Christian Tavernier

MICROCONTRÔLEURS PIC 18

Description et mise en œuvre

DUNOD

- *Bit 7 ou bit IDLEN pour IDLe ENable*

Ce bit permet de déterminer dans quel mode va se trouver le processeur suite à l'exécution d'une instruction SLEEP. Lorsqu'il est à 0, le processeur entre réellement en mode SLEEP « complet » alors qu'il passe seulement en mode IDLE dans le cas contraire (voir tableau 2.4).

2.2 Le reset

2.2.1 Principes généraux

Les microcontrôleurs PIC 18 sont particulièrement bien fournis au plan du reset. Ils disposent en effet de huit sources potentielles de reset distinctes avec :

- un reset à la mise sous tension du circuit appelée POR pour *Power On Reset* ;
- un reset par action sur la patte $\overline{\text{MCLR}}$ alors que le circuit est en mode normal ;
- un reset par action sur la patte $\overline{\text{MCLR}}$ alors que le circuit est en mode sommeil ;
- un reset par débordement du timer chien de garde (WDT pour *Watch Dog Timer*) ;
- un reset par détection d'une chute anormale de la tension d'alimentation, même temporaire, appelé BOR (pour *Brown Out Reset*) ;
- un reset suite à l'exécution de l'instruction de même nom (RESET) ;
- un reset suite à une saturation de la pile (pile pleine) ;
- un reset suite à un débordement inférieur de la pile (tentative de retirer une donnée de la pile alors qu'elle est vide).

Le comportement du circuit et l'état des registres affectés par un reset sont différents selon celle de ces situations qui se produit ; situation qui peut être identifiée au moyen d'un registre spécial appelé RCON dont les 5 bits de poids faibles sont dévolus à cet effet. Ce registre est en effet organisé de la façon suivante :

Accès E : Écriture L : Lecture	L/E	L/E	L	L/E	L	L	L/E	L/E
RCON	IPEN	SBOREN	—	$\overline{\text{RI}}$	$\overline{\text{TO}}$	$\overline{\text{PD}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$
État à la mise sous tension	0	1	0	1	1	1	0	0

- *Bit 0 ou bit* $\overline{\text{BOR}}$ *pour Brown Out Reset*

Ce bit indique, lorsqu'il est à 0, qu'un reset dû à une détection de chute de la tension d'alimentation vient de se produire. Ce bit est un peu particulier en ce sens que son état est inconnu suite à une mise sous tension du circuit. Il doit donc être positionné à 1 par programme. De même, après son passage à 0 suite à la détection d'une baisse de tension d'alimentation, il faut le remettre à 1 par programme.

- *Bit 1 ou bit $\overline{\text{POR}}$ pour Power On Reset*

Ce bit indique, lorsqu'il est à 0, qu'un reset suite à une mise sous tension du circuit vient de se produire. Comme le bit 0, il faut le remettre à 1 par programme après son passage à 0 suite à la détection d'un reset de mise sous tension.

- *Bit 2 ou bit $\overline{\text{PD}}$ pour Power Down*

Ce bit est mis à 0 lors de l'exécution d'une instruction SLEEP. Il est mis à 1 lors d'une mise sous tension ou lors de l'exécution d'une instruction CLRWDT.

- *Bit 3 ou bit $\overline{\text{TO}}$ pour watchdog Timer Out*

Ce bit est mis à 0 lorsque le timer chien de garde déborde. Il est mis à 1 lors d'une mise sous tension ou lors de l'exécution d'une instruction CLRWDT ou SLEEP.

- *Bit 4 ou bit $\overline{\text{RI}}$ pour Reset Instruction*

Ce bit est mis à 0 lors de l'exécution d'une instruction RESET ayant eu comme conséquence de provoquer un reset du circuit. Comme le bit 0, il faut le remettre à 1 par programme après son passage à 0 suite à l'exécution d'une instruction RESET.

- *Bit 5*

Bit non utilisé et lu comme étant à 0.

- *Bit 6 ou bit SBOREN pour Software BOR ENable*

Ce bit permet de contrôler par logiciel la validation (SBOREN = 1) ou non (SBOREN = 0) du reset en cas de baisse de l'alimentation ou reset BOR. Pour que ce bit puisse fonctionner, il faut que les bits BOREN0 et BOREN1 du registre de configuration du circuit aient au préalable été positionnés respectivement à 1 et 0 lors de la programmation du circuit (voir § 2.2.2).

- *Bit 7 ou bit IPEN pour Interrupt Priority ENable*

Lorsque ce bit est mis à 1, la gestion de priorité des interruptions est assurée alors qu'elle ne l'est pas dans le cas contraire afin d'assurer la compatibilité avec les PIC 16 par exemple.

Dans tous les cas de reset, sauf réveil du mode sommeil par le timer chien de garde ou l'arrivée d'une interruption, le compteur ordinal ou PC est mis à 0000. La première instruction exécutable de votre programme doit donc se trouver à cette adresse. En cas de réveil d'un mode sommeil par le timer chien de garde, le PC est simplement augmenté de deux unités (nous verrons pourquoi deux lors de la présentation de l'organisation mémoire) pour passer à l'instruction qui suit l'instruction SLEEP.

Les contenus des registres de contrôle, d'état ou de données du microcontrôleur sont diversement affectés par les différentes sources de reset possibles mais, en règle générale, l'état de la majorité des registres internes du circuit doit être considéré comme inconnu suite à un reset. En outre, il n'est généralement pas affecté par les resets autres que celui causé par une mise sous tension. Un tableau figure dans toutes les

fiches techniques des PIC 18 pour préciser, registre par registre et bit par bit, l'é des différents resets sur tous les registres du circuit concerné. En ce qui r concerne, nous indiquons, tout au long de l'ouvrage lors de la description des bits registres internes, leur état suite à un reset de mise sous tension.

2.2.2 Circuiterie de reset externe

Tous les circuits de la famille PIC 18 adoptent la même démarche, à savoir la prése d'une seule patte de reset baptisée $\overline{\text{MCLR}}$. Cette patte de reset dispose de quelq particularités pour qui est habitué aux microcontrôleurs classiques. En effet, circuits intègrent en interne une circuiterie de reset automatique à la mise s tension qui, si cette procédure s'avère suffisante (pas de besoin de reset exte manuel par exemple) et si la vitesse de croissance de la tension d'alimentation est as élevée (typiquement supérieure à 0,05 V/ms), se suffit à elle-même.

Il y a quelques années, Microchip autorisait donc dans ce cas le raccordement dir de cette patte $\overline{\text{MCLR}}$ à l'alimentation comme indiqué figure 2.5.

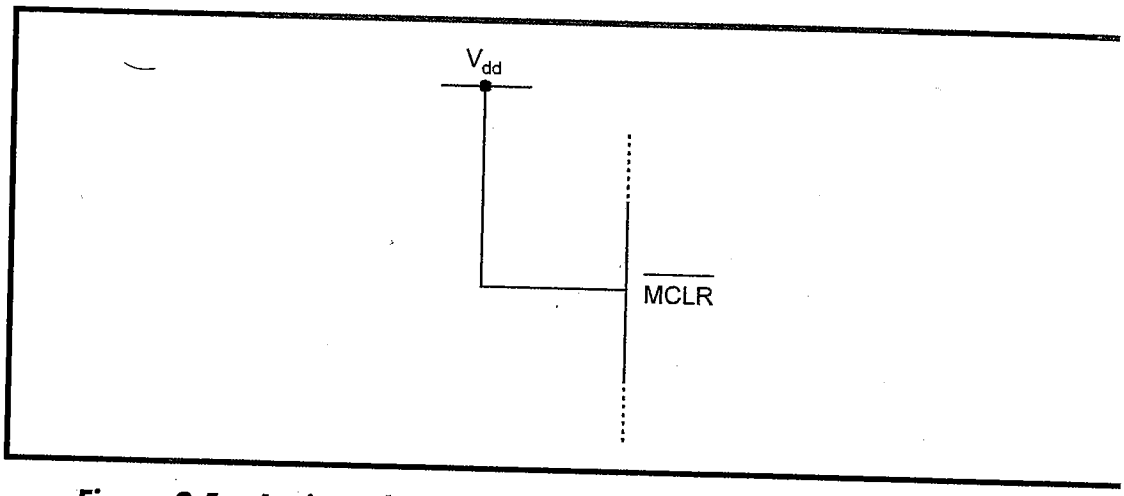

Figure 2.5 – Ancien schéma de reset automatique à la mise sous tension, déconseillé aujourd'hui.

Depuis quelque temps, suite à une modification de la protection contre les décha ges électrostatiques (ESD) intégrée dans les circuits, ce mode de connexion e déconseillé par Microchip car il peut conduire à des resets intempestifs, voire même une destruction de l'entrée $\overline{\text{MCLR}}$ et/ou de l'intégralité du circuit en cas justemer de décharge électrostatique. Le schéma de la figure 2.6 est donc le seul à utiliser dar toutes les situations où l'on souhaite bénéficier d'une procédure de reset automatiqu à la mise sous tension sans procédure de reset manuel au moyen d'un poussoir.

Si la vitesse de croissance minimum de la tension d'alimentation n'est pas atteinte, vous avez besoin d'une commande de reset externe ou bien encore si vous utilisez u quartz de fréquence relativement basse, dont le temps de mise en oscillation est don plus important que ce que peuvent supporter les différents timers d'attente internes la circuiterie de la figure 2.7 doit alors être mise en œuvre.

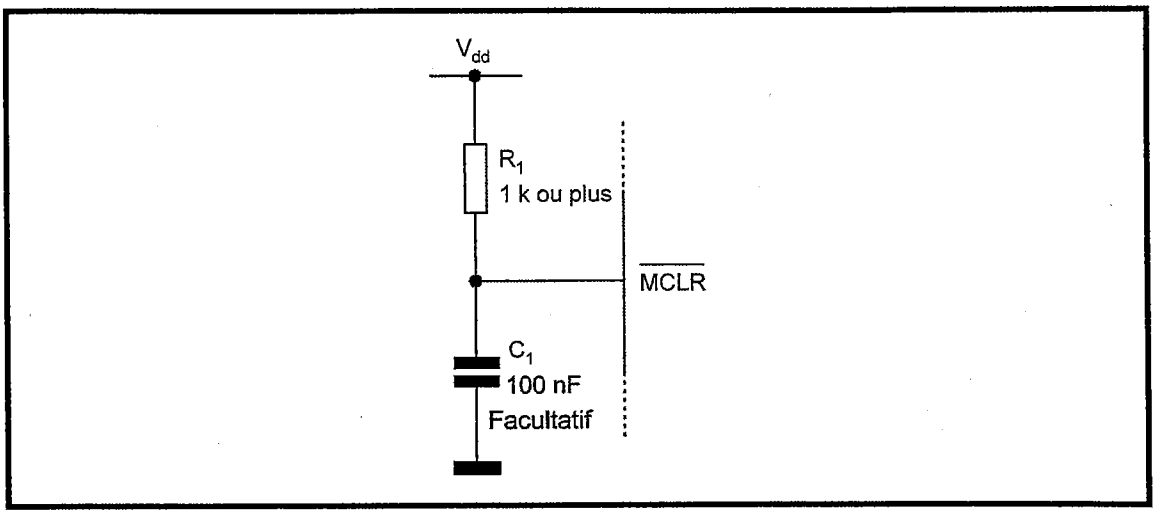

Figure 2.6 – Schéma de reset automatique à la mise sous tension préconisé par Microchip afin d'éviter un risque de destruction du circuit.

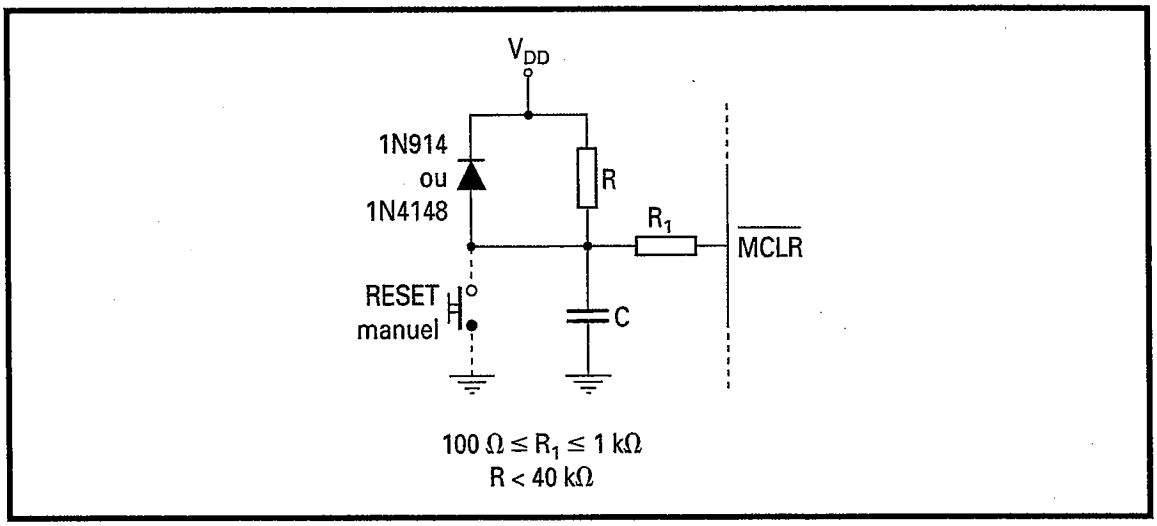

Figure 2.7 – Schéma à utiliser pour bénéficier d'une commande de reset manuelle.

