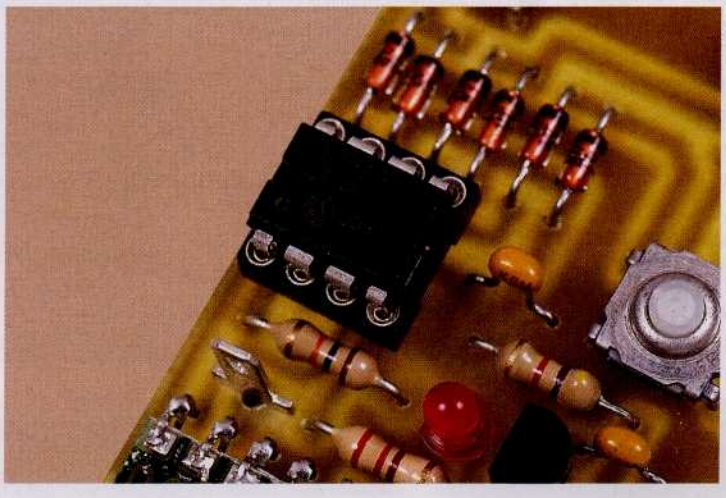
À la découverte des microcontrôleurs PIC (cinquième partie)



Les instructions suite... et fin. Nous allons terminer avec ce numéro les dernières instructions qu'il nous restait à détailler puis, dans notre prochaine leçon, nous analyserons un programme afin de nous familiariser avec tout le jeu d'instructions que nous avons étudié.

3 - Instructions concernant les registres banalisés

Ces instructions concernent tous les types de traitements appliqués sur un registre banalisé. Un registre banalisé peut être par exemple une variable que vous avez déclarée, il peut y en avoir 68 au maximum et c'est en fait la taille de la RAM du PIC. Chaque registre banalisé (ou variable que vous déclarez) possède une taille de 8 bits (1 octet).

Instruction : **ADDWF (ADD W and F)**

Rôle : Cette instruction additionne le contenu du registre de travail w avec le contenu d'un registre banalisé. Le résultat est stocké dans w si l'opérande (voir syntaxe de la commande) d = 0 sinon le résultat sera stocké dans le registre banalisé.

Quand doit-on utiliser cette instruction ?

Lorsque l'on veut additionner une valeur avec un registre banalisé.

Syntaxe : ADDWF + f,d

d = adresse du registre banalisé

f = direction pour le stockage du résultat (0 = résultat dans W, 1 = résultat dans le registre banalisé dont l'adresse est f)

Nombre de cycle(s) d'horloge nécessaire(s) : 1

Exemple d'utilisation :

#define f 1	'la variable f vaut 1
#define w 0	'la variable w vaut 0
monregistre EQU 12	*'on déclare une variable appelée 'monregistre' qui se trouve à l'adresse 12 en RAM (de 12 à 79 soit 68 variables utilisables)*
- CLRW	*'on fait une raz du registre w*
- ADDLW 01	*'on additionne le contenu de w avec 1 donc w=1*
- ADDWF monregistre,f	*'on additionne w et le contenu du registre "monregistre", le résultat est stocké dans "monregistre" car l'opérande "d" ici vaut 1 (f est déclarée à 1en début de programme)*
- ADDWF monregistre,w	'on additionne w et le contenu du registre "monregistre", le résultat est stocké dans "w" car l'opérande "d" ici vaut 0 (w est déclarée à 0 en début de programme)

Instruction : **ANDWF (AND W and f)**

Rôle : Cette instruction fait un "et logique" entre le contenu du registre de travail w et le contenu du registre spécifié f. Selon l'opérande d, le résultat est soit stocké dans w (d=0), soit stocké dans le registre f (d=1).

Quand doit-on utiliser cette instruction ?

On utilise souvent le "ET LOGIQUE" lorsque l'on veut sélectionner un ou plusieurs bits d'un registre, on parle alors de masque.

Syntaxe : ANDWF f,d

f = adresse du registre banalisé
d = direction pour le stockage du résultat (0 = résultat dans W, 1 = résultat dans le registre banalisé dont l'adresse est f)

Nombre de cycle(s) d'horloge nécessaire(s) : 1

Exemple d'utilisation :

#define f 1	'la variable f vaut 1

monregistre EQU 12	*'on déclare une variable appelée 'monregistre' qui se trouve à l'adresse 12 en RAM (de 12 à 79 soit 68 variables utilisables)*
- CLRW	*'on fait une raz du registre w*
- ADDLW 01	*'on additionne le contenu de w avec 1 donc w = 1*
- ANDWF monregistre,f	*'on fait un "et logique" entre le contenu de w et le contenu du registre "monregistre", le résultat est stocké dans "monregistre" car l'opérande "d" ici vaut 1 (f est déclarée à 1 en début de programme)*

Instruction : **CLRF (Clear f)**

Rôle : Cette instruction met le contenu du registre spécifié d'adresse f à zéro.

