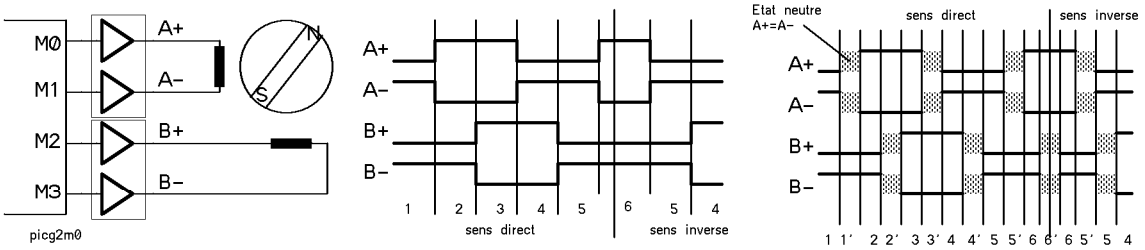
CH-1092 Belmont, Switzerland
info@didel.com, www.didel.com

1. Commande du moteur pas-à-pas switec

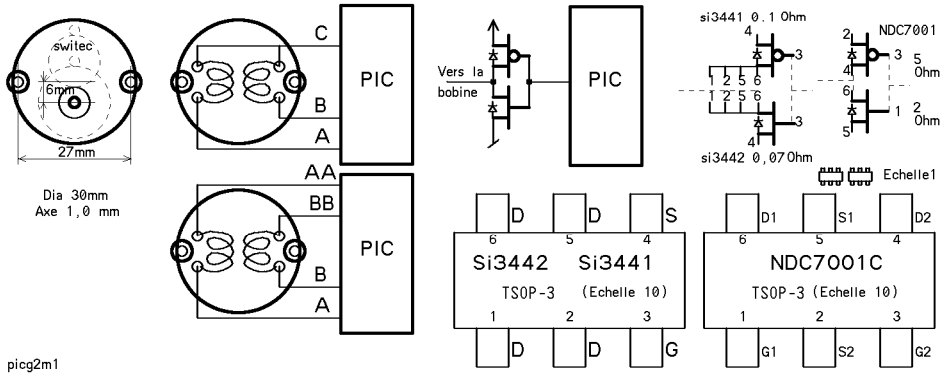
Les moteurs pas-à-pas ont 4 phases avec 2 bobines, ou 6 phases avec 2 bobines. Rappelons ici seulement le principe des moteurs à 4 phases, commandés en pas ou demi-pas. Par rapport aux moteurs à 6 phases expliqués plus loin, il suffit de changer les tables. Ces moteurs sont en général commandés par une tension de 12 Volts ou plus. Ils nécessitent donc une interface de commande appropriée; jusqu´à 12V, le circuit Si9986 convient bien.