On retrouve ici un schéma connu des utilisateurs de microcontrôleurs classiques ! Notez la présence de la résistance R1 dont la valeur varie de 100 Ω à 1 kΩ. Elle a pour but de protéger le circuit contre un courant de décharge de C dans la patte $\overline{\text{MCLR}}$ en cas de décharge électrostatique à proximité.

2.2.3 Le reset par détection de chute de tension d'alimentation ou BOR

Si la tension d'alimentation est susceptible de varier dans des proportions telles que le bon fonctionnement du circuit peut être compromis, il est prudent de prévoir une circuiterie de reset capable de déclencher ce dernier, si justement l'alimentation descend en dessous d'un certain seuil. Tous les PIC 18 disposent à cet effet de la fonction BOR (*Brown Out Reset*) et aucun composant externe supplémentaire n'est nécessaire sous réserve de configurer correctement cette ressource.

Notez cependant qu'il ne faut pas confondre le reset automatique à la mise sous tension (POR) avec le reset en cas de baisse anormale de la tension d'alimentation (BOR). En effet, la circuiterie de reset automatique à la mise sous tension est incapable de générer un signal de reset si la tension d'alimentation baisse de sa valeur nominale à une valeur inférieure à la valeur de fonctionnement normale. La circuiterie de reset automatique à la mise sous tension impose en effet que la tension appliquée sur la patte d'alimentation positive du circuit (V_{DD}) devienne quasi nulle pendant au moins 100 µs avant de produire un signal de reset interne.

La circuiterie BOR quant à elle est capable de réagir à une baisse anormale de cette tension sans que cette dernière ait besoin de devenir nulle et sans conditions de durée particulières.

Elle est sous le contrôle de cinq bits distincts : les bits BORV1, BORV0, BOREN1 et BOREN0 qui sont contenus dans les registres de configuration du circuit et ne peuvent donc être modifiés que lors de la programmation du circuit, et le bit SBOREN, contenu dans le registre RCON présenté précédemment, qui peut donc être modifié pendant l'exécution d'un programme.

Les bits BORV1 et BORV0 permettent de fixer la tension de seuil à partir de laquelle un reset de type BOR pourra avoir lieu. Cette information étant dépendante du circuit, nous vous renvoyons à la lecture de la partie « Caractéristiques électriques » de la fiche technique du circuit concerné pour connaître la valeur à donner à ces bits en fonction du seuil désiré.

Pour ce qui est des bits BOREN1 et BOREN0 ils peuvent prendre les différents états présentés tableau 2.7.

Tableau 2.7 – Les différents modes de fonctionnement du reset de type BOR.

BOREN1	BOREN0	SBOREN	Fonctionnement du reset BOR
0	0	Sans effet	BOR inactif
0	1	Actif	BOR activé. Validation contrôlée par SBOREN
1	0	Sans effet	BOR activé sauf en mode SLEEP
1	1	Sans effet	BOR activé dans tous les cas

Comme on peut le constater à la lecture de ce tableau, il est possible de valider ou non de façon « définitive », c'est-à-dire lors de la programmation du circuit, le reset de type BOR ou, ce qui est plus souple et donc vivement conseillé, de permettre le contrôle de ce reset de type BOR au moyen du bit SBOREN du registre RCON.

2.2.4 Schéma interne de la circuiterie de reset

La figure 2.8 vous propose de découvrir une partie du schéma interne de la circuiterie de reset des microcontrôleurs PIC 18 ce qui va nous permettre de comprendre comment se déroulent réellement les différentes séquences de reset.

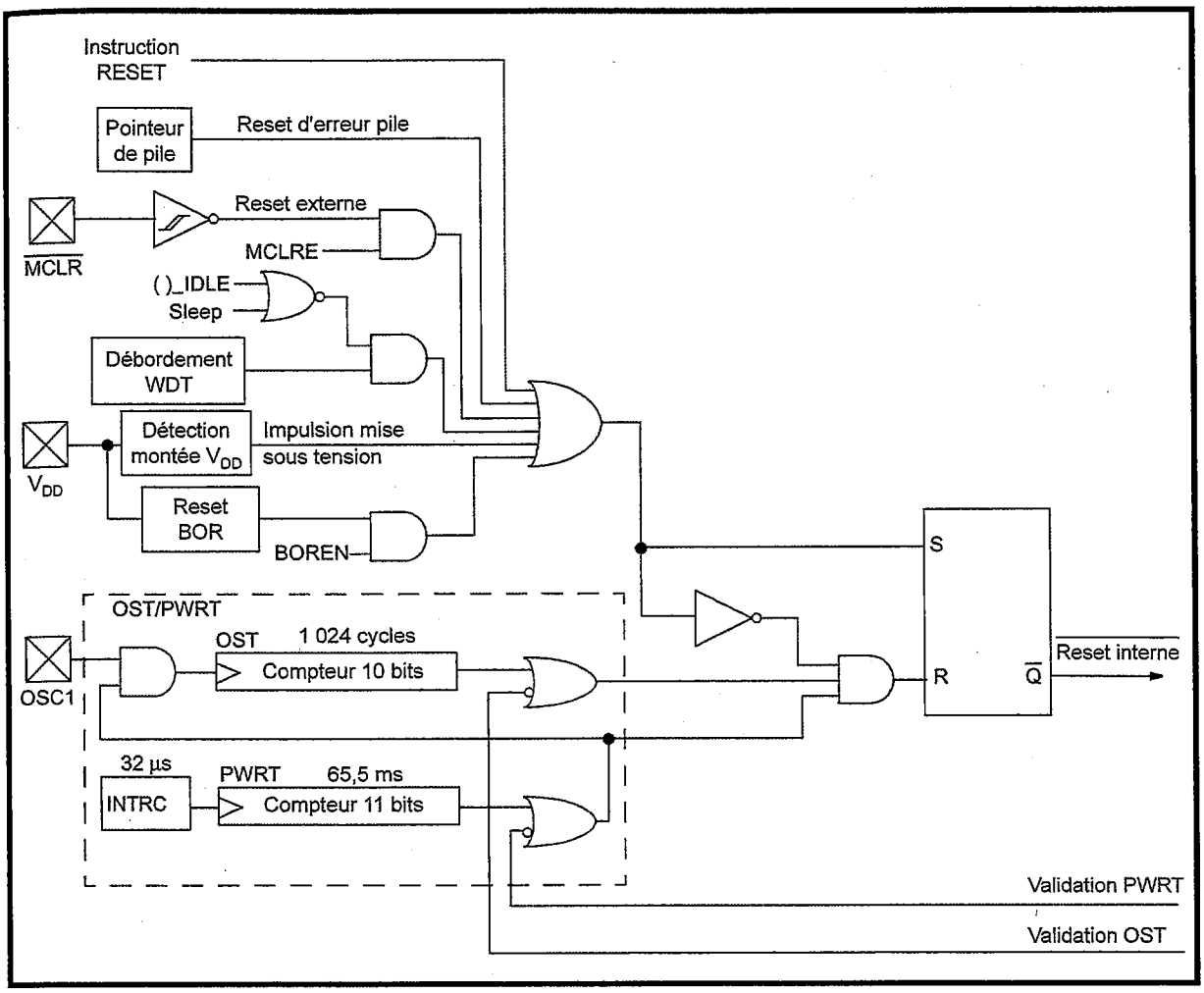

Figure 2.8 – Schéma équivalent de la circuiterie de reset interne des PIC 18.

Tout d'abord, notez sur cette figure que la seule information importante pour l'unité centrale est celle située en bas à droite, appelée reset interne. Elle est générée par la combinaison d'un certain nombre de phénomènes correspondant aux diverses sources possibles de reset que nous venons d'énumérer.

En fait, les resets logiques, c'est-à-dire ceux provenant d'une source interne telle que le timer chien de garde (WDT) ou bien encore les débordements de pile ne posent aucun problème. Ce sont en effet des signaux logiques « normaux » qui sont appliqués à la circuiterie de reset pour générer quasi immédiatement un reset interne. La partie la plus délicate est celle qui concerne le reset à la mise sous tension car de nombreux paramètres divers doivent être pris en compte.

En effet, lors de la mise sous tension du circuit, d'une part l'oscillateur d'horloge ne démarre pas immédiatement, d'autre part la tension d'alimentation n'atteint pas immédiatement la valeur nominale nécessaire au bon fonctionnement du microcontrôleur. Il est donc indispensable de s'assurer que le reset interne n'est généré qu'après que cet oscillateur a effectivement démarré et que lorsque la tension d'alimentation a atteint une valeur suffisante.

Pour cela, les PIC mettent en œuvre deux timers distincts. Le premier, appelé PWRT pour *PoWeR up Timer*, repose sur un oscillateur RC interne, visible en bas à gauche de la figure 2.8. Grâce au compteur à 11 bits qui le suit, il impose d'attendre un temps baptisé T_{PWRT} dans toutes les fiches techniques des PIC 18 (qui vaut approximativement 70 ms) suite à une mise sous tension avant de pouvoir générer un reset interne. La tension d'alimentation est alors supposée avoir atteint un niveau stable et correct. Si tel n'est pas le cas, le reset automatique à la mise sous tension doit être considéré comme inutilisable et il faut alors faire appel au circuit de la figure 2.7 en s'assurant que sa constante de temps est supérieure à celle d'établissement de la tension d'alimentation. Cette situation reste toutefois exceptionnelle avec des circuits d'alimentation bien conçus.

Pour ce qui est de l'oscillateur d'horloge, un autre compteur 10 bits, validé par le bit OST, permet là encore d'attendre 1 024 cycles d'horloge avant que ne soit généré le reset interne, ce qui permet à l'oscillateur d'horloge de se stabiliser.

La figure 2.9 montre ainsi le chronogramme type d'une séquence de reset à la mise sous tension avec mise en évidence de l'action des différents timers. Comme le montrent bien les deux premiers tracés de ce chronogramme, le microcontrôleur fonctionne ici avec sa patte $\overline{MCLR}$ reliée à l'alimentation positive ce qui, comme nous l'avons vu en figure 2.6, est le mode de câblage normal à utiliser pour exploiter la fonction de reset automatique à la mise sous tension.

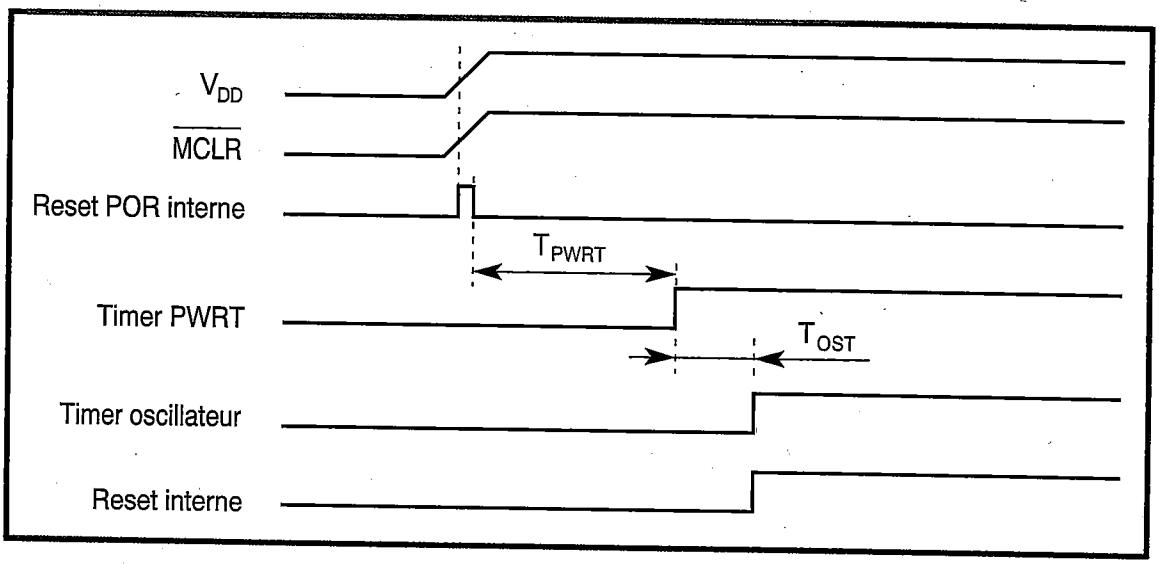

Figure 2.9 – Chronogramme d'un reset suite à une mise sous tension du circuit.