Quand doit-on utiliser cette instruction ?

Étant donné que l'on utilise souvent les registres banalisés, il est dans certain cas nécessaire de partir avec une valeur nulle dans ce registre.

Syntaxe : CLRF f
f = adresse du registre banalisé

Nombre de cycle(s) d'horloge nécessaire(s) : 1

Exemple d'utilisation :

tempo EQU 12	*'on déclare une variable appelée 'tempo' qui se trouve à l'adresse 12 en RAM (de 12 à 79 soit 68 variables utilisables)*
- CLRF tempo	*'on fait une raz du registre tempo déclaré ci-dessus*

Instruction : **COMF (Complement f)**

Rôle : Cette instruction fait une opération de complément sur le contenu du registre spécifié d'adresse f. En fait, les 0 sont remplacés par des 1 et vice versa. Selon l'opérande d, le résultat est, soit stocké dans w (d=0), soit stocké dans le registre f (d=1).

Quand doit-on utiliser cette instruction ?

Lorsque l'on doit réaliser une opération de complément (inversion de tous les bits) sur un registre.

Syntaxe : COMF f,d
f = adresse du registre banalisé
d = direction pour le stockage du résultat (0 = résultat dans W, 1 = résultat dans le registre banalisé dont l'adresse est f)

Nombre de cycle(s) d'horloge nécessaire(s) : 1

Exemple d'utilisation :

tempo EQU 12	*'on déclare une variable appelée 'tempo' qui se trouve à l'adresse 12 en RAM (de 12 à 79 soit 68 variables utilisables)*
- MOVLW 0x55	*'on met le registre w à 55 (0101 0101)*
- MOVWF tempo	*'on transfert le contenu de w (ici c'est la valeur 55) dans la variable (ou registre banalisé) nommée tempo*
- COMF tempo,1	*'on complémente le contenu de tempo (ici c'est la valeur 55) et on stocke le résultat AA (1010 1010) dans la variable (ou registre banalisé) nommée tempo puisque l'opérande d = 1*

Instruction : **DECF (Decrement f)**

Rôle : Cette instruction fait une décrémentation (contenu - 1) sur la valeur du registre spécifié d'adresse f. Selon l'opérande d, le résultat est soit stocké dans w (d=0), soit stocké dans le registre f (d=1).

Quand doit-on utiliser cette instruction ?

Lorsque l'on doit réaliser une soustraction unitaire sur un registre, pour réaliser une temporisation ou par exemple un décomptage.

Syntaxe : DECF f,d
f = adresse du registre banalisé
d = direction pour le stockage du résultat (0 = résultat dans W, 1 = résultat dans le registre banalisé dont l'adresse est f)

Nombre de cycle(s) d'horloge nécessaire(s) : 1

Dans l'exemple qui suit, nous allons réaliser une temporisation et, pour se faire, nous utilisons l'instruction BTFSS (qui est détaillée plus loin) ainsi que le flag Z du registre STATUS (voir les premières leçons).
Ce flag (ou bit) indique, s'il vaut 1, que le résultat de la dernière opération effectuée vaut 0 et vice versa.

Exemple d'utilisation :

tempo EQU 12	*'on déclare une variable appelée 'tempo' qui se trouve à l'adresse 12 en RAM (de 12 à 79 soit 68 variables utilisables)*
- MOVLW 0x55	*'on met le registre w à 55 (0101 0101)*
- MOVWF tempo	*'on transfert le contenu de w (ici c'est la valeur 55) dans la variable (ou registre banalisé) nommée tempo*
loop	
- DECF tempo,1	*'on décrémente le contenu de tempo (valeur initiale = 55) et on stocke le résultat dans la variable (ou registre banalisé) nommée tempo puisque l'opérande d = 1*
-BTFSS STATUS,2	*'On test le bit Z (zéro) du registre status, si Z=1 alors on saute la prochaine instruction (goto loop) sinon, si le flag Z = 0, alors on revient à l'étiquette 'loop et on redécrémente le registre tempo. Ceci permet de faire une boucle (qui est en fait une temporisation), cette boucle s'effectuera tant que le contenu du registre tempo ne vaut pas zéro*
- GOTO loop	*'retour à l'étiquette 'loop'*

Instruction : **DECFSZ (Decrement f, skip if zéro)**

Rôle : Cette instruction est double, elle fait une décrémentation (contenu - 1 comme l'instruction DECF) sur la valeur du registre spécifié d'adresse f. Selon l'opérande d, le résultat est soit stocké dans w (d=0), soit

stocké dans le registre f (d=1), puis teste ensuite si le contenu du registre spécifié vaut zéro et dans ce cas saute la prochaine instruction.

