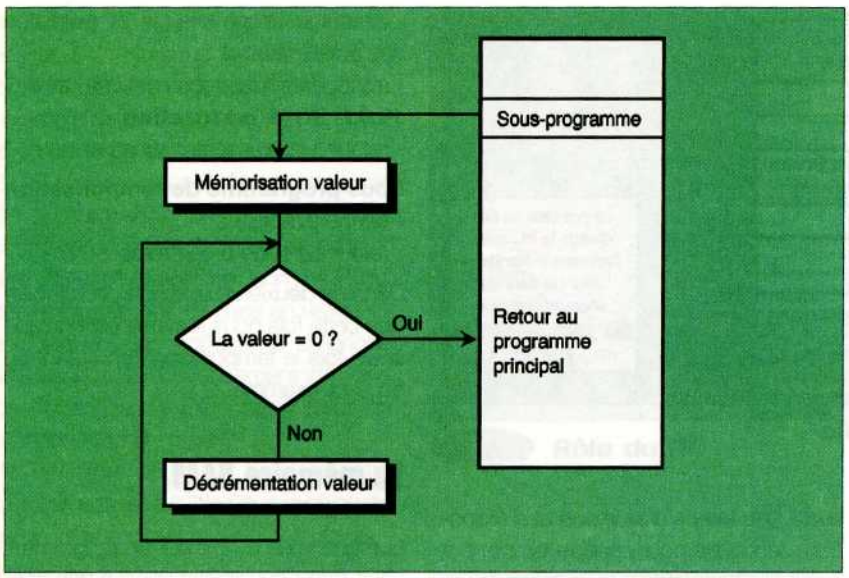
Dans ce deuxième exemple on voit bien que cette fois nous n'avons écrit qu'une seule fois la temporisation en créant un sous-programme.

La mémoire RAM

Lorsque vous allez créer un programme vous allez pouvoir faire appel à des données qui seront stockées temporairement. La mémoire RAM interne du PIC 16F84 vous met à disposition 68 octets banalisés vous permettant à votre gré de sauvegarder ou de rapatrier une information.
Exemple : nous devons réaliser une temporisation, comment allons nous procéder ?
Dans un premier temps nous allons dans un emplacement de la mémoire RAM stocker une valeur qui dépendra de la durée de la temporisation souhaitée. Puis nous allons créer un sous-programme qui va se reboucler (**figure 15**) tant que la valeur que nous avons stockée en mémoire ne vaut pas 0. Dans la boucle du sous-programme bien sûr nous allons décrémenter la variable mémorisée. Une instruction de test va permettre de vérifier le contenu de la variable à analyser.

Organisation de la mémoire RAM

La mémoire RAM est organisée en 128 octets ayant chacun une fonction bien définie (**figure 16**). Nous avons vu précédemment que pour l'utilisateur 68 octets sont réservés.
Les 11 premiers octets de l'espace mémoire sont réservés pour la configuration et l'accès à certains registres spécifiques comme par exemple les ports d'entrées- sorties A et B, le registre STATUS etc...
De l'adresse 11 à l'adresse 79 nous



15 Organigramme

retrouvons les 68 octets réservés à l'utilisateur.

Un peu plus compliqué...

En fait la mémoire RAM est scindée en 2 parties appelées bank 0 et bank 1. Ces deux parties sont accessibles par la même adresse, la sélection de telle ou telle bank se fait selon la position (0 ou 1) de deux bits de contrôle, contenus dans le registre STATUS qui est situé à l'adresse 3 en mémoire RAM. Ces deux bits de contrôle se nomment RP0 et RP1.

Si RP1 = 0 et RP0 = 0 alors on accède à la mémoire située en bank 0

Si RP1 = 0 et RP0 = 1 alors on accède à la mémoire située en bank 1

Lorsque nous allons commencer les cours de programmation nous devrons accéder à différents registres situés dans chacune des deux zones mémoire.

La zone située en bank 1 entre l'adresse 140 et 208 est une image de la bank 0, de plus certains registres tels que le registre STATUS sont accessibles depuis n'importe quelle bank (bank 0 ou bank 1).