Les autres sources de reset visibles sur la figure 2.8 sont plus classiques et ne font appel qu'à des fonctions logiques combinatoires destinées à assurer leur prise en compte normale. Ce sont :

- la patte de reset externe $\overline{MCLR}$;
- l'information en provenance du timer chien de garde (WDT) ;

– les informations d'erreurs de pile ;
– l'information de chute de tension d'alimentation BOR ;
– l'instruction RESET.

Compte tenu du schéma logique de l'ensemble de cette circuiterie de reset, les délais entre la situation devant provoquer le reset et la génération effective du reset interne sont indiqués tableau 2.8.

Tableau 2.8 – Délais entre la cause du reset et le reset interne dans différentes situations.

Oscillateur	Mise sous tension ou chute de tension		Réveil
	$\overline{PWRTEN} = 0$	$\overline{PWRTEN} = 1$	
HS, XT	$T_{PWRT} + 1\,024\ T_{OSC}$	$1\,024\ T_{OSC}$	$1\,024\ T_{OSC}$
HSPLL, XTPLL	$T_{PWRT} + 1\,024\ T_{OSC} + 2$ ms	$1\,024\ T_{OSC} + 2$ ms	$1\,024\ T_{OSC} + 2$ ms
EC, ECIO	T_{PWRT}	–	–
ECPLL, ECPIO	$T_{PWRT} + 2$ ms	2 ms	2 ms
INTIO, INTCKO	T_{PWRT}	–	–
INTHS, INTXT	$T_{PWRT} + 1\,024\ T_{OSC}$	$1\,024\ T_{OSC}$	$1\,024\ T_{OSC}$

Note : 2 ms est le temps typique de verrouillage de la boucle PLL.

Dans ce tableau, T_{PWRT} est évidemment le délai généré par le timer PWRT ou timer de mise sous tension dont la valeur typique est de l'ordre de 70 ms à quelques pour cent près selon les versions de circuits.

2.3 Le timer chien de garde

Le timer chien de garde, ou WDT pour *Watch Dog Timer*, est une fonction utilisée pour se prémunir des errements de certains logiciels ou des blocages pouvant se produire dans certaines situations. En effet, il est constitué d'un compteur, incrémenté au rythme d'une horloge qui fonctionne quoi qu'il advienne et, lorsque ce compteur déborde, un reset est généré. Le logiciel qui s'exécute sur un circuit muni d'une telle ressource doit donc régulièrement venir remettre à zéro ce compteur afin d'éviter ce débordement fatal. Si le logiciel « se plante » ou si, pour une raison quelconque, il est empêché de venir effectuer cette mise à zéro, par exemple parce qu'il reste indéfiniment en attente dans une boucle, le timer chien de garde se charge de mettre fin à cette situation indésirable grâce au reset provoqué par son débordement.

2.3.1 Le timer chien de garde des PIC 18

Le timer chien de garde dont sont munis tous les microcontrôleurs PIC 18 respecte à la lettre ce principe général. Il est muni de son propre oscillateur autonome qui n'est autre que l'oscillateur d'horloge RC interne et ne nécessite donc aucun composant externe. De plus, il continue de fonctionner même lorsque l'horloge système du PIC est arrêtée lors d'une instruction SLEEP de mise en sommeil par exemple.

Comme tous ses homologues, ce timer compte en permanence et le fait de déborder lui fait générer un reset du microcontrôleur. Si le PIC se trouve en mode sommeil lorsque ce débordement se produit, ceci a pour effet de le faire sortir de ce mode.

L'utilisation du timer chien de garde n'étant aucunement obligatoire, vous pouvez le valider ou non au moyen du bit WDTEN du registre CONFIG2H, comme nous le verrons au chapitre 3, étant entendu que ce bit a en fait une double fonction.

Si vous positionnez à 0 le bit WDTEN de ce registre, cela a pour effet de ne pas valider le timer chien de garde mais, lors de l'exécution du programme de l'application, ce dernier pourra agir sur le bit SWDTEN du registre WDTCON que nous décrirons dans un instant, pour valider à la demande ce timer.

Par contre, le fait de valider le timer chien de garde en positionnant à 1 le bit WDTEN du registre CONFIG2H valide définitivement le timer chien de garde et rend le bit SWDTEN du registre WDTCON inopérant.

2.3.2 Schéma synoptique du timer chien de garde

Le timer chien de garde des PIC 18 est une entité autonome contrairement à ce dont vous aviez l'habitude si vous venez du monde des PIC 10, 12 ou 16 où il était plus ou moins partagé, selon les versions de circuits, avec d'autres ressources internes. Son schéma synoptique vous est présenté figure 2.10.

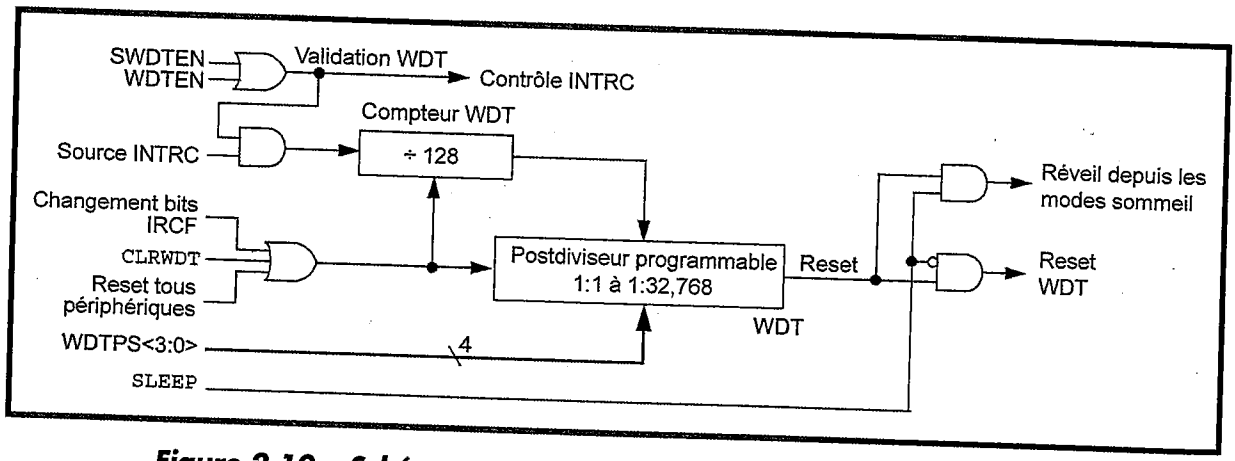

Figure 2.10 – Schéma synoptique du timer chien de garde ou WDT.

L'horloge du timer chien de garde provient de l'horloge RC interne du PIC qui fonctionne avec une période typique de 4 ms. Elle passe par un premier compteur par 128 avant d'arriver sur un diviseur programmable qui permet de fixer la vitesse de

gistres PIE
lecture ou
bit PSPIF

.ogrammes
mode PSP.
sur les bits

| Q4 |

parallèle.

| Q4 |

parallèle.

CHAPITRE 6

LES TIMERS

Dans tout microcontrôleur qui se respecte, ce que l'on nomme timer est en fait un compteur prépositionnable, qui compte ou décompte au rythme d'une horloge, interne ou externe, et les trois à cinq timers que contiennent les PIC 18 ne font pas exception à cette règle.

Ils ne fonctionnent cependant pas tous de la même façon car ils peuvent être associés, selon le cas, à des modules fonctionnels appelés CCP et PWM, présentés au chapitre 7, qui permettent de réaliser de la capture de données en entrée, de la comparaison de données en sortie et de la modulation de largeur d'impulsions.

De ce fait, nous allons les étudier indépendamment les uns des autres, en commençant bien entendu par celui qui est tout à la fois le plus simple d'emploi et le plus fréquemment utilisé, à savoir le timer 0.

6.1 Le timer 0

Le timer 0 est le plus simple des trois car c'est un timer autonome, qu'il n'est pas prévu d'associer à d'autres ressources internes dans le but de réaliser des fonctions plus complexes. Pour ceux d'entre vous qui viennent du monde des PIC 10, 12 et 16, c'est le digne successeur du timer 0 qui équipait déjà ces circuits, même si quelques petites améliorations lui ont été apportées.

Ce timer est capable de fonctionner en mode 8 bits, auquel cas il ne peut compter que de 00 à FF, ou en mode 16 bits, auquel cas il sait compter de 0000 à FFFF. Son schéma interne peut donc revêtir l'un des deux aspects visibles figures 6.1 et 6.2.

Si l'on part du registre TMR0L qui est le registre de comptage du timer en mode 8 bits ou les 8 bits de poids faibles de ce même registre en mode 16 bits, on constate qu'il peut recevoir son horloge directement ou après passage de celle-ci par un prédiviseur, la sélection étant faite par un bit appelé PSA contenu dans le registre T0CON que nous décrirons dans un instant.

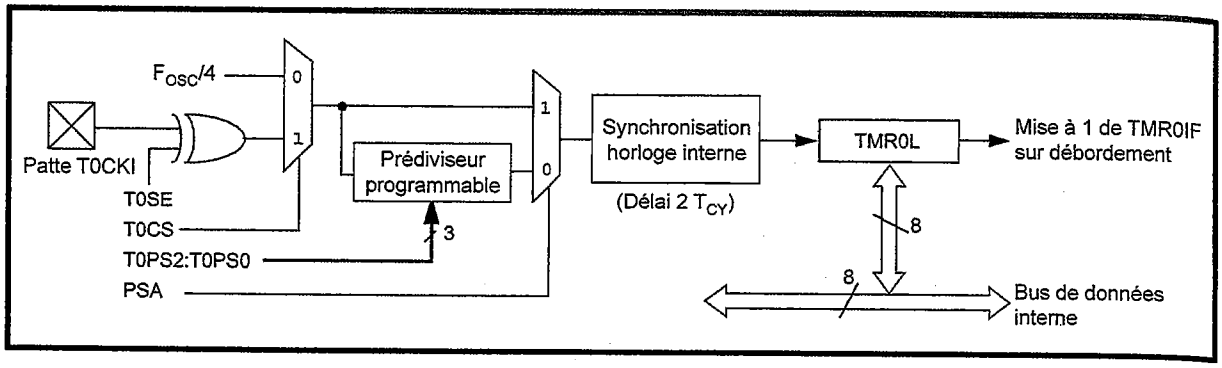

Figure 6.1 – Schéma synoptique du timer 0 en mode 8 bits.

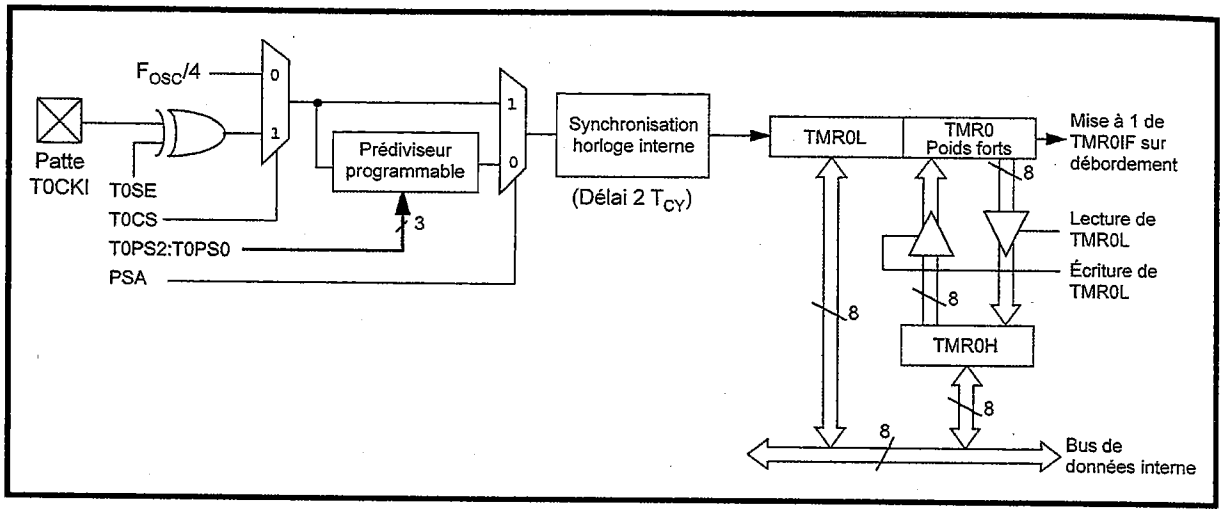

Figure 6.2 – Schéma synoptique du timer 0 en mode 16 bits.