Quand doit-on utiliser cette instruction ?

Cette instruction est souvent utilisée pour réaliser une temporisation ou par exemple un décomptage.

Syntaxe : DECFSZ f,d
f = adresse du registre banalisé
d = direction pour le stockage du résultat (0 = résultat dans W, 1 = résultat dans le registre banalisé dont l'adresse est f)

Nombre de cycle(s) d'horloge nécessaire(s) : 1 ou 2 selon résultat du test

Exemple d'utilisation :

Reprenons l'exemple de la temporisation réalisée avec l'instruction DECF, nous voyons ici qu'il y a une ligne en moins pour réaliser le même programme.

tempo EQU 12	*'on déclare une variable appelée 'tempo' qui se trouve à l'adresse 12 en RAM (de 12 à 79 soit 68 variables utilisables)*
- MOVLW 0x55	*'on met le registre w à 55 (0101 0101)*
- MOVWF tempo	*'on transfert le contenu de w (ici c'est la valeur 55) dans la variable (ou registre banalisé) nommée tempo*
delay	
- DECFSZ tempo,1	*'on décrémente le contenu de tempo (valeur initiale = 55) et on stocke le résultat dans la variable (ou registre banalisé) nommée tempo puisque l'opérande d = 1 puis on saute la prochaine instruction (goto delay) si le contenu de tempo vaut zéro. Cela permet de faire une boucle comme dans l'exemple avec DECF*
- GOTO delay	*'retour à l'étiquette 'delay*

Instruction : **INCF (INCrement f)**

Rôle : Cette instruction fait une incrémentation (contenu + 1) sur la valeur du registre spécifié d'adresse f. Selon l'opérande d, le résultat est soit stocké dans w (d=0), soit stocké dans le registre f (d=1).

Quand doit-on utiliser cette instruction ?

Lorsque l'on doit réaliser une addition unitaire sur un registre, pour réaliser un compteur. On peut ainsi déterminer par exemple le nombre de défauts constatés sur une machine.

Syntaxe : INCF f,d
f = adresse du registre banalisé
d = direction pour le stockage du résultat (0 = résultat dans W, 1 = résultat dans le registre banalisé dont l'adresse est f)

Nombre de cycle(s) d'horloge nécessaire(s) : 1

Exemple d'utilisation :

tempo EQU 12	*'on déclare une variable appelée 'tempo' qui se trouve à l'adresse 12 en RAM (de 12 à 79 soit 68 variables utilisables)*
- MOVLW 0x55	*'on met le registre w à 55 (0101 0101)*
- MOVWF tempo	*'on transfert le contenu de w (ici c'est la valeur 55) dans la variable (ou registre banalisé) nommée tempo*
- INCF tempo,1	*'on incrémente le contenu de tempo qui passe à 56 (hexa) et on stocke le résultat dans la variable (ou registre banalisé) nommée tempo puisque l'opérande d = 1*

Instruction : **INCFSZ (Increment f, skip if zéro)**

Rôle : Cette instruction est semblable à DECFSZ. Elle fait une incrémentation (contenu + 1 comme l'instruction INCF) sur la valeur du registre spécifié d'adresse f. Selon l'opérande d, le résultat est soit stocké dans w (d=0), soit stocké dans le registre f (d=1), puis elle teste ensuite si le contenu du registre vaut zéro et dans ce cas saute la prochaine instruction.

Quand doit-on utiliser cette instruction ?

Cette instruction est souvent utilisée pour réaliser une temporisation ou par exemple un comptage.