picg2m0 *Fig. 1 Moteur 4 phases en pas et demi-pas*

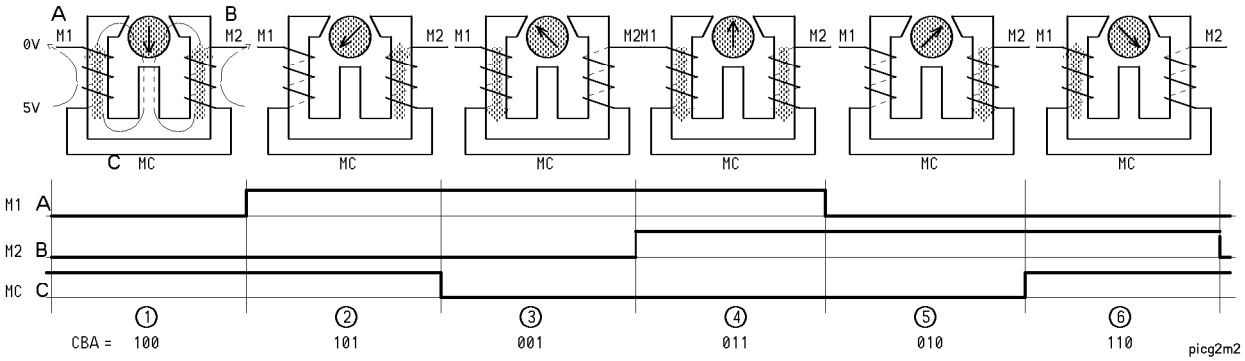
1.0.1. switec

Le moteur switec (www.switec.com) est un moteur à deux bobines équivalent à un moteur synchrone à trois phases. Chaque bobine a une résistance de 290 Ohm, soit une consommation de 17mA à 5V. Le moteur peut être connecté directement sur les sortie d´un PIC sans transistors amplificateurs, avec une perte de puissance de 10 à 30%, selon que le moteur est commandé par 3 ou 4 sorties du PIC (fig 1). Des transistors Mos complémentaires peuvent être utilisés en dessus de 3V. Le paramètre important est la résistance interne des transistors (5% de la résistance des bobines pour ne pas perdre plus de 5%). Des diodes de protection sont indispensables lorsque la charge est inductive et la tension élevée. Des transistors en boitier miniature sont documentés dans la figure 1. On peut naturellement utiliser des transistors dans des boitiers plus gros (BF170, BSS84 ou équivalent). Il n´y a pas avantage à utiliser des transistors bipolaires, qui ont des chutes de tension plus importantes et nécessitent des résistances sur les bases.



picg2m1 *Fig. 2 Câblage du moteur switec*

Avec les moteurs switec, un pas peut être exécuté toutes les ms, et on peut accélérer le moteur jusque vers 0.6 ms par pas. Le couple est maximum et constant à partir de 1,2 ms par pas (environ). Le switec ACC a une réduction 60 pour l´aiguille des minutes et naturellement une division supplémentaire par 12 pour les heures. Le moteur switec M-S, utilisé dans les robots **swibot**, a une réduction unique par 180.



picg2m2 *Fig. 3 Principe de fonctionnement du moteur switec*

Le moteur peut s´arrêter sur chaque pas si la tension est maintenue. Les moteurs switec ACC ont une position d´équilibre tous les demi-tours (positions (1) et (4)), car ils sont prévus pour une horloge alimentée par pile, et il faut minimiser la consommation moyenne de puissance. On peut couper le courant lorsque le rotor est (1) ou (4), ce qui est naturellement essentiel pour une montre, qui ne consomme ainsi que pendant 6 ms toutes les secondes.

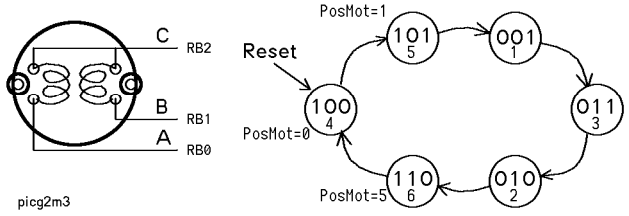
Le moteur miniature de montre ETA bidirectionnel se programment exactement de la même façon que le moteur switec.

Il y a de très nombreuses façons de programmer un moteur pas-à-pas, selon le nombre de moteurs et l´application. Avec un pas par milliseconde, le processeur peut faire beaucoup de choses en parallèle, s´il est bien programmé.

1.0.2. Commande simple d´un moteur

Les états successifs du moteur sont pris dans une table. Le pointeur PosMot correspond à la position angulaire du moteur. Il prend les valeurs 0 à 5. Ce pointeur est incrémenté après chaque pas. Quand il arrive à 6, il est remis à zéro. Il faut être attentif aux différentes numérotations utilisées. Les pas ont été repérés 1.6 dans des cercles, figure 9; Les valeurs binaires correspondant au code généré sur le port B ont des équivalents décimaux dans cette figure. Le pointeur qui parcourt la table de valeurs a les états 0 à 6. Quand il arrive à 6, on le force à zéro pour parcourir circulairement la table.

Une fois que l´on sait faire un pas, on peut compter les pas ou les tours, ou les millimètres d´avance du robot après un calibrage adéquat.



picg2m3 *Fig. 4 Séquence de commande*

La partie de programme suivante fait avancer le moteur d´un pas. Le moteur est connecté sur les trois bits de poids faible du port B. La table TaMot contient les positions successives du moteur. Une macro allège son écriture. Si le moteur est câblé sur 4 bits, la table est facilemment adaptée. Si le moteur doit tourner dans l´autre sens, il suffit d´inverser 2 colonnes, ou toute la table.

Program **Moteur switec** Un sens, vitesse selon délai entre l'appel du module Pas:

```
; Initialisation:
      Clr PosMot
      Call        Tamot
      Move        W,PortB       ; Position initiale
                                ; (attendre 5ms pour la
.....
; Module exécutée chaque milliseconde ou plus
 Pas: Move        PosMot,W
      Call        Tamot
      Move        W,PortB       ; Avance d'un pas
      Inc         PosMot
      Move        #6,W
      Sub         PosMot,W      ; Comparaison: =6?
      Skip,NE
      Clr         PosMot
```

Table **TaMot** Moteur switec

```
.macro            b             ; data pour tables
      RetMove     #2'%1,W
.endmacro
TaMot:                          ; Avance d'un pas
      Add         W,PCL
      b 00000100  ; 0
      b 00000101
      b 00000001
      b 00000011
      b 00000010
      b 00000110  ; 5
```