Si le prédiviseur est sélectionné, son taux de division est programmé au moyen de trois bits ayant pour noms T0PS0, T0PS1 et T0PS2 également contenus dans le registre T0CON.

Dans tous les cas, le signal entrant dans ce prédiviseur peut à nouveau provenir de deux sources selon l'état du bit de sélection T0CS. Soit il provient de l'oscillateur d'horloge après division de fréquence par quatre ; il est donc bien à la fréquence de l'horloge système ou horloge instruction. Soit il provient de l'extérieur *via* la patte T0CKI. Le OU EXCLUSIF commandé par le bit T0SE permet quant à lui de sélectionner si l'incrémentation du registre TMR0L aura lieu sur un front montant ou descendant du signal appliqué à T0CKI.

Ce principe reste vrai que le timer fonctionne en mode 8 ou 16 bits. Dans le premier cas, le registre de comptage se résume au seul registre TMR0L qui peut être lu et écrit à tout instant. Notez cependant que, suite à une écriture dans ce registre, l'incrémentation du timer est suspendue pendant les deux cycles d'horloge instruction suivants.

Lorsque le timer fonctionne en mode 16 bits, le registre TMR0L est complété par 8 bits de poids forts contenus dans le registre TRM0. Comme le montre très clairement la figure 6.2, ce registre n'est cependant pas accessible directement et il faut faire appel

pour le lire ou pour y écrire au registre tampon TMR0H. Ceci étant, et afin de garantir une lecture ou une écriture correcte dans le registre 16 bits constitué par la concaténation de ces deux registres, l'accès au timer 0 en mode 16 bits se passe de la façon suivante.

Lorsque vous lisez TMR0L, le contenu de TMR0 à cet instant est automatiquement recopié dans TMR0H. De ce fait, en lisant ensuite TMR0H vous êtes certain de disposer du contenu réel qu'avait le registre 16 bits au moment de la lecture de ses 8 bits de poids faibles.

Lorsque vous écrivez dans TMR0L, le contenu de TMR0H est automatiquement recopié dans le registre TMR0. Il faut donc l'avoir chargé au préalable avec la valeur que vous voulez placer dans TMR0. Ici aussi, ce processus permet d'écrire une véritable valeur codée sur 16 bits dans le registre du timer sans courir le risque que l'un ou l'autre des deux registres qui le constituent ait pu varier, ce qui aurait pu être le cas si l'écriture avait eu lieu en deux temps.

Que le timer fonctionne en mode 8 ou 16 bits, le débordement de son registre TRM0L ou du couple TMR0L-TMR0 positionne le bit TMR0IF et peut générer une interruption s'il y est autorisé au moyen du bit de validation correspondant (voir chapitre 3 si nécessaire).

Le timer 0 est contrôlé au moyen des bits contenus dans le registre T0CON dont voici précisée la signification :

Accès E : Écriture L : Lecture	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E
T0CON	TMR0ON	T08BITS	T0CS	T0SE	PSA	T0PS3	T0PS2	T0PS1
État à la mise sous tension	1	1	1	1	1	1	1	1

- *Bits 0 à 2 ou bits T0PS0, T0PS1 et T0PS2 pour Timer 0 PreScaler*

Ces trois bits servent à définir le taux de prédivision appliqué à l'entrée du timer 0 conformément aux indications du tableau 6.1.

Tableau 6.1 – Taux de division du prédiviseur du timer 0 en fonction des bits T0PS0 à T0PS2 du registre T0CON.

T0PS2	T0PS1	T0PS0	Taux de division
0	0	0	2
0	0	1	4
0	1	0	8
0	1	1	16

CHAPITRE 7

LES MODULES DE CAPTURE ET COMPARAISON (CCP ET ECCP) ET DE MODULATION DE LARGEUR D'IMPULSIONS (PWM ET EPWM)

Tous les PIC de la famille PIC 18 disposent d'au moins un module capture, comparaison et modulation de largeur d'impulsions qui, en utilisant une partie des ressources des timers 1, 2 et parfois 3 lorsqu'il est présent, permettent de réaliser avec une grande souplesse, de la mesure de temps, de période ou de fréquence et de la génération de signaux périodiques calibrés. L'intérêt de ces modules est de fonctionner de manière quasiment automatique, libérant ainsi au maximum l'unité centrale pour d'autres tâches.

Ces modules s'appellent CCP dans les documentations Microchip, pour *Capture, Compare PWM* ou ECCP pour *Enhanced Capture Compare PWM* lorsqu'ils disposent de fonctions étendues en matière de génération de signaux PWM. Ils peuvent être présents en un ou plusieurs exemplaires identiques selon les versions de circuits PIC. Dans la majorité des cas, ils utilisent trois registres et une ligne d'entrée/sortie qui sont donc numérotés en fonction du nombre de modules CCP présents dans le circuit comme indiqué tableau 7.1.

Tableau 7.1 – Affectation des registres aux divers modules de capture et de comparaison.

Nom générique	CCP1	CCP2	Commentaire
CCPxCON	CCP1CON	CCP2CON	Registre de contrôle
CCPRxH	CCPR1H	CCPR2H	Octet haut registre CCP
CCPRxL	CCPR1L	CCPR2L	Octet bas registre CCP
CCPx	CCP1	CCP2	Entrée/sortie CCP

Le registre CCPxCON est un registre de contrôle définissant le mode de fonctionnement de chaque module tandis que les registres CCPRxL et CCPRxH forment en fait un registre CCPRx sur 16 bits qui est un registre de comptage.

Un module CCP utilise une partie des ressources des timers 1, 2 et 3 lorsqu'il existe selon la fonction réalisée conformément aux indications du tableau 7.2. Il est donc évident que le timer correspondant ne pourra pas être utilisé librement en même temps que la fonction du module CCP correspondante et il importe donc d'en tenir compte.

Tableau 7.2 – Exploitation des timers 1, 2 et 3 selon la fonction du module CCP choisie.

Mode CCP	Ressource utilisée
Capture	Timer 1 ou 3
Comparaison	Timer 1 ou 3
PWM	Timer 2

Enfin, lorsque deux modules CCP distincts sont présents dans un même circuit, il n'est pas toujours possible de choisir librement la fonction de chacun d'eux, du fait justement de l'utilisation des ressources des timers 1, 2 et 3. Le tableau 7.3 montre ainsi l'interaction qui existe entre deux modules CCP distincts selon les fonctions exécutées par chacun d'eux.

Tableau 7.3 – Interactions possibles entre deux modules CCP.

Mode CCP1	Mode CCP2	Interaction
Capture	Capture	Chaque module peut utiliser le timer 1 ou le timer 3 comme base de temps. Elle peut être différente pour CCP1 et CCP2
Capture	Comparaison	CCP2 peut être programmé en mode déclenchement spécial pour remettre le timer 1 ou le timer 3 à 0. Une conversion A/D automatique peut également être programmée. Le fonctionnement de CCP1 peut être affecté s'il utilise la même base de temps que CCP2
Comparaison	Capture	CCP1 peut être programmé en mode déclenchement spécial pour remettre le timer 1 ou le timer 3 à 0. Le fonctionnement de CCP2 peut être affecté s'il utilise la même base de temps que CCP1
Comparaison	Comparaison	CCP1 ou CCP2 peut être programmé en mode déclenchement spécial pour remettre le timer 1 ou le timer 3 à 0. Une conversion A/D automatique peut également être programmée sur CCP2. Des conflits peuvent avoir lieu s'ils utilisent la même base de temps
PWM	PWM	Fréquence des signaux identiques
PWM	Capture	Aucune
PWM	Comparaison	Aucune
Capture	PWM	Aucune
Comparaison	PWM	Aucune

Ces limitations sont toutefois assez peu contraignantes lorsque l'on connaît la puissance des différents modes de travail des modules CCP que nous allons découvrir maintenant.

7.1 Le mode capture

Dans ce mode, le registre CCPRx capture le contenu du timer 1 ou du timer 3 à l'une des échéances suivantes sur la patte CCPx correspondante :

- tous les fronts descendants ;
- tous les fronts montants ;
- tous les quatre fronts montants ;
- tous les seize fronts montants.

Le choix entre ces divers événements est réalisé grâce aux bits contenus dans le registre CCPxCON que nous verrons dans un instant.

Lorsqu'une capture a lieu, le bit CCPxIF est positionné et une interruption peut être générée sous réserve d'y être autorisée par le bit approprié du registre PIEx correspondant présenté au chapitre 3. Si une autre capture a lieu avant que le contenu de CCPRx ait été lu, ce contenu est écrasé par le nouveau. Par contre, une capture ne remet pas à 0 le contenu des registres CCPRxL et CCPRxH du timer 1 ou du timer 3 (selon le cas) qui continuent à évoluer normalement. Ceci permet, sous certaines conditions, de l'utiliser simultanément à d'autres tâches.

Pour que le système fonctionne correctement, le timer 1 (ou 3 selon le cas) doit fonctionner en mode timer ou en mode compteur synchronisé par l'horloge interne. S'il fonctionne en mode non synchronisé, la capture peut ne pas se dérouler correctement.

Notez également que pour que ce système fonctionne correctement, la patte CCPx doit évidemment être programmée en entrée. Si elle est configurée en sortie et qu'une écriture dans le registre du port parallèle avec lequel elle est commune a lieu, une fausse capture peut être générée.

La figure 7.1 montre de façon très visuelle le fonctionnement de ce mode de capture dans le cas d'un PIC 18 équipé d'un double module de capture et de deux timers 1 et 3.

Les registres de comptage du timer 1 et du timer 3 fonctionnent en continu sous le contrôle de l'horloge qui leur a été affectée et, lorsque l'événement choisi se produit sur l'entrée CCP correspondante, le tampon qui les relie aux registres CCPR est validé et permet donc le transfert de leur contenu à cet instant dans ces registres.

Remarquez, au niveau des entrées CCP, la présence du prédiviseur qui permet la capture tous les fronts, tous les quatre fronts ou tous les seize fronts. Son contenu est mis à zéro lors d'un reset ainsi que lors de tout arrêt du module CCP.

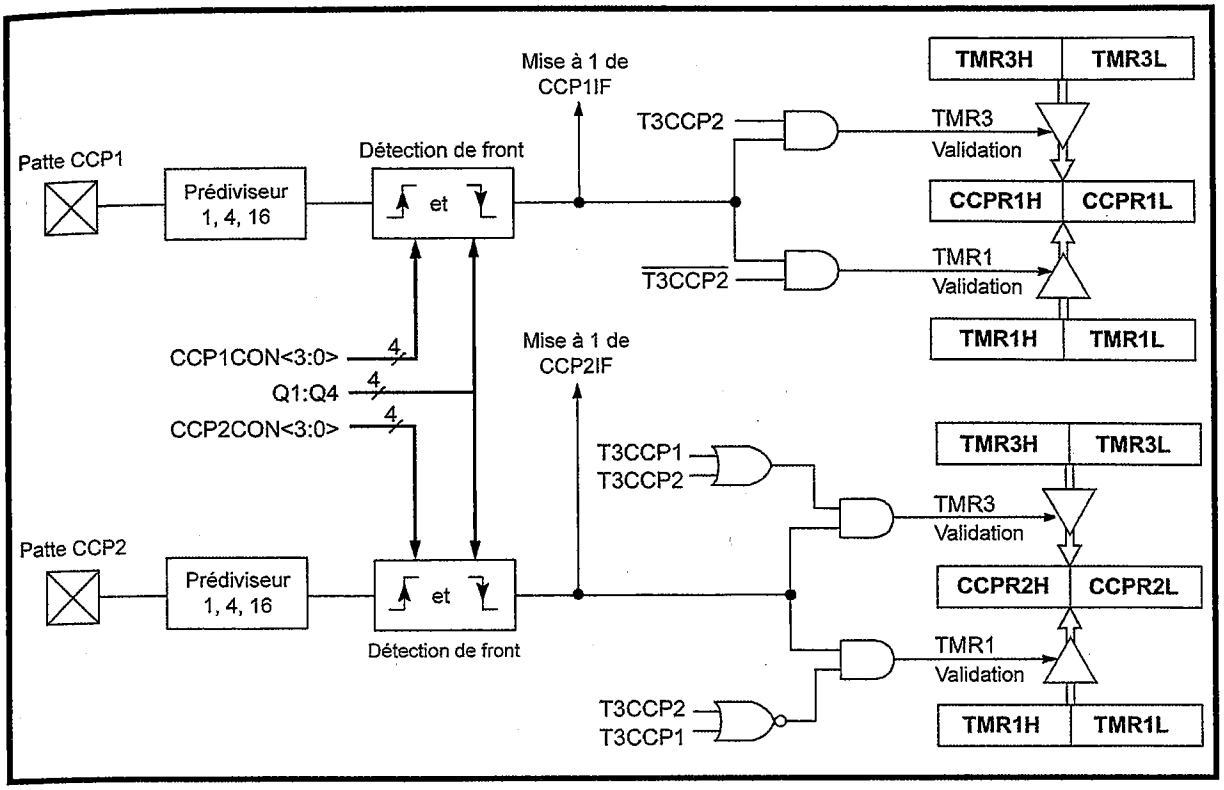

Figure 7.1 – Schéma équivalent du module CCP en mode capture.