Syntaxe : INCFSZ f,d
f = adresse du registre banalisé
d = direction pour le stockage du résultat (0 = résultat dans W, 1 = résultat dans le registre banalisé dont l'adresse est f)

Nombre de cycle(s) d'horloge nécessaire(s) : 1 ou 2 selon résultat du test

Exemple d'utilisation : Dans cet exemple, nous allons faire une boucle (ou une temporisation) de 256 tops d'horloge, puisque quand le contenu du registre (8 bits) est à 255 et qu'on lui ajoute + 1, il revient à zéro.

tempo EQU 12	*'on déclare une variable appelée 'tempo' qui se trouve à l'adresse 12 en RAM (de 12 à 79 soit 68 variables utilisables)*
- CLRF tempo	*'on fait une raz du contenu du registre banalisé nommé tempo*
delay	
- INCFSZ tempo,1	*'on incrémente le contenu de tempo (valeur initiale = 0) et on stocke le résultat dans la variable (ou registre banalisé) nommée tempo puisque l'opérande d = 1, puis on saute la prochaine instruction (goto delay). Si le contenu de tempo vaut zéro, cela permet de faire une boucle de 256 tops d'horloge*
- GOTO delay	*'retour à l'étiquette 'delay*

Instruction : **IORWF (Inclusive OR W with F)**

Rôle : Cette instruction fait un "ou logique" entre le contenu du registre de travail w et le contenu du registre spécifié d'adresse f. Selon l'opérande d, le résultat est, soit stocké dans w (d=0), soit stocké dans le registre f (d=1).

Quand doit-on utiliser cette instruction ?

On utilise cette instruction pour réaliser par exemple un ou logique entre le registre w et une valeur qui n'est pas figée et qui dépend du contenu d'un registre.

Syntaxe : IORLW f,d
f = adresse du registre banalisé

d = direction pour le stockage du résultat (0 = résultat dans W, 1 = résultat dans le registre banalisé dont l'adresse est f)

Nombre de cycle(s) d'horloge nécessaire(s) : 1

Exemple d'utilisation :

tampon EQU 15	*'on déclare une variable appelée 'tampon' qui se trouve à l'adresse 15 en RAM (de 12 à 79 soit 68 variables utilisables)*
- CLRW	*'on fait une raz du registre w*
- IORWF tampon,0	*'ou logique entre le contenu de w et le contenu du registre nommé 'tampon'. Le résultat est stocké dans w puisque l'opérande d = 0. En fait c'est une copie du registre 'tampon' dans le registre w puisque auparavant, nous avons fais une raz de w (CLRW) -> 0 ou contenu registre tampon = registre tampon*

Instruction : **XORWF (eXclusive OR W with F)**

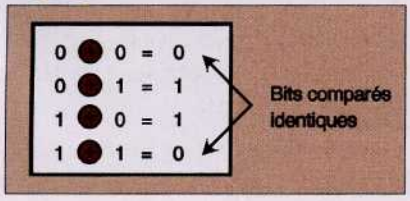
Rôle : Cette instruction fait un "ou exclusif " entre le contenu du registre de travail w et le contenu du registre spécifié d'adresse f. Selon l'opérande d, le résultat est soit stocké dans w (d=0), soit stocké dans le registre f (d=1).

Quand doit-on utiliser cette instruction ?

Souvent, pour comparer le contenu de deux registres, on utilise un ou exclusif. Si le résultat de l'opération vaut 0, alors les contenus des registres sont identiques.

Syntaxe : XORLW f,d
f = adresse du registre banalisé
d = direction pour le stockage du résultat

Le ou exclusif



(0 = résultat dans W, 1 = résultat dans le registre banalisé dont l'adresse est f)

Nombre de cycle(s) d'horloge nécessaire(s) : 1

Exemple d'utilisation :

tampon EQU 15	*'on déclare une variable appelée 'tampon' qui se trouve à l'adresse 15 en RAM (de 12 à 79 soit 68 variables utilisables)*
- XORWF tampon,0	*ou exclusif entre le contenu de w et le contenu du registre nommé 'tampon'. Le résultat est stocké dans w puisque l'opérande d = 0. Il suffit ensuite de tester le contenu du registre w, si celui-ci vaut zéro alors les deux registres (w et tampon) étaient identiques avant la comparaison.*

Instruction : **MOVF (Move f)**

Rôle : Cette instruction copie le contenu du registre spécifié d'adresse f sur lui même ou bien dans le registre w (selon l'opérande d, le résultat est soit stocké dans w (d=0), soit stocké dans le registre f (d=1)). Le fait de copier un registre sur lui-même peut paraître absurde mais cela permet entre autre de tester le flag z (zéro) du registre d'état et ainsi permettre de savoir si le contenu du registre vaut zéro.

Quand doit-on utiliser cette instruction ?