La mémoire EEPROM

(Electrically Erasable Programmable Read Only Memory)

La mémoire E2PROM est une particularité du PIC 16F84, c'est une zone comportant 64 octets mise à la disposition de l'utilisateur et dont la particularité est de pouvoir être sauvegardée même en absence d'alimentation.

Cette mémoire peut être très utile pour par exemple : mémoriser le nombre de passages vers un programme, mémoriser un code défaut, mémoriser une valeur invariable (ex : π = 3.14), mémoriser un code d'accès, mémoriser la version d'un programme, etc...

Il est à noter toutefois que les temps pour accéder à cette mémoire sont relativement longs par rapport à un accès en RAM par exemple.

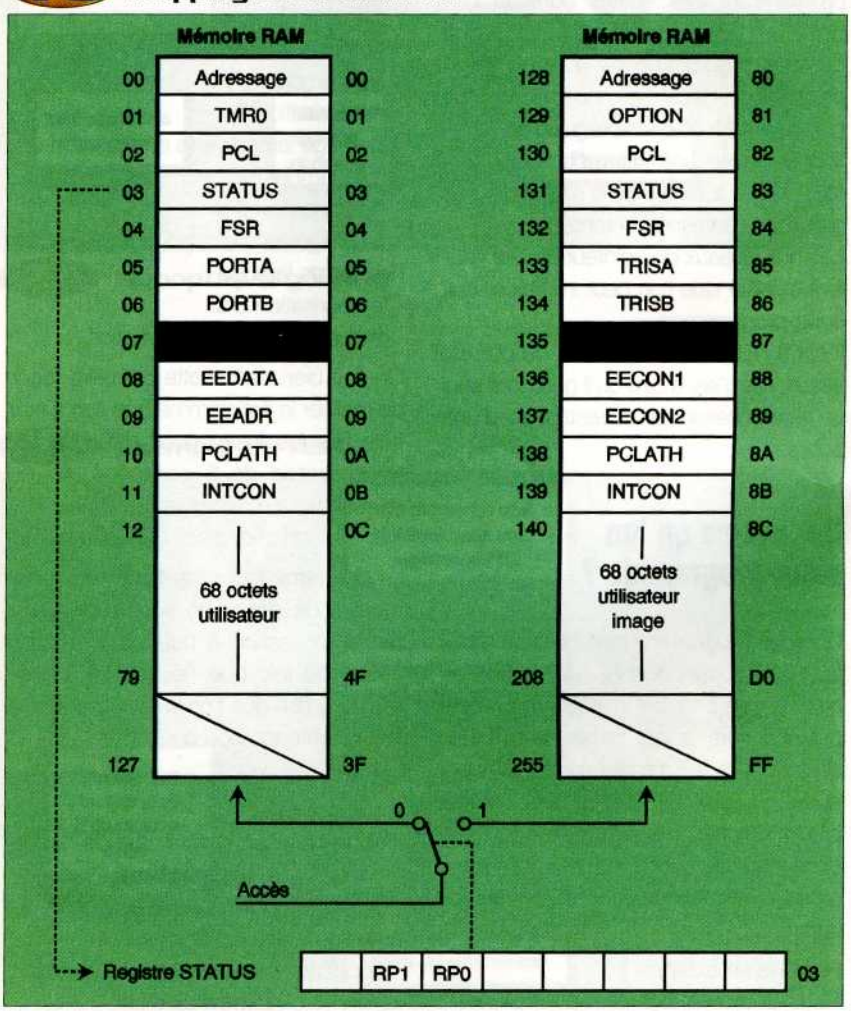
La première case de la mémoire E2PROM débute à l'adresse hexadécimale 2100 h.

Un ordre spécifique permet, lors du transfert du fichier compilé, d'écrire dans la mémoire E2PROM.

Quatre registres spécifiques (EEDATA ; EEADR ; EECON1 et EECON2) permettent les accès à l'E2PROM, que ce soit en lecture ou en écriture. Nous verrons lors d'une application la façon de lire et d'écrire dans cette mémoire un peu particulière.