Prenez garde au fait que, si vous changez le taux de division de ce prédiviseur afin de définir d'autres conditions de capture, une fausse capture peut être générée. Il est donc conseillé d'arrêter le module CCP ou d'interdire temporairement les interruptions émanant du module CCP avant de changer les conditions de capture pour éviter ce phénomène.

7.2 Le mode comparaison

Dans ce mode, le contenu du registre CCPRx est comparé en permanence avec le contenu du timer 1 (ou 3 selon le cas). Lorsque l'égalité a lieu, l'un des phénomènes suivants peut se produire :

- passage au niveau haut de la patte CCPx correspondante ;
- passage au niveau bas de la patte CCPx correspondante ;
- changement d'état de la patte CCPx correspondante (niveau haut s'il était bas et vice versa) ;
- aucun changement d'état de la patte mais génération d'une interruption si bien sûr le module CCP y est autorisé.

Dans les trois premiers cas, la patte correspondante du port parallèle avec lequel elle est commune doit évidemment être programmée en sortie par mise à zéro du bit correspondant du registre TRISx.

Pour que le système fonctionne correctement, le timer 1 ou 3 (selon le cas) doit fonctionner en mode timer ou en mode compteur synchronisé par l'horloge interne. S'il fonctionne en mode non synchronisé, la comparaison peut ne pas se dérouler correctement.

La figure 7.2 montre de façon très visuelle le principe de fonctionnement de ce mode comparaison dans le cas d'un PIC 18 équipé d'un double module de capture et de deux timers 1 et 3.

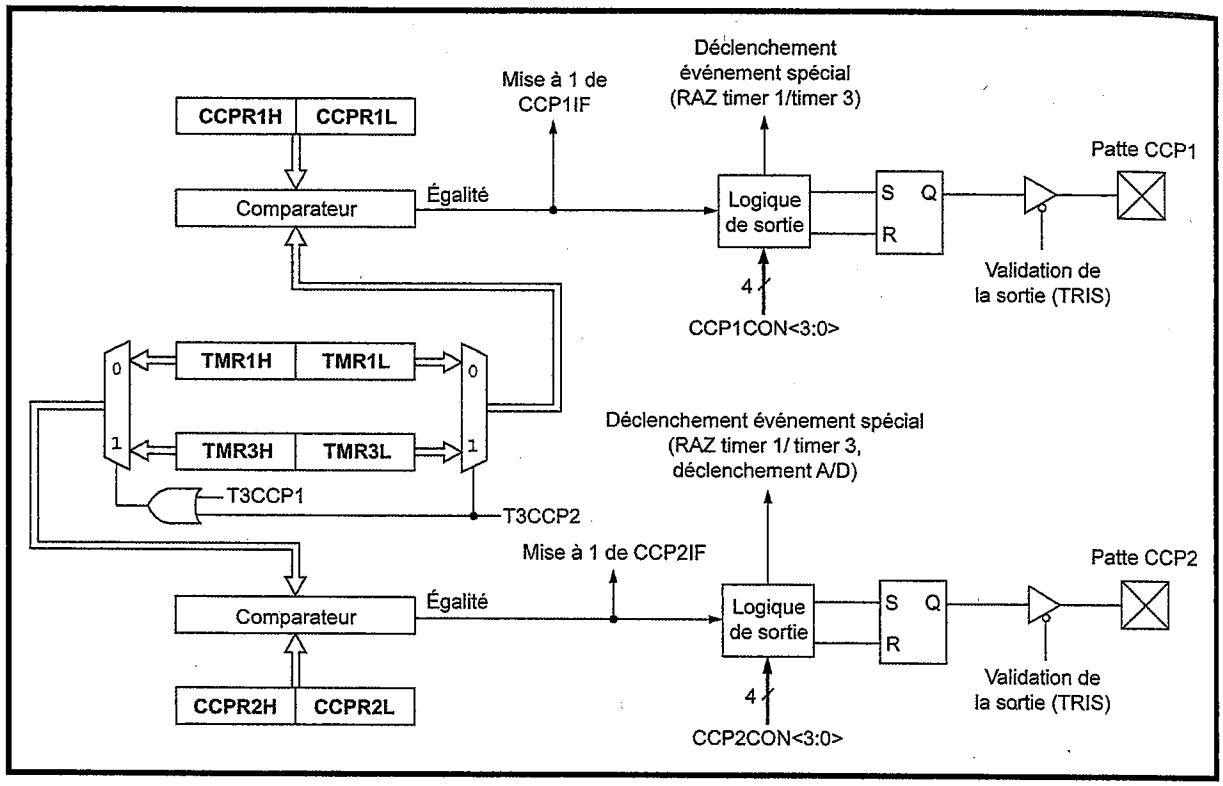

Figure 7.2 – Schéma équivalent du module CCP en mode comparaison.

Les registres de comptage des timers 1 et 3 fonctionnent en continu sous le contrôle de l'horloge qui leur a été affectée et leur contenu est comparé en permanence au contenu des registres CCPR. Lorsque le comparateur détecte une égalité, il génère un signal de sortie qui peut, selon le cas, aboutir sur la patte de sortie CCP correspondante ou bien seulement avoir une action interne.

Un mode supplémentaire est disponible et s'appelle mode déclenchement spécial. Dans ce cas, une comparaison réussie provoque les événements suivants :

- le timer 1 (ou 3 selon le cas) est remis à 0. Ceci permet d'utiliser le registre CCPRx correspondant comme un registre de période programmable sur 16 bits pour le timer 1 ou le timer 3 ;
- dans certains circuits, le timer 1 ou 3 est remis à 0 et une conversion analogique/digitale est déclenchée, si bien sûr la fonction correspondante est autorisée au niveau du convertisseur A/D. Ceci permet de réaliser très facilement des conversions à intervalles réguliers sans qu'il soit nécessaire de faire appel à l'unité centrale.

7.3 Le mode modulation de largeur d'impulsions ou mode PWM

Dans ce mode, la patte CCPx correspondant au module CCP choisi permet de disposer d'une sortie d'impulsions modulées en largeur ou PWM (*Pulse Width Modulation*) avec une résolution pouvant aller jusqu'à 10 bits. La patte correspondante du port parallèle avec laquelle elle est commune doit évidemment être configurée en sortie au moyen du bit adéquat du registre TRISx

Comme le montre la figure 7.3, le principe de ce mode de fonctionnement est le suivant. L'utilisateur écrit dans le registre CCPRxL le rapport cyclique désiré, codé sur 8 bits si cette résolution suffit. Si une résolution supérieure est désirée, les 8 bits de poids forts sont toujours écrits dans CCPRxL tandis que le ou les 2 bits de poids faibles sont placés dans les bits DCxB1 et DCxB0 du registre CCPxCON. Le registre CCPRxH est utilisé comme esclave de CCPRxL et ne peut donc être écrit par vos soins.

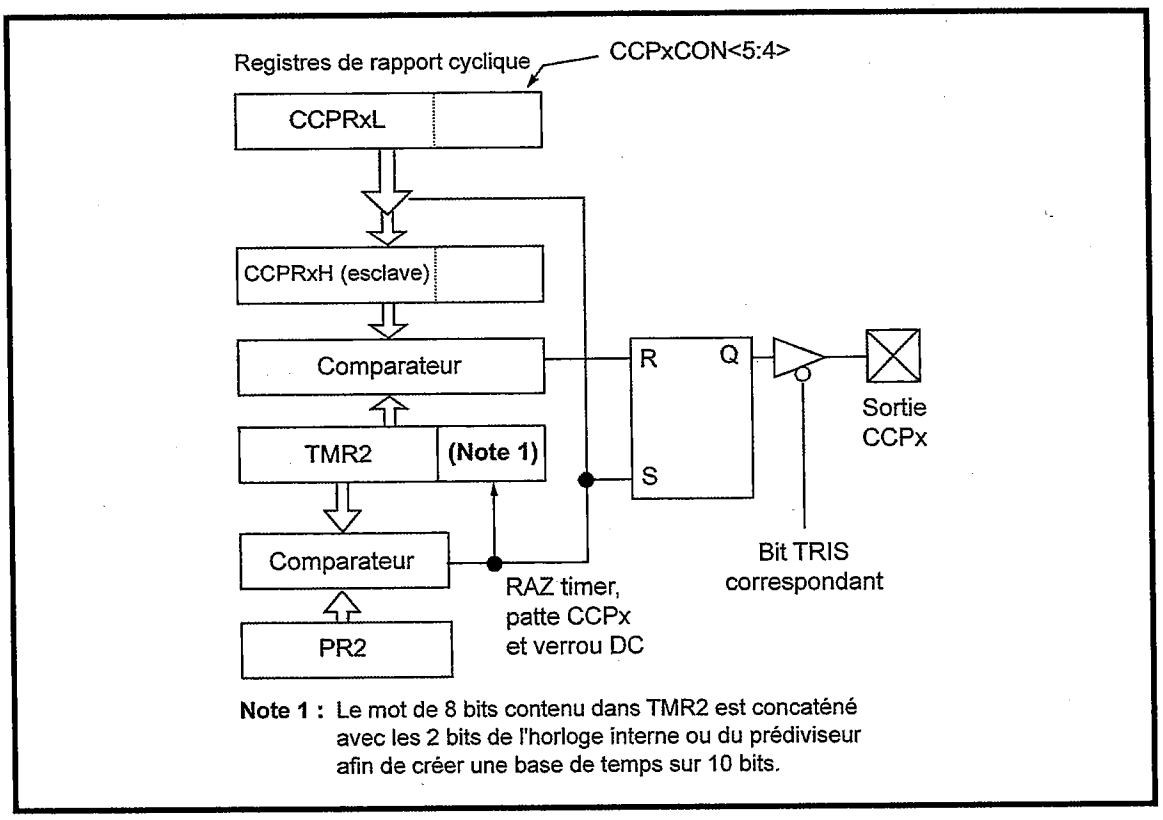

Figure 7.3 – Schéma équivalent du module CCP en mode PWM.

La période du signal généré quant à elle est déterminée par le contenu du registre période du timer 2 (PR2). Dans ces conditions, et compte tenu du principe de fonctionnement du timer 2 vu au chapitre 6, on peut écrire :

Période du signal = (PR2 + 1) . 4 T_{OSC} . (Contenu du prédiviseur de TMR2)

où T_{OSC} est la période du signal d'horloge du PIC.

Rapport cyclique = DC1 . T_{OSC} . (Contenu du prédiviseur de TMR2)

où T_{OSC} est la période d'horloge du PIC et où DC1 est le mot de 10 bits formé des 8 bits contenus dans CCPRxL et des bits DCxB0 et DCxB1 de CCPxCON utilisés comme bits de poids faibles.

La résolution de ce mode PWM est donc bien programmable jusqu'à pouvoir atteindre 10 bits si ces bits DCxB0 et DCxB1 sont différents de zéro. Dans le cas contraire, le mode PWM fonctionne en résolution sur 9 bits (un seul bit à zéro) ou 8 bits (les deux bits à zéro).

Compte tenu du principe de fonctionnement du mode PWM et des registres qui y sont utilisés, il est possible de dresser deux tableaux mettant en évidence diverses limites techniques. Le tableau 7.4 présente tout d'abord la résolution minimum qu'il est possible d'obtenir en fonction de la programmation du prédiviseur du timer 2.

Tableau 7.4 – Résolution minimum du mode PWM.

Prédiviseur	T2CKPS1	T2CKPS0	Résolution minimum
1	0	0	T_{OSC}
4	0	1	T_{CY}
16	1	X	4 T_{CY}

Le tableau 7.5 quant à lui indique la résolution maximale qu'il est possible d'atteindre en fonction de la fréquence du signal PWM généré, son contenu ayant été calculé pour un PIC fonctionnant avec un quartz à 40 MHz (horloge système à 10 MHz donc).

Tableau 7.5 – Fréquence type et résolution du mode PWM pour une fréquence d'horloge de 40 MHz.