Une utilisation peut être envisagée lorsque l'on veut copier un registre banalisé vers un deuxième registre du même type. Dans un premier temps, on copie le registre dans le registre de travail w (MOVF registre, w) puis ensuite on transfert le contenu du registre de travail w vers le registre destination avec l'instruction MOVWF (que l'on verra plus loin).

Syntaxe : MOVF f,d
f = adresse du registre banalisé
d = direction pour le stockage du résultat (0 = résultat dans W, 1 = résultat dans le registre banalisé dont l'adresse est f)

Nombre de cycle(s) d'horloge nécessaire(s) : 1

Exemple d'utilisation :

Compteur EQU 15	*'on déclare une variable appelée 'compteur' qui se trouve à l'adresse 15 en RAM (de 12 à 79 soit 68 variables utilisables)*
recharge EQU 16	*'on déclare une variable appelée 'recharge' qui se trouve à l'adresse 16 en RAM (de 12 à 79 soit 68 variables utilisables)*
#define w, 0	*'Pour le compilateur, la lettre w vaudra 0 dans le reste du programme*
- MOVF recharge, w	*'on copie le contenu du registre recharge dans le registre de travail w*
- MOVWF Compteur	*'on transfert le contenu de w (c'est le contenu du registre recharge) dans la variable (ou registre banalisé) nommée Compteur*

Instruction : **MOVWF (Move w to f)**

Rôle : Cette instruction copie le contenu du registre de travail w vers le registre spécifié d'adresse f.

Quand doit-on utiliser cette instruction ?

Cette instruction sera souvent utilisée. C'est elle qui permettra de charger une valeur dans un registre. Encore une fois, nous voyons que le registre de travail w a une place importante dans le PIC.

Syntaxe : MOVWF f
f = adresse du registre banalisé

Nombre de cycle(s) d'horloge nécessaire(s) : 1

Exemple d'utilisation : Voir exemple précédent (instruction MOVF)

Instruction : **NOP (No oPération)**

Rôle : Cette instruction ne réalise aucune opération.

Quand doit-on utiliser cette instruction ?

Cette instruction est particulièrement utilisée lorsque l'on veut introduire un retard dans un programme, tout en sachant que l'instruction NOP prend un cycle machine.

Syntaxe : NOP

Nombre de cycle(s) d'horloge nécessaire(s) : 1

Exemple d'utilisation :

Notre application est cadencée avec un quartz de 4 MHz. Il faut savoir qu'en interne le PIC divise la fréquence du quartz par 4 ce qui fait une horloge à 1 MHz, donc un cycle machine dure 1 / 1 10^{-6} soit 1 micro seconde (1 µs). Nous souhaitons avoir une temporisation (ou un retard) dans une boucle de programme de 5 µs. Pour ce faire, nous allons insérer 5 instructions de type NOP. Toutefois, pour des temporisations très longues, il est bien certain que l'on n'utilisera pas ce genre d'artifice. Il faudra réaliser une temporisation avec le décomptage d'un registre par exemple.

mavariable EQU 16	*'on déclare une variable appelée 'mavariable' qui se trouve à l'adresse 16 en RAM (de 12 à 79 soit 68 variables utilisables)*
- CLRW	*'on fait une raz du registre w*
- ADDLW 06	*'on additionne le contenu de w avec 6 donc w = 6*
NOP	*'on introduit ici un retard de 5 µs*
NOP NOP NOP NOP	
-- MOVWF mavariable	*'on transfert le contenu de w (ici c'est la valeur 6) dans la variable (ou registre banalisé) nommée mavariable*

Instruction : **RLF (Rotate Left f throught carry)**

Rôle : Cette instruction exécute une rotation à gauche du contenu du registre spécifié d'adresse f. Le bit situé le plus à gauche est transféré dans la carry et la valeur précédente de la carry est transférée dans le bit 0 du registre. Selon l'opérande d, le résultat est soit stocké dans w (d=0), soit stocké dans le registre f (d=1).

Quand doit-on utiliser cette instruction ?

Il y a de nombreuses applications pour cette instruction, allant par exemple d'un simple chenillard à une multiplication par 2.