.....On remarque que tous les autres bits du portA sont mis à zéro. Si ces bits sont des entrées, ce n'est pas gênant. S'il y a des sorties, il faut masquer et superposer comme ci-dessous (voir aussi la section 2.2).

```
      Call        Tamot
      Move        W,SaveW
      Move        PortB,W
      And         #2'111,W
      Or          SaveW,W
      Move        W,PortB
```

La correction, quand on arrive en fin de table peut être faite de façon astucieuse en mettant dans la table l'instruction de remise à zéro, ce qui est possible en fin de table. S'il faut compter les tours, cela pourrait aussi se faire à ce moment. A noter que maintenant la table est décalée de 1; quand le pointeur est utilisé, il vaut de 1 à 6.

Program **Moteur switec** Un sens, vitesse selon délai entre l'appel du module Pas:

```
; Initialisation:
      Clr PosMot
      Call        Tamot
      Move        W,PortB       ; Position initiale
.....
; Module exécutée chaque ms ou plus
 Pas: Move        PosMot,W
      Call        Tamot
      Move        W,PortB
.....
```

Table **TaMot** Moteur

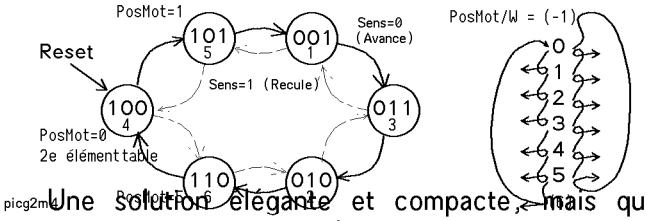
```
TaMot:                          ; Avance circulairement
      Inc         PosMot
      Add         W,PCL
      b 0         ; Position inutilisée (Macro b: voir exe
      b 00000100
      b 00000101
      b 00000001
      b 00000011
      b 00000010
      Clr   PosMot
      b 00000110
```

1.0.3. Commande bidirectionnelle

Si le moteur doit tourner dans un sens ou l'autre selon un bit bSens dans une variable Mode, le plus simple est d'appeler deux tables. Le pointeur PosMot est incrémenté ou décrémenté selon bSens. Le passage de 0 à 5 lorsque le moteur tourne en sens inverse est fait en rajoutant 6 lorsque l'on arrive à zéro. On peut imaginer de nombreuses variantes pour cette table bidirectionnelle. La meilleure sera la plus courte en nombre d'instructions et temps d'exécution, avec le même temps d'exécution que l'on tourne dans un sens ou dans l'autre. Comme les pas sont de 1 ms, on peut toutefois admettre plusieurs microsecondes de différence.

Program **Picgw1** **Moteur switec bidirectionnel, selon bit bSENS dans registre Mode**

```
bSens = 3
; Initialisation:
      Clr PosMot
      Call        NextStep
      Move        W,PortB       ; Position initiale
...
; Changement de sens
      Modification du sens selon un poussoir sur PortA:#
      TestSkip,BS   PortA:#bSw0
      Clr         Mode:#bSens
      TestSkip,BC   PortA:#bSw0
      Set         Mode:#bSens
....
; Module pas
      Call        NextStep      ; Tient compte du sens
      Move        W,PortB       ; Prochain pas
.....
```



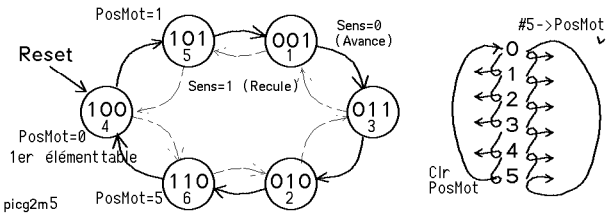
Routine **NextStep** **Prépare pas suivant selon bSens**

```
NextStep:               ; Avance ou recule circulairement
      Move        PosMot,W
      TestSkip,BC   Mode:#bSens
      Jump        Back$
      Inc         PosMot
      Add         W,PCL
      b           0     ; A cause de Inc avant l'addition
      b 00000100
      b 00000101
      b 00000001
      b 00000011
      b 00000010
      Clr   PosMot
      b 00000110
Back$:
      Sub         #1,W
      Skip,CS             ; Si PosMot=0, 0+(-1) =25
      Add         #6,W    ; 6+(-1)=5
      Move        W,PosMot
      Add         W,PCL
      b 00000100
      b 00000101
      b 00000001
      b 00000011
      b 00000010
      b 00000110
```

Une solution élégante et compacte, mais qui a le défaut d'avoir une durée d'exécution variable entre 6 et 9 μs (ce qui n'est pas gênant pour un moteur pas-à-pas) est de mettre la direction dans le pointeur PosMot, et de doubler cette valeur au moment de l'accès dans la table, pour coder dans la table la modification du pointeur. La table est plus longue, mais l'exécution plus rapide. Ce type de programmation est typique du PIC et montre la flexibilité de programmation donnée par les tables. Mais évidemment, il faut compter précisément les bits et mots, et vérifier sur le listage que le code assemblé correspond à ce que l'on attend.