Fréquence (kHz)	2,44	9,77	39,06	156,25	312,50	416,67
Prédiviseur	16	4	1	1	1	1
Contenu de PR2	0xFF	0xFF	0xFF	0x3F	0x1F	0x17
Résolution maximale (nombre de bits)	10	10	10	8	7	6,58

7.4 Le registre de contrôle des modules CCP

Ces divers modes de fonctionnement étant vus, l'analyse du contenu du registre de contrôle de chaque module CCP va être facilitée, ce qui justifie le fait que nous ne vous la présentions que maintenant.

Chaque module CCP dispose de son propre registre de contrôle, appelé CCPxCON, dont chaque bit a la fonction que voici :

Accès								
E : Écriture L : Lecture	L	L	L/E	L/E	L/E	L/E	L/E	L/E
CCPxCON	—	—	DCxB1	DCxB0	CCPxM3	CCPxM2	CCPxM1	CCPxM0
État à la mise sous tension	0	0	0	0	0	0	0	0

- *Bits 0 à 3 ou bits CCPxM0 à CCPxM3 pour CCPx Mode*

Ces bits permettent de sélectionner le mode de fonctionnement en capture, comparaison ou PWM selon les indications du tableau 7.6 dans lequel on retrouve bien toutes les configurations que nous venons d'étudier.

Tableau 7.6 – Programmation des bits de modes de fonctionnement des modules CCP.

CCPxM3	CCPxM2	CCPxM1	CCPxM0	Mode
0	0	0	0	Arrêt tous modes
0	0	1	0	Changement d'état sortie si comparaison réussie
0	1	0	0	Capture tous les fronts descendants
0	1	0	1	Capture tous les fronts montants
0	1	1	0	Capture tous les 4 fronts montants
0	1	1	1	Capture tous les 16 fronts montants
1	0	0	0	Sortie à 1 si comparaison réussie
1	0	0	1	Sortie à 0 si comparaison réussie
1	0	1	0	Interruption si comparaison réussie
1	0	1	1	Mode déclenchement spécial
1	1	X	X	Mode PWM

- *Bits 4 et 5 ou bits DCxB1 et DCxB0 pour Duty Cycle Bits*

Ces deux bits ne sont utilisés qu'en mode PWM. Ils constituent les deux bits de poids faibles lorsque l'on veut travailler en mode résolution sur 10 bits. S'ils sont laissés à 0, le mode PWM fonctionne alors en résolution sur 8 bits ou sur 9 bits.

- *Bits 6 et 7*

Non utilisés et lus comme étant à 0.

7.5 Le mode EPWM ou mode PWM étendu

Ce mode, appelé *Enhanced PWM*, ce qui signifie PWM amélioré mais que nous trouvons plus logique d'appeler PWM étendu, est disponible dans de nombreux circuits de la famille PIC 18 afin de faciliter, entre autres choses, le contrôle de moteurs. Sur ces circuits, il cohabite avec le mode PWM normal que nous venons de présenter et qui reste donc disponible, ainsi bien sûr que les modes comparaison et capture.

Sa principale nouveauté par rapport au mode PWM « classique » est de permettre la génération de signaux PWM sur une, deux ou quatre sorties différentes du PIC afin de permettre la commande de moteurs selon des schémas classiques de type demi-pont ou pont entier comme indiqué figure 7.4.

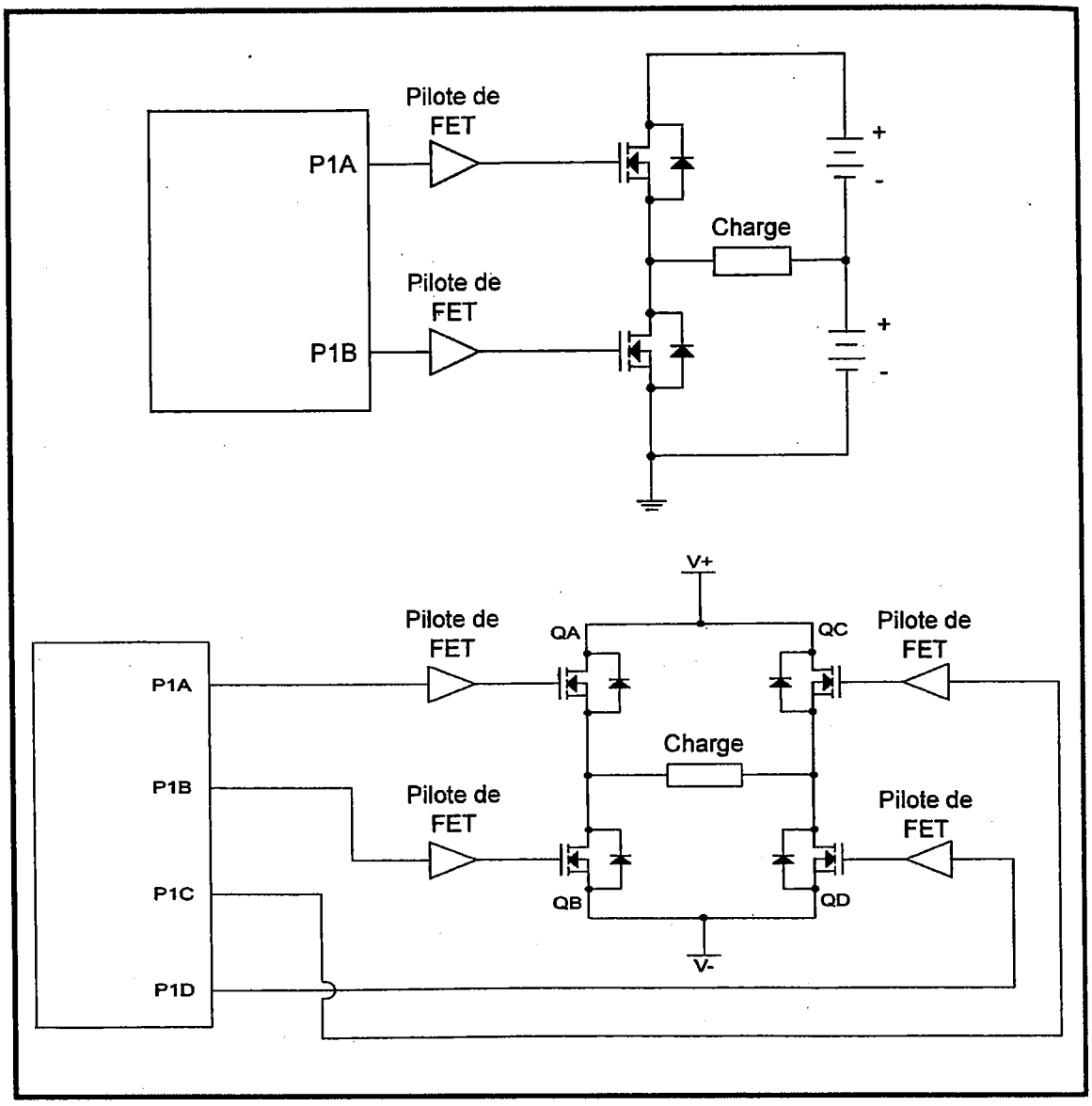

Figure 7.4 – Commande de moteurs en mode demi-pont ou pont entier grâce au mode PWM étendu.

Même si de nombreuses fonctionnalités spécifiques de la commande de moteurs ont été ajoutées à la fonction PWM de base, le principe de fonctionnement de ce mode PWM étendu reste identique à celui vu ci-dessus. Le schéma synoptique de la génération de ces signaux peut en effet être représenté comme indiqué figure 7.5.

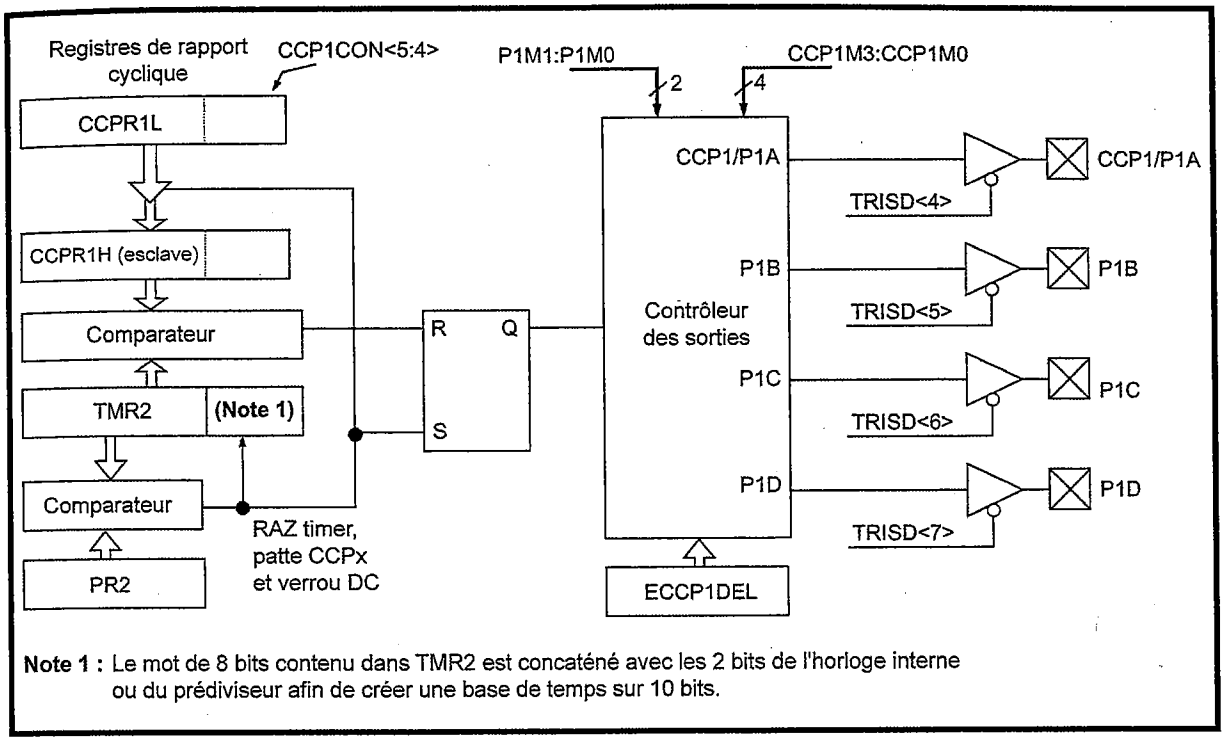

Figure 7.5 – Schéma équivalent du module PWM étendu.

Comme vous pouvez le constater, la partie gauche de cette figure est identique à la figure 7.3, ce qui implique que la programmation des caractéristiques des signaux PWM générés, en termes de période et de rapport cyclique, reste similaire à ce que nous venons d'étudier pour le mode PWM classique.

Par contre, un nouveau bloc fonctionnel fait son apparition sur la droite de cette figure ; bloc qui permet, entre autres choses, de répartir les signaux PWM vers les quatre sorties disponibles sous le contrôle des deux bits P1M1 et P1M0 qui font leur apparition dans le registre de contrôle CCP1CON des circuits disposant de la ressource PWM étendue.

Pour ces seuls circuits, ce registre CCP1CON adopte donc la configuration suivante :

Accès E : Écriture L : Lecture	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E
CCP1CON (mode EPWM)	P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0
État à la mise sous tension	0	0	0	0	0	0	0	0

CHAPITRE 17

PROGRAMMATION NORMALE, PROGRAMMATION EN CIRCUIT OU ICSP ET MODE *IN-CIRCUIT DEBUGGER* OU ICD

Comme bon nombre de microcontrôleurs actuels, les microcontrôleurs PIC de la famille PIC 18 supportent deux modes de programmation : la programmation « normale », c'est-à-dire encore celle réalisée en mettant le circuit sur un programmateur dédié, et la programmation « en circuit », c'est-à-dire réalisée sans devoir enlever le circuit de l'application sur laquelle il est implanté.

Ils peuvent à leur tour fonctionner de deux façons différentes : un mode « haute » tension, que l'on peut considérer comme une survivance du passé, qui nécessite une tension de programmation de 12 ou 13 V, et un mode basse tension qui se contente de la seule tension d'alimentation de 5 V du circuit.

Par ailleurs, tous les PIC de la famille PIC 18 supportent également un mode de fonctionnement particulier appelé *in-circuit debugger*, que l'on peut traduire en français par mise au point (c'est plus joli que « débugging » ou « débogage ») en circuit.

Dans ce mode particulier, et sous réserve de disposer d'une interface adéquate telle que, par exemple, le module ICD ou ICD2 de Microchip, ou bien encore un produit compatible, il devient possible, alors même que le PIC est implanté sur son application définitive avec son logiciel d'exploitation, de mettre au point ce dernier. Ce mode particulier permet en effet de faire de l'exécution en pas à pas ou de la pose de points d'arrêt au sein même du programme de votre propre application pendant que celui-ci s'exécute.

Sans permettre toutes les vérifications que rend possible un émulateur en circuit, ce mode facilite cependant grandement la mise au point de programmes tout en étant très peu contraignant au niveau de son interface avec le PIC de votre application.