Syntaxe : RLF f,d
f = adresse du registre banalisé
d = direction pour le stockage du résultat (0 = résultat dans W, 1 = résultat dans le registre banalisé dont l'adresse est f)

Nombre de cycle(s) d'horloge nécessaire(s) : 1

Exemple d'utilisation :

reg_tour EQU 15	*'on déclare une variable appelée 'reg_tour' qui se trouve à l'adresse 15 en RAM (de 12 à 79 soit 68 variables utilisables)*
#define f, 1	*'Pour le compilateur, la lettre f vaudra 1 dans le reste du programme*
- CLRW	*'on fait une raz du registre w*
- ADDLW 01	*'on additionne le contenu de w avec 1 donc w = 1*
- MOVWF reg_tour	*'on transfert le contenu de w (c'est le contenu du registre recharge) dans la variable (ou registre banalisé) nommée reg_tour*
- RLF reg_tour,f	*'On réalise une rotation à gauche du contenu du registre nommé 'reg_tour'. Si on considère que la carry, avant le RLF, était positionnée à 0, la valeur du registre 'reg_tour' au départ valait 1, après la rotation il vaut 2*

Instruction : **RRF (Rotate right f throught carry)**

Rôle : Cette instruction exécute une rotation à droite du contenu du registre spécifié d'adresse f. Le bit situé le plus à gauche est transféré dans la carry et la valeur précédente de la carry est transférée dans le bit 0 du registre. (selon l'opérande d, le résultat est soit stocké dans w (d=0), soit stocké dans le registre f (d=1).

Quand doit-on utiliser cette instruction ?

Il y a de nombreuses applications pour cette instruction, allant par exemple d'un simple chenillard à une division par 2.

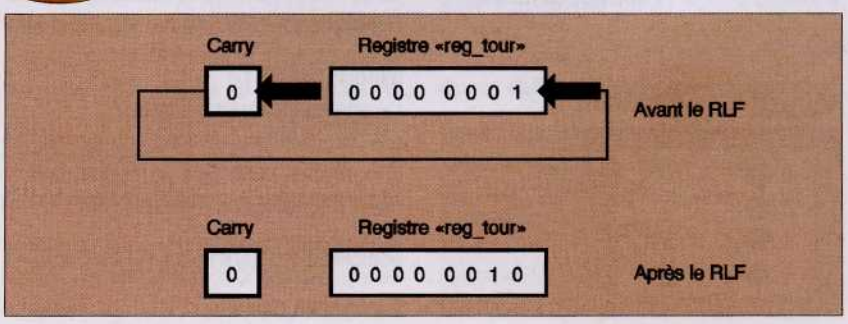
Syntaxe : RRF f,d
f = adresse du registre banalisé
d = direction pour le stockage du résultat (0 = résultat dans W, 1 = résultat dans le registre banalisé dont l'adresse est f)

Nombre de cycle(s) d'horloge nécessaire(s) : 1

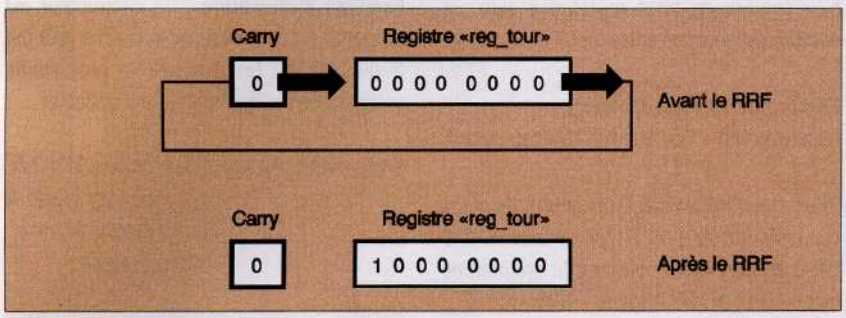
Exemple d'utilisation :

reg_tour EQU 15	*'on déclare une variable appelée 'reg_tour' qui se trouve à l'adresse 15 en RAM (de 12 à 79 soit 68 variables utilisables)*
#define f, 1	*'Pour le compilateur, la lettre f vaudra 1 dans le reste du programme*
- MOVLW 0xFF	*'on met le registre w à 255 ou FF en héxa (1111 1111)*
- ADDLW 01	*'on additionne le contenu de w avec 1 donc w = w+1 (la carry est positionnée et w=0)*
- MOVWF reg_tour	*'on transfert le contenu de w (c'est le contenu du registre recharge) dans la variable (ou registre banalisé) nommée reg_tour*

2 ***L'instruction RLF***



- RRF reg_tour,f	*'On réalise une rotation à droite du contenu du registre nommé 'reg_tour'. La valeur du registre 'reg_tour' au départ valait 0, après la rotation, il vaut 80 héxa (1000 0000)*



3 L'instruction RRF

Instruction : **SUBWF (SUBstract W from F)**

Rôle : Cette instruction exécute une soustraction entre le contenu du registre spécifié d'adresse f et le contenu du registre w , selon l'opérande d, le résultat est soit stocké dans w (d=0), soit stocké dans le registre f (d=1).