Program **Picgw2** **Moteur switec bidirectionnel, selon bit bSENS dans registre PosMot, bit 3**

```
; PosMot  00000xxx sens positif  00001xxx sens invers
bSens = 3       ; Ne pas changer, lié à la structure d
; Initialisation:
      Clr PosMot
      Call        NextStep
      Move        W,PortB       ; Position initiale
;...
; Changement de sens
;     Modification du sens selon un poussoir sur PortA:#
      TestSkip,BS   PortA:#bSw0
      Clr         PosMot:#bSens
      TestSkip,BC   PortA:#bSw0
      Set         PosMot:#bSens
;....
; Module pas
      Call        NextStep      ; Tient compte du sens
      Move        W,PortB       ; Prochain pas
;.....
```




```
.Macrob
RetMove           #2'%1,W
.Endmacro
```

Routine **NextStep** **Prépare pas suivant selon bSens**

```
NextStep:               ; Avance ou recule circulairement
      RLC         PosMot,W      ; Double sans modifier P
      Add         W,PCL
      Inc         PosMot        ; +0
      b 00000100
      Inc         PosMot        ; +2
      b 00000101
      Inc         PosMot
      b 00000001
      Inc         PosMot
      b 00000011
      Inc         PosMot        ; +8
      b 00000010
      Clr   PosMot  ; +10
      b 00000110
Cor$: Move        #2'1101,W     ; 12
      Move        W,PosMot
      b 00001100 ; +14
      b 0         ; +15 remplissage
Back$:                          ; +16
      Jump        Cor$          ; On prépare 5 avant de
      b           0             ; remplissage
      Dec         PosMot        ; +18
      b 00001101
      Dec         PosMot
      b 00001001
      Dec         PosMot
      b 00001011
      Dec         PosMot
      b 00001010
      Dec         PosMot        ; +26 = 2**bSens+2*5
      b 00001110
```

1.0.4. Commande simultanée de plusieurs moteurs

Sur un port on peut mettre deux moteurs, mais ils ne tourneront pas nécessairement à la même vitesse. En appliquant plusieurs fois les routines ci-dessus, et en copiant les bits dans les bonnes parties des registres SavePortA, SavePortB (section 2.2), on s'adapte au schéma de câblage en quelques instructions. Un câblage astucieux permet parfois d'économiser des instructions.

1.0.5. Vitesse variable

La vitesse du moteur pas-à-pas peut dépendre d'une variable proportionnelle à la période ou d'une variable proportionnelle à la vitesse, comme cela a été vu en section 2.5.

Une boucle d'attente entre chaque pas définit la période. Elle est simple à comprendre et programmer. La variable PeriodePas exprime par exemple cette période en unités de 50 microsecondes, qui correspond à des instructions que l'on va faire pour s'occuper aussi des éléments autres que notre moteur. La vitesse maximum correspond à la valeur 20, soit 20 boucles de 50 μs (1 ms), la vitesse minimum, valant 255, est 12 fois plus lente. Il faut définir une variable 16 bits (sur 2 mots) pour avoir des vitesses aussi lentes que 1 pas toutes les 3 secondes.

Avec une commande en vitesse, on ajoute à la variable C1 de l'exemple de la section 2.5, la variable vitesse, valant de 0 à 25 au maximum (si les tâches à faire régulièrement prennent 50 μs). Avec la vitesse 1, un Carry est généré tous les 256 fois 50 μs, soit tous les 0,013 sec (77 Hz). La condition "Carry Set" apelle la routine qui calcule le pas suivant, et l'axe du moteur switec tournera à 77Hz divisé par 6 pas par tour et 180 tours du rotor par tour de l'axe, soit un tour en 14 secondes. Si la vitesse est de 25, le Carry se produira toutes les 1 ms, la vitesse est maximale (2 tours par seconde) et un robot avec des roues de 20mm avancera à 12cm/s. La vitesse 0 est très pratique, puisqu'elle passe par les mêmes routines. Des vitesses négatives se gèrent assez facilement.

On verra ultérieurement avec la programmation synchrone (3e partie) comment gérer simultanément le moteur et d'autres tâches.