17.1 La programmation « haute tension »

Ce premier mode de programmation est qualifié de « haute tension » car il impose de disposer, pendant la phase de programmation du circuit, d'une tension de l'ordre de 13 V. C'était le seul mode disponible lors de la première introduction des mémoires Flash dans les microcontrôleurs de la famille PIC 16 ; mode qui a contribué au succès planétaire du PIC 16C84 qui fut le premier circuit a en être équipé. Grâce aux progrès de la technologie des mémoires Flash, il est aujourd'hui possible de les programmer avec une unique tension de 5 V mais, par souci de compatibilité ascendante et également pour conserver un maximum de lignes de ports parallèles, il est toujours présent, même sur les plus récents des PIC 18.

La programmation d'un PIC 18 en mode haute tension nécessite seulement d'accéder à cinq pattes du boîtier : les deux lignes d'alimentation V_{SS} et V_{DD}, la ligne de reset $\overline{MCLR}$qui s'appelle aussi V_{PP} dans ce cas, et deux lignes du port parallèle B, RB6 qui s'appelle alors CLOCK et RB7 qui s'appelle alors DATA.

Pour faire passer le circuit en mode programmation haute tension, il faut maintenir ses lignes de ports parallèles RB6 et RB7 au niveau bas pendant que l'on fait monter la tension sur l'entrée de reset $\overline{MCLR}$ de V_{IL} à V_{IHH} (voir fiche technique de chaque circuit pour la valeur exacte de ce paramètre généralement voisin de 13 V) et que la tension d'alimentation positive V_{DD} du circuit adopte la valeur de la tension de programmation indiquée elle aussi dans la fiche technique du circuit (généralement 5 V).

RB6 devient alors l'horloge de programmation et se comporte donc comme une entrée, alors que RB7 devient l'entrée/sortie série des données. Elle fonctionne en entrée pendant toute la phase de programmation proprement dite et en sortie lors de la phase de vérification. RB6 et RB7 disposent de trigger de Schmitt en entrée alors que RB7 est un buffer CMOS lorsqu'elle fonctionne en sortie.

Pendant toute la durée de la programmation, le timer chien de garde est automatiquement invalidé afin d'éviter qu'il génère un reset qui serait alors pour le moins indésirable !

Ce mode de programmation peut être mis en œuvre sur un programmateur spécialisé, du commerce ou de réalisation personnelle, tel ceux que nous vous proposons dans nos divers autres ouvrages consacrés aux PIC : *Les microcontrôleurs PIC – Recueil d'applications*, *Programmation en C des PIC*, *Microcontrôleurs PIC : programmation en Basic*, tous publiés chez Dunod ; programmateur dont vous pouvez également découvrir un autre schéma sur le site Internet de l'auteur : www.tavernier-c.com.

17.2 La programmation basse tension ou LVP

Même si ce que nous venons de voir reste valide pour tous les microcontrôleurs PIC 18, ces derniers disposent également tous d'un mode de programmation supplémentaire appelé LVP pour *Low Voltage Programming* ce qui veut dire, bien entendu, programmation basse tension.

Comme son nom le laisse supposer, ce mode de programmation rend inutile toute « haute tension » de programmation puisque toute l'opération peut être réalisée avec la seule tension d'alimentation normale de 5 V.

Par contre, il faut sélectionner explicitement ce mode de programmation en positionnant correctement le bit LVP du registre de configuration CONFIG4L (voir chapitre 3) ; bit qui, par défaut c'est-à-dire lorsque le circuit est vierge, est programmé à 1 ce qui valide le mode LVP.

17.2.1 Le passage en mode programmation LVP

Le seul « défaut » de la programmation basse tension, qui peut justifier le maintien du mode haute tension, est qu'il consomme une ligne de port parallèle de plus, en l'occurrence RB5.

Dans ce mode en effet, cette patte RB5 acquiert une signification nouvelle car c'est alors elle qui décide du passage du circuit en mode programmation, passage qui a lieu de la façon suivante.

- application de la tension d'alimentation normale V_{DD} à la patte V_{DD} du circuit ;
- mise au niveau bas de la patte de reset $\overline{MCLR}$;
- application de la tension V_{DD} à la patte RB5 qui s'appelle dans ce cas PGM, ce qui a pour effet de mettre effectivement le circuit en mode programmation ;
- mise au niveau V_{DD} de la patte de reset $\overline{MCLR}$;
- dès cet instant, RB6 et RB7 acquièrent les mêmes fonctions que celles vues pour la programmation « haute tension » et le cycle se déroule ensuite de la même façon.

17.2.2 Les contraintes de la programmation LVP

Le fait d'utiliser la programmation LVP induit quelques contraintes que voici résumées :

- la ligne de port RB5 ne peut plus être utilisée en tant qu'entrée/sortie à usage général lorsque le bit LVP du registre de configuration du circuit est mis à 1 ;
- si les résistances de tirage au niveau haut du port B sont validées (voir chapitre 5 si nécessaire), il faut impérativement mettre à 0 le bit 5 du registre TRISB afin de désactiver la résistance reliée à la ligne RB5 faute de quoi le passage en mode programmation LVP n'aurait pas lieu correctement ;
- la ligne RB5 ne doit en aucun cas pouvoir se retrouver flottante lorsque le mode LVP est utilisé au risque de mettre le circuit en mode programmation de façon imprévisible. Il faut donc la maintenir au niveau bas lors du fonctionnement normal du circuit au moyen d'une résistance de tirage vers la masse par exemple.

Par contre, si la programmation en mode LVP n'est pas utilisée, la patte RB5 redevient une entrée/sortie à usage général.

Notez également que, comme le bit de sélection entre la programmation LVP et la programmation normale n'est accessible que dans le registre de configuration CONFIG4L du circuit, il ne peut être modifié que par programmation de ce dernier. Or, les circuits étant livrés par défaut avec ce bit positionné à 1, ce qui valide le mode LVP comme nous l'avons vu ci-dessus, il semblerait donc que la programmation LVP soit le seul choix possible, au moins lors de la première programmation du circuit c'est-à-dire le temps de modifier ce bit. Afin de ne pas vous imposer cette contrainte, Microchip a fait en sorte que la programmation classique « haute tension » fonctionne normalement, même si le circuit est configuré en mode LVP.

Notez par ailleurs que si, lors de la programmation du circuit, vous mettez à 0, le bit LVP du registre de configuration, vous interdisez la programmation LVP ou, en d'autres termes, vous n'autorisez plus que la seule programmation haute tension. Pour revenir à la programmation LVP il vous faudra donc nécessairement recourir au préalable à la programmation haute tension afin de remettre ce bit à 1.

17.3 La programmation en circuit ou ICSP

Tous les microcontrôleurs PIC de la famille PIC 18 supportent la programmation en circuit appelée aussi ICSP pour *In Circuit Serial Programming*. Ce mode de programmation particulier permet de programmer la mémoire du microcontrôleur alors que celui-ci est déjà installé dans l'application finale. Il est ainsi possible de stocker à l'avance des produits vierges et de les personnaliser au moment de la livraison en fonction des commandes des clients. Ce mode de programmation permet aussi de mettre très facilement à jour des produits existants en remplaçant le programme contenu dans la mémoire par une version plus récente.

Enfin, indépendamment de toute notion de programmation en circuit, cette façon de faire simplifie la réalisation des programmateurs puisque l'on passe d'un programmateur dédié, avec une circuiterie parfois complexe et de multiples supports destinés à recevoir les multiples boîtiers des PIC, à un programmateur très simple muni d'un unique connecteur à quelques contacts qu'il ne reste plus qu'à raccorder à l'application sur laquelle se trouve déjà implanté le PIC à programmer.

En fait, la programmation en circuit ne diffère en aucune manière de la programmation haute ou basse tension que nous venons de voir. Elle utilise les mêmes lignes et les mêmes principes mais, du fait qu'elle est réalisée sans devoir retirer le circuit de l'application, il faut prendre quelques précautions lors de la conception de cette dernière afin qu'elle puisse « partager » ses ports avec la programmation en circuit. La figure 17.1 montre ainsi un exemple de schéma de principe de ce partage et va servir de base à nos explications.

La ligne $\overline{\text{MCLR}}$ est généralement connectée à une circuiterie de reset de type R-C ou similaire. Comme elle doit pouvoir être amenée à la tension V_{IHH} avec un temps de montée relativement court de surcroît, une circuiterie d'isolement doit être prévue, par exemple au moyen d'une diode comme schématisé sur la figure. Afin d'éviter que le seuil de la diode ne modifie de manière trop importante celui de la circuiterie de reset, cette dernière peut être avantageusement de type Schottky.

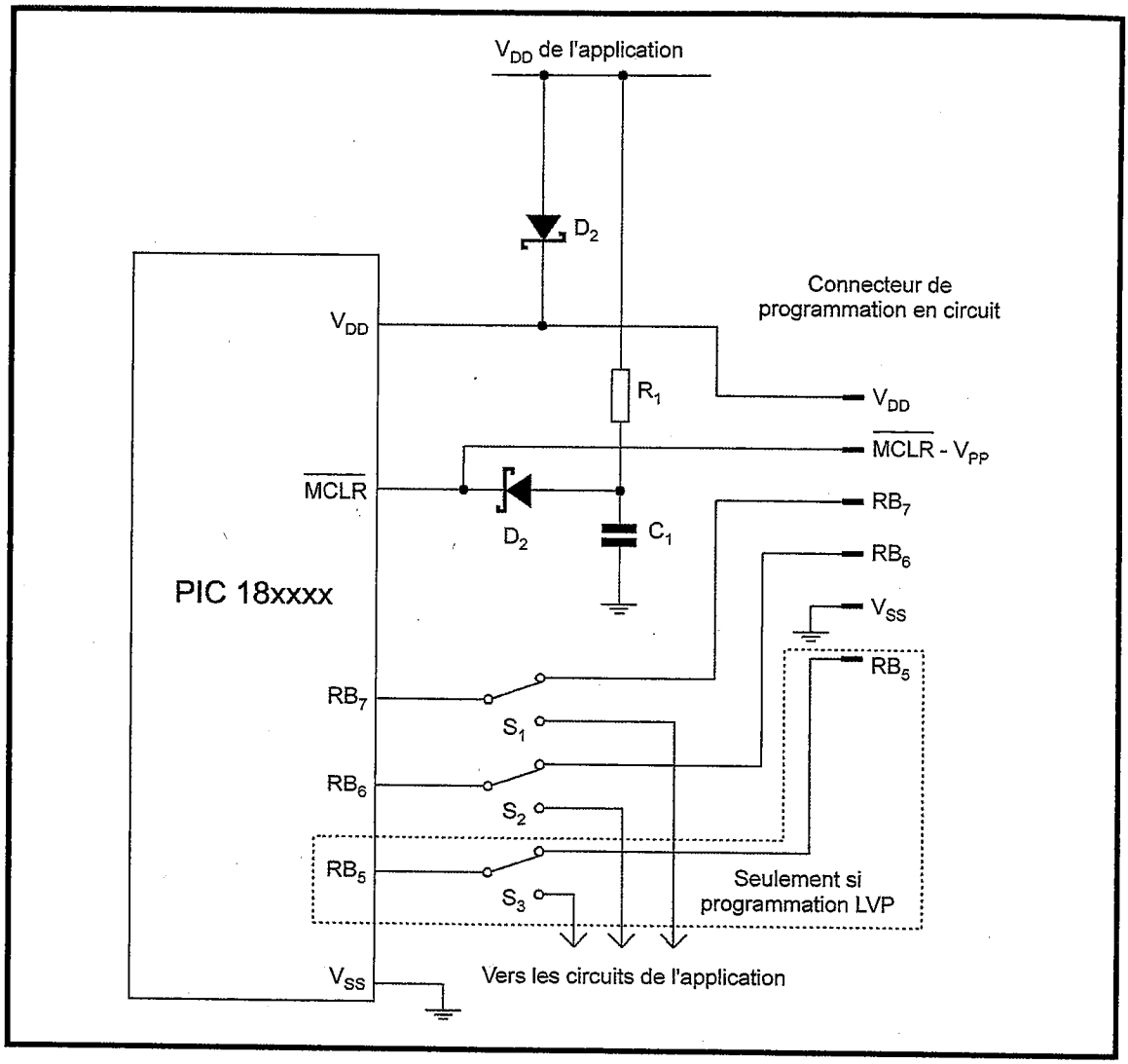

Figure 17.1 – Schéma de principe de la programmation en circuit ou ICSP.