Quand doit-on utiliser cette instruction ?

On utilise cette instruction pour réaliser par exemple une soustraction conditionnelle, la valeur à soustraire est dynamique car est peut évoluer, c'est en fait le contenu d'un registre.

Syntaxe : SUBWF f,d
f= adresse du registre banalisé
d= direction pour le stockage du résultat (0 = résultat dans W , 1 = résultat dans le registre banalisé dont l'adresse est f)

Nombre de cycle(s) d'horloge nécessaire(s) : 1

Exemple d'utilisation :

mavariable EQU 16	*'on déclare une variable appelée 'mavariable' qui se trouve à l'adresse 16 en RAM (de 12 à 79 soit 68 variables utilisables)*
- CLRW	*'on fait une raz du registre w*
- ADDLW 06	*'on additionne le contenu de w avec 6 donc w=6*
- SUBWFmavariable,0	*'on soustrait le contenu du registre 'mavariable' avec le contenu de w, le résultat est stocké dans w car l'opérande d = 0*

Instruction : **SWAPF (SWAP nibbles in F)**

Rôle : Cette instruction permet d'échanger les quatre bits de poids faible par les quatre bits de poids fort du contenu du registre spécifié d'adresse f. Selon l'opérande d, le résultat est soit stocké dans w (d=0), soit stocké dans le registre f (d=1).

Quand doit-on utiliser cette instruction ?

On utilise souvent cette instruction lorsque l'on travail sous interruption afin de sauvegarder le registre STATUS (registre d'état) car l'instruction SWAPF à la particularité de n'affecter aucun flag du registre d'état.

Syntaxe : SWAPF f,d
f = adresse du registre banalisé
d = direction pour le stockage du résultat (0 = résultat dans W, 1 = résultat dans le registre banalisé dont l'adresse est f)

Nombre de cycle(s) d'horloge nécessaire(s) : 1

Exemple d'utilisation :

- CLRW	*'on fait une raz du registre w*
- SWAPF STATUS,0	*'on sauvegarde le registre STATUS (registres d'état qui contient tous les flags) dans le registre de travail w puisque l'opérande d = 0*

4 - Instructions sur un bit d'un registre

Instruction : **BCF (Bit Clear f)**

Rôle : Cette instruction permet de faire une RAZ du bit indiqué (0 à 7) *du registre spécifié* d'adresse f.

Quand doit-on utiliser cette instruction ?

Cette instruction est très utilisée. Elle permet de travailler rapidement sur un bit d'un registre sans modifier le reste du registre.

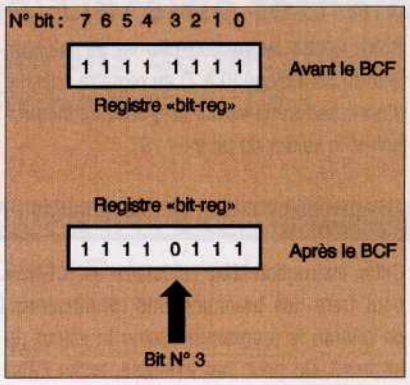
Syntaxe : BCF f,b
f = adresse du registre banalisé
b = numéro du bit à remettre à zéro, ce nombre est compris entre 0 et 7.

Nombre de cycle(s) d'horloge nécessaire(s) : 1

Exemple d'utilisation :

bit-reg EQU 15	*'on déclare une variable appelée bit-reg' qui se trouve à l'adresse 15 en RAM (de 12 à 79 soit 68 variables utilisables)*
- MOVLW 0xFF	*'on met le registre w à 255 ou FF en héxa (1111 1111)*
- MOVWF bit-reg	*'on transfert le contenu de w (FF ou 255) dans la variable (ou registre banalisé) nommée bit-reg*
- BCF bit-reg,3	*'On fait une raz du 3e bit du registre nommé 'bit-reg'. La valeur du registre 'bit-reg' au départ valait 1111 1111, après le BCF, il vaut 1111 0111 (3e bit à zéro)*

4 L'instruction BCF



Instruction : **BSF (Bit Set f)**

Rôle : Cette instruction permet de faire une mise à "1" du bit indiqué (0 à 7) du registre spécifié d'adresse f.

Quand doit-on utiliser cette instruction ?

Cette instruction est très utilisée. Elle permet de forcer rapidement à "1" la valeur d'un bit d'un registre sans modifier le reste du registre.

Syntaxe : BSF f,b
f = adresse du registre banalisé
b = numéro du bit à mettre à "1", ce nombre est compris entre 0 et 7.

