CH-1092 Belmont, Suisse
Tel +41 21 728-6156, Fax 728-6157
Email info@didel.com, www.didel.com

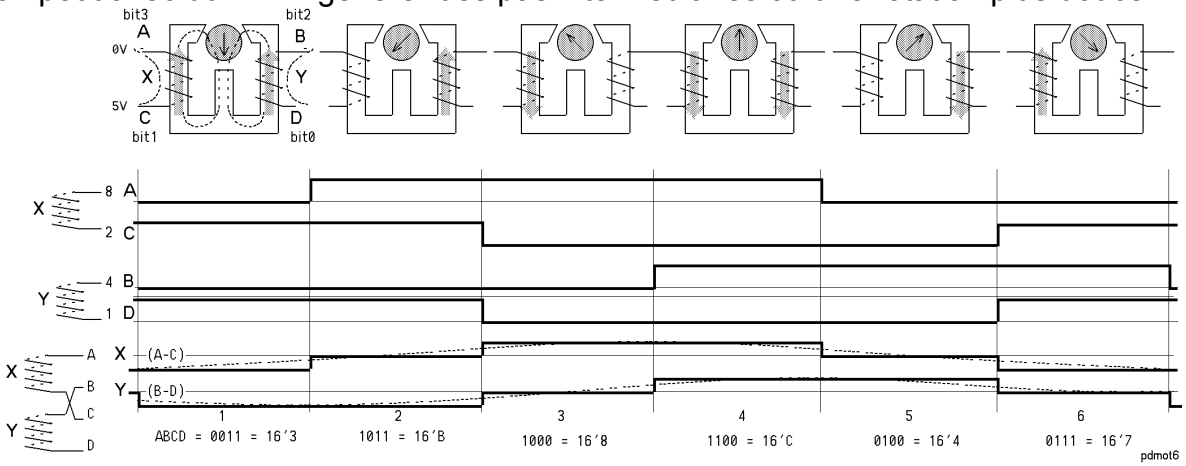
DidelBot – Comprendre les techniques informatiques

Fun 4 – Moteurs pas-à-pas

Ce module suppose que l'étude du module introduction a été bien compris. Il est recommandé mais pas indispensable de jouer avec le module 2 (Sons.doc) avant. Il exerce les boucles d'attentes et montre comment commander les moteurs pas à pas du robot et effectuer des déplacements complexes..

4.1 Principe de fonctionnement

Les moteurs du Didelbot sont des moteurs horloger (typo Lavet) à très faible consommation de puissance. Ils sont utilisés dans les tableaux de bord des voitures comme indicateur de vitesse, horloge, etc. Leur documentation peut se trouver sur le site www.vid.wellgain.com.
Le rotor est un aimant orienté par le champ de 2 bobines. Les fils sont reliés sur le port B du microcontrôleur. Le séquencement correct des 4 signaux crée un champ tournant et on peut avec du PWM générer des pas intermédiaires ou une rotation plus douce.



Les bits 0 – 2 et 1 - 3 du portD commandent le moteur gauche et le bits 4 - 6 et 5 - 7 le moteur droit. La séquence hexa 3 – B – 8 – C - 4 – 7 correspond aux 6 excitation actives et ont été déclarées dans le fichier _Base.asi sous les noms _Pos1 .. _Pos6.

Il faut donc envoyer au port B ces valeurs dans l'ordre (ou l'ordre inverse pour tourner dans l'autre sens) avec un temps d'attente entre chaque valeur dépendant de la vitesse de rotation voulue.
La vitesse maximale à vide est de 3 tours par seconde. La réduction est de 180, donc la période minimale pour un pas est de 0.33s divisé par 180 et par 6, soit 300 microseconds. Le processeur a donc le temps de calculer chaque pas. Un programme naif écrirait

```
UnTour:   Move  #_Pos1,W
          Move  W, PortB
          Call  Delai
                ….
          Move  #_Pos6,W
          Call  Delai
          Jump  UnTour
```

.