- Le timer : Un timer est un registre 8 bits dans lequel on charge une valeur. Son

16 Mapping de la mémoire

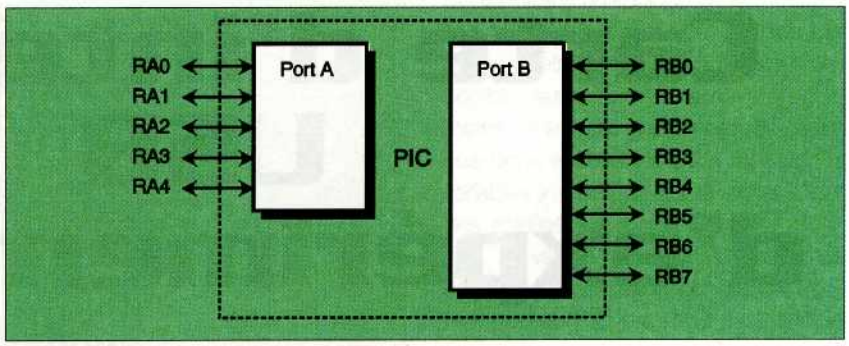


fonctionnement est cadencé soit avec l'horloge interne, soit avec une horloge externe (ou bien des fronts sur une broche spécifique du PIC).
Dès que le timer est validé, il réalise un comptage depuis la valeur que vous avez prédéterminée jusqu'à 255. Il vous prévient (on le verra par la suite en provoquant une interruption) alors que son contenu passe de 255 à 0, puis recommence son comptage.
Nous reviendrons largement sur son fonctionnement exacte et nous réaliserons une application. Cette première approche nous permet de se familiariser avec le TIMER.
Parmi les applications nombreuses pouvant être réalisées avec le TIMER, citons : la base de temps, la temporisation, le comptage, etc... La configuration du TIMER se réalisera dans un registre spécifique du PIC : le registre OPTION.

Les Ports d'entrées - sorties

Pour dialoguer avec l'extérieur (application) le PIC 16F84 vous met à disposition 13 Entrées-Sorties programmables individuellement soit en entrée soit en sortie (**rep : 3**).
Ces 13 entrées - sorties sont issues de 2 ports nommés PORT A pour les cinq entrées - sorties RA0 à RA3 et PORT B pour les huit entrées - sorties RB0 à RB7.

Exemple de configuration du port A et du port B :



17 Les ports d'entrées/sorties

Imaginons que pour réaliser une serrure codée, nous ayons besoin de 4 lignes de sorties pour piloter les 4 colonnes d'un clavier matricé, 4 lignes d'entrées pour recevoir les 4 lignes du clavier et puis 1 entrée pour lancer le programme et 1 sortie pour piloter un relais.

Le schéma équivalent peut être celui de la figure 18

Nous utilisons RB0 à RB3 configurées en sortie pour piloter les 4 colonnes du clavier et RB4 à RB7 pour recevoir les lignes du clavier à 16 touches. Le principe de décodage est le suivant : on passe à "1" séquentiellement chaque colonne et l'on vérifie l'état de chaque ligne, selon la colonne qui est alimentée et la ligne activée, on détermine la touche appuyée. La ligne RA0 du port A est configurée en entrée pour recevoir l'état du poussoir de mise en service et la ligne RA1 est configurée en sortie pour pouvoir actionner un relais.

Dans le prochain cours nous verrons comment programmer les registres de contrôle du port A (TRISA) et du port B (TRISB) afin de configurer leurs broches soit en entrée soit en sortie.

Pour conclure...

Avec cette deuxième partie nous avons abordé simplement la plupart des constituants internes d'un PIC. Le fonctionnement exacte de certains blocs tels que le TIMER, la mémoire E2PROM, le port d'entrées-sorties, le watch dog sera détaillé lorsque nous les utiliserons dans un programme. Le temps est donc venu de s'intéresser un peu au "hard" du PIC 16F84. Nous verrons dans la prochaine partie le brochage ainsi que les conditions de RESET et l'horloge système.

P. Mayeux

18 Exemple d'utilisation des ports d'entrées/sorties