Nombre de cycle(s) d'horloge nécessaire(s) : 1

Exemple d'utilisation :

bit-reg EQU 15	*'on déclare une variable que l'on appelle bit-reg' qui se trouve à l'adresse 15 en RAM (de 12 à 79 soit 68 variables utilisables)*
- MOVLW 0x00	*'on met le registre w à 0 (0000 0000)*
- MOVWF bit-reg	*'on transfert le contenu de w (00) dans la variable (ou registre banalisé) nommée bit-reg*
- BSF bit-reg,7	*'On fait une mise à "1" du 7e bit du registre nommé 'bit-reg'. La valeur du registre 'bit-reg' au départ valait 0000 0000. Après le BSF, il vaut 1000 0000 (7e bit à un)*

Instruction : **BTFSC (Bit Test F Skip if Clear)**

Rôle : Cette instruction permet de faire un test du bit indiqué (0 à 7) du registre spécifié d'adresse f et de sauter la prochaine instruction si la valeur du bit vaut "0".

Quand doit-on utiliser cette instruction ?

Cette instruction très puissante est utilisée pour faire des branchements conditionnels, on oriente le programme selon la valeur du bit testé. On peut, par exemple, tester l'état d'un bit d'un port configuré en entrée sur lequel est connecté un interrupteur. Tant que cet interrupteur est actionné, le programme ne démarre pas (interrupteur type "coup de poing" par exemple).

Syntaxe : BTFSC f,b
f = adresse du registre banalisé
b = numéro du bit à remettre à zéro, ce nombre est compris entre 0 et 7.

Nombre de cycle(s) d'horloge nécessaire(s) : 1 ou 2 (selon résultat du test)

Exemple d'utilisation : Un interrupteur est connecté sur le port A sur la broche RA0 (bit 0 du port A). On démarre le programme uniquement si l'interrupteur est ouvert .

#define inter, 0	*'Pour le compilateur le mot inter représente la valeur 0 dans le reste du programme*
debut	
- BTFSC PORTA,inter	*'on teste le bit "0" du port A et si celui-ci vaut "0" alors on saute la prochaine instruction qui reboucle au début du programme*
- goto debut	*'on retourne à l'étiquette 'debut' tant que la valeur de l'inter vaut "1"*
instruction	*'suite du programme*
instruction	

Instruction : **BTFSS (Bit Test F Skip if Set)**

Rôle : Cette instruction permet de faire un test du bit indiqué (0 à 7) du registre spécifié d'adresse f et de sauter la prochaine instruction si la valeur du bit vaut "1".

Quand doit-on utiliser cette instruction ?

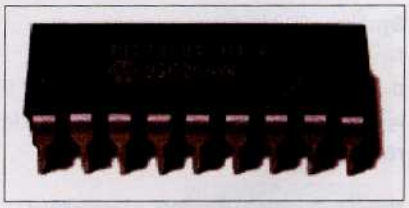
Cette instruction très puissante est utilisée pour faire des branchements conditionnels. On oriente le programme selon la valeur du bit testé. On peut par exemple tester l'état d'un bit d'un port configuré en entrée sur lequel est connecté un interrupteur. Tant que cet interrupteur n'est pas actionné, le programme ne démarre pas.

Syntaxe : BTFSC f,b
f = adresse du registre banalisé
b = numéro du bit à remettre à zéro, ce nombre est compris entre 0 et 7.

Nombre de cycle(s) d'horloge nécessaire(s) : 1 ou 2 (selon résultat du test)

Exemple d'utilisation : Un interrupteur est connecté sur le port A sur la broche RA0 (bit 0 du port A). On démarre le programme uniquement si l'interrupteur est actionné.

#define inter, 0	*'Pour le compilateur le mot inter représente la valeur 0 dans le reste du programme*
debut	
- BTFSS PORTA,inter	*'on teste le bit "0" du port A et si celui-ci vaut "1" alors on saute la prochaine instruction qui reboucle au début du programme*
- goto debut	*'on retourne à l'étiquette 'debut' tant que la valeur de l'inter vaut "0"*
instruction	*'suite du programme*
instruction	



Pour conclure ce chapitre

Avec cette cinquième partie, nous avons terminé de détailler les instructions des PIC de la série 16Fxx et 16Fxxx.

Dans le prochain numéro, nous détaillerons et réaliserons notre premier programme en utilisant le logiciel MPLAB.

P. MAYEUX
http://perso.libertysurf.fr/p.may