Les pattes RB6 et RB7 (et éventuellement RB5 si la programmation basse tension doit être prévue), utilisées respectivement comme lignes d'horloge et de données pendant la programmation, doivent pouvoir être isolées de la circuiterie du reste de l'application. Le schéma de cette « isolation » dépend évidemment de l'application et aucun schéma général ne peut être fourni dans le cadre de cet ouvrage. Pour sa conception, notez bien que RB6 (et RB5) ne fonctionne qu'en entrée pendant la programmation, puisque c'est l'horloge de programmation, alors que RB7 doit pouvoir fonctionner en entrée et en sortie. En outre, comme ces deux lignes

véhiculent des signaux logiques aux chronogrammes bien précis, il importe de minimiser les capacités parasites à leur niveau afin de ne pas dégrader les transitions des signaux logiques. Afin de simplifier la réalisation de cette partie de la schématique de la programmation en circuit, Microchip recommande d'essayer de respecter les indications suivantes, présentées par ordre de préférence :

- ne pas utiliser RB6 et RB7 (et éventuellement RB5) dans l'application, ce qui est évidemment la solution idéale ;
- n'utiliser ces lignes qu'en sorties avec une charge capacitive minimale ;
- et enfin réaliser une circuiterie d'isolation conforme aux indications ci-dessus si les deux solutions précédentes n'ont pu être mises en œuvre.

Pour ce qui est de l'alimentation, il faut que le programmateur puisse piloter l'alimentation du microcontrôleur pendant toute la phase de programmation. En effet, d'une part la tension d'alimentation doit être égale à 5 V pendant que la programmation a lieu, d'autre part celle-ci doit pouvoir être réglée sur 2,7 et 4,5 V en phase de vérification de la programmation, tout au moins sur les programmateurs dits de production. Les programmateurs de laboratoire ou destinés à un usage « amateur » peuvent quant à eux se limiter à une simple vérification de la programmation sous la tension d'alimentation nominale de 5 V. Compte tenu des éléments associés au microcontrôleur sur l'application concernée, une circuiterie d'isolation peut donc être rendue nécessaire ou pas. Elle peut être réalisée au moyen d'un simple commutateur manuel si le processus de programmation en circuit n'a pas besoin d'être automatisé, ou faire appel à un commutateur électronique à diodes et/ou transistors dans le cas contraire.

Même si ces diverses solutions de commutation électronique sont acceptables, la meilleure solution consiste cependant à faire appel à des cavaliers ou, plus exactement, à un bouchon de court-circuit que l'on met en place en fonctionnement normal et que l'on enlève pour le remplacer par le connecteur de liaison au programmateur en circuit.

Afin que cette opération puisse être exécutée avec un maximum de facilité, il suffit d'implanter sur l'application un connecteur de programmation « intelligent » tel celui que nous vous proposons figure 17.2.

Lorsque le bouchon de court circuit est en place sur ce connecteur, il relie les contacts qui se font face et raccorde donc normalement V_{DD}, $\overline{MCLR}$, RB5, RB6 et RB7 à la circuiterie de l'application.

Lorsque le bouchon de court-circuit est enlevé pour être remplacé par le connecteur du programmateur, les plots de droite du connecteur intelligent ne sont plus raccordés au PIC tandis que les lignes V_{DD}, $\overline{MCLR}$, RB5, RB6 et RB7 de ce dernier sont librement pilotées par le programmateur.

Notez que si la programmation basse tension n'est pas nécessaire, la ligne RB5 n'a plus besoin d'être prise en compte et peut donc disparaître de ce connecteur.

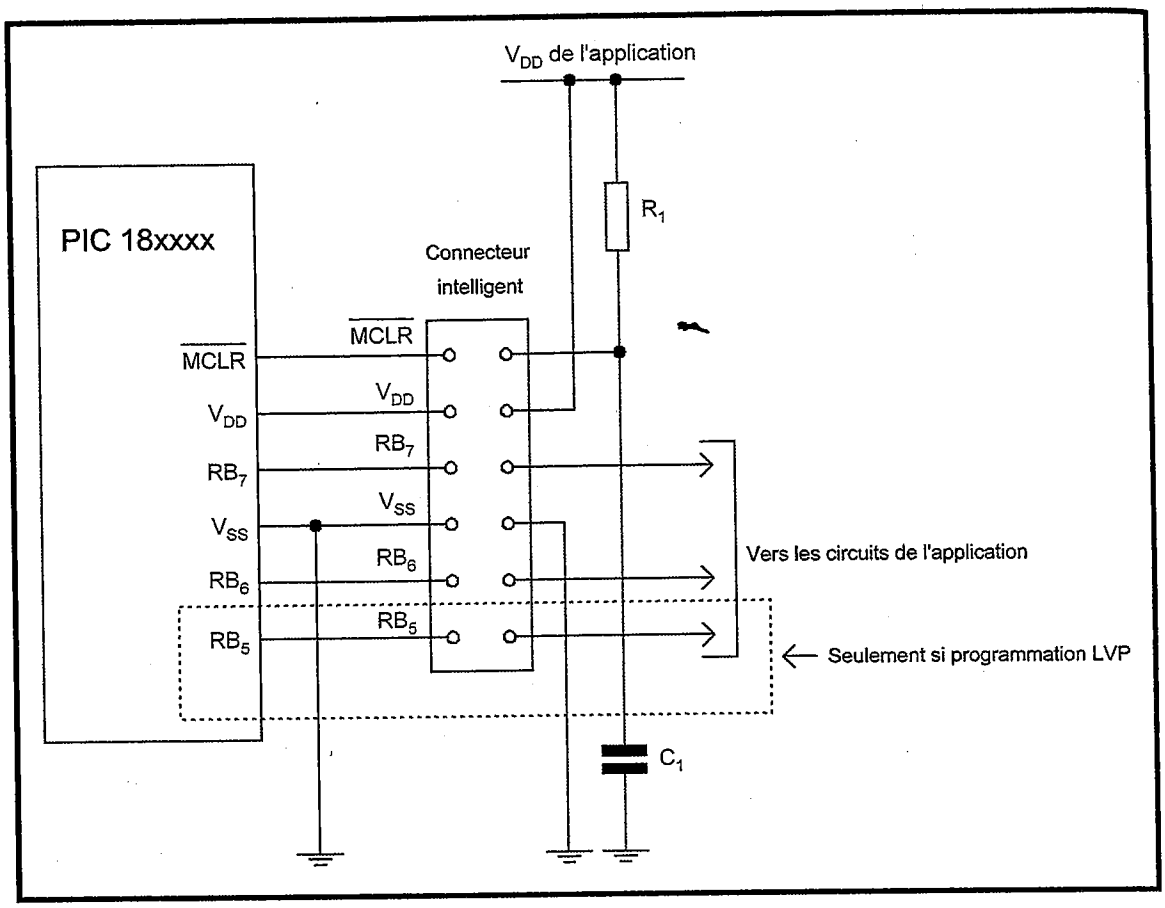

Figure 17.2 – Connecteur de programmation en circuit « intelligent ».

Hormis ces problèmes de connectique, il importe de prendre des précautions au niveau de l'oscillateur d'horloge système du PIC car il faut absolument éviter que celle-ci puisse commencer à fonctionner avant que le circuit soit mis en mode programmation. En effet, le PC est utilisé comme compteur pendant la phase de programmation et il est censé partir de 000. Or, si l'horloge système a commencé à fonctionner, le PC n'est plus à 0 et la programmation de la mémoire de programme est réalisée avec un décalage qui la rend alors inutilisable.

Si l'oscillateur d'horloge est un modèle à quartz, ce problème ne se pose pas puisque, comme nous l'avons vu au chapitre 2, un délai de 1 024 cycles d'oscillateur est nécessaire avant que l'horloge système interne démarre. Par contre, si l'oscillateur est de type RC, ce délai se réduit à 4 cycles ce qui est très court. Dans ce dernier cas, il est donc nécessaire de prévoir un moyen d'inhiber cet oscillateur pendant la phase de programmation en circuit, par exemple en déconnectant la résistance externe de cet oscillateur ou bien encore en reliant à la masse la patte OSC1 du microcontrôleur.

Compte tenu du fait que les lignes RB5, RB6 et RB7 véhiculent des signaux logiques mais aussi que $\overline{\text{MCLR}}$ doit changer de niveau avec des transitions rapides, le câble de liaison entre le connecteur de programmation en circuit placé sur l'application et le programmateur doit être maintenu aussi court que possible et doit être à faible capacité parasite.

17.4 Le mode *in-circuit debugger* ou ICD

Ainsi que nous l'avons expliqué en début de chapitre, ce mode de fonctionnement particulier, disponible sur tous les PIC de la famille PIC 18, permet de faire très facilement de la mise au point de programme (du débogage si vous préférez, bien que cette francisation de *debugging* n'ait aucun sens). Il permet en effet de poser des points d'arrêt, d'exécuter des programmes en pas à pas, de visualiser le contenu des registres internes, etc. alors même que le PIC est déjà programmé avec le logiciel de votre application et qu'il est implanté sur cette dernière.

Il faut pour cela utiliser un *debugger* en circuit tel que par exemple le modèle ICD2 de Microchip, à moins que vous ne préfériez un produit fabriqué par un de ses partenaires.

Pour pouvoir utiliser ce mode de fonctionnement particulier il faut réunir deux conditions :

- programmer le bit $\overline{\text{DEBUG}}$ du registre de configuration CONFIG4L du PIC à 0 afin de valider ce mode de fonctionnement ;
- câbler sur l'application quasiment le même dispositif que celui présenté précédemment pour la programmation en circuit car ce mode de fonctionnement exploite $\overline{\text{MCLR}}$, RB6 et RB7 et doit donc pouvoir en prendre le contrôle. Seule la tension V_{DD} reste celle de l'application puisque cette dernière doit fonctionner normalement.

Dans ces conditions, il est alors possible de relier l'interface matérielle du *debugger* en circuit à votre application au moyen d'un connecteur similaire par exemple au connecteur « intelligent » de la figure 17.2 (sans les contacts V_{DD} et RB5) afin de procéder à la mise au point de votre programme en utilisant le logiciel fourni.

Attention ! Le *debugger* Microchip ICD2 est prévu pour travailler avec l'environnement de développement MPLAB de Microchip, ce qui est particulièrement confortable puisque l'on dispose ainsi, au travers de ce dernier, de toute la chaîne de production de programme : édition, assemblage et/ou compilation, simulation et mise au point ou *debugging* en circuit. Ce n'est pas toujours le cas de certains produits concurrents du commerce qui utilisent parfois leur propre logiciel. C'est donc un élément à vérifier lors de l'achat d'un tel outil.

Ceci étant, l'utilisation de ce mode impose quelques restrictions mineures au niveau du PIC. En effet, d'une part les lignes de port RB6 et RB7 sont réservées par le *debugger* en circuit, d'autre part ce dernier nécessite un programme, certes de petite taille, mais qui doit tout de même être placé en mémoire de programme du PIC conjointement à votre application.

Lorsque l'on exploite ce mode de fonctionnement, il est donc possible de dresser la liste des restrictions que voici :

- les lignes RB6 et RB7 ne peuvent pas être utilisées par l'application ;
- la pile dispose de deux niveaux de moins car il faut les réserver pour le programme de gestion du *debugger* ;

- la première instruction de « votre » programme, c'est-à-dire celle placée à l'adresse 0000 doit être un NOP ;
- les 512 derniers octets disponibles en mémoire de programme doivent être laissés libres car c'est là qu'est implanté le programme du *debugger* ;
- 10 octets de la mémoire vive de données sont utilisés.

17.5 Les circuits avec un port ICSP et ICD dédié

Sur certains circuits de la famille PIC 18 utilisant un boîtier disposant de nombreuses pattes (boîtiers CMS de type TQFP ou similaire en général) certaines lignes sont spécifiquement dédiées à la programmation ICSP et au mode ICD ce qui permet de libérer $\overline{\text{MCLR}}$, RB6 et RB7 qui sont alors librement utilisables pour l'application. Elles sont présentées tableau 17.1.

Tableau 17.1 – Noms et fonctions des lignes de port dédiées à la programmation ICSP ou au mode ICD.

Mode « normal »	Mode port dédié	Fonction
$\overline{\text{MCLR}}$/VPP	$\overline{\text{ICRST}}$/ICVPP	Reset et passage en mode programmation
RB6/PGC	ICCK/ICPGC	Horloge série (ICSP et ICD)
RB7/PGD	ICDT/ICPGD	Données série (ICSP et ICD)

Pendant le fonctionnement normal de l'application, ces trois lignes n'interfèrent en aucun cas avec les lignes $\overline{\text{MCLR}}$, RB6 et RB7 qu'elles remplacent. Elles ne prennent ces fonctions que lorsque le circuit passe en mode programmation.

Même sur les boîtiers où elles sont présentes, ces trois lignes doivent cependant être explicitement validées par programmation du bit ICPRT du registre de configuration CONFIG4L (voir chapitre 3).