Utilisons plutôt une table avec un pointeur qui incrémente circulairement, quant il arrive à 6 il recommence. Les instruction proposées pour calculer un pas (on peut imaginer différentes solution) sont les suivantes:

```
StepAv:
      Inc   PtStep
      Move  #-6,W
      Add   PtStep,W
      Skip,NE
      Clr   PtStep
      Move  PtStep,W
      Add   W,PCL
      tt    _Pos1
      tt    _Pos2
      tt    _Pos3
      tt    _Pos4
      tt    _Pos5
      tt    _Pos6
```

Les trois premières instructions testent si PtStep pointe sur _Pos6, la dernière valeur valeur envoyée. Si l'addition de –6 donne zéro, on met PtStep à zéro. Le pointeur est ensuite incrémenté dans tous les cas pour aller chercher la position suivante à donner au moteur. Add W,PCL saute à la position qui contient un RetMove #Posi,W. Une macro tt a été définie pour que cela soit plus facile à écrire. La routine ci-contre rend la position, envoyée ensuite sur le PortB. La variable PtStep doit être initialisée à une valeur entre 0 et 6.
Une boucle d'attente de plus de 300 microseconde se fait en appelant la routine _Delai100us qui existe dans le processeur et dont l'adresse est déclarée dans _Base.asi. Le programme complet est **r41a.asm**.

```
StepAr:
      Move  #6,W
      Test  PtStep
      Skip,NE
      Move  W,PtStep
      Dec   PtStep
      Add   W,PCL
      tt    _Pos1   etc.
```

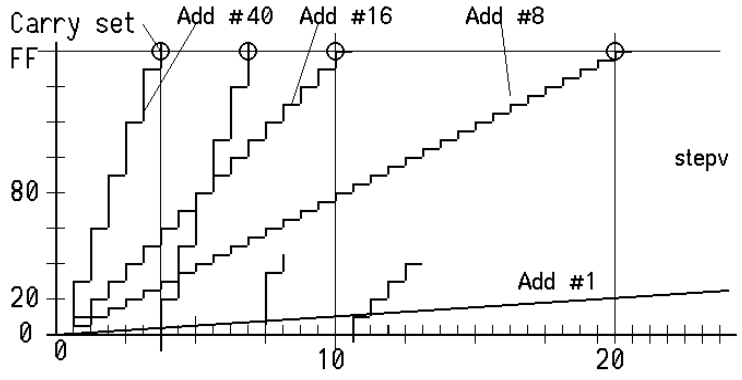
Pour reculer, il faut parcourir la table en sens inverse. Quant on arrive à zéro, on force la valeur 6 avant de décrémenter et passer dans la table.Le programme de test est **r41b.asm**. Agissez sur le délai pour vérifier le maximum à vide (roues en l'air) et sur le sol.
Le programme **r41c.asm** fait 256 pas dans un sens et 256 dans l'autre. Il suffit de compter.

Le moteur droit commande les 4 bits de poids fort. Un Swap suffit pour faire passer l'information lue dans la table du bon côté du refiste. On remarque alors que le moteur tourne en sens inverse (en fait le schéma a été défini pour qu'il tourne dans le même sens, mais il y a symétrie centrale ente gauche et droite!)
Le programme de test est **r41d.com**

4.2 Commande en vitesse

Les boucles d'attente ne permettent pas de commander les deux moteurs à des vitesses différentes. Rien n'est du reste si peu satisfaisant que ces boucles d'attente qui bloquent le processeur à ne rien faire d'utile.

L'idée est d'ajouter une valeur à un compteur, et de faire un pas chaque fois que le compteur déborde, donc le Carry est à 1. La valeur additionnée correspond à une vitesse, ce qui par ailleurs est très satisfaisant: si on additionne 0 (vitesse nulle) il n'y a jamais de Carry, donc de pas.



Si on additionne 255, il y a carry 255 fois sur 256 et la période entre pas est minimale (selon le nombre d'instructions dans la boucle).

Les instructions sont très simples. La variable Cnt additionne le paramètre de vitesse, et un pas est effectué toute les fois qu'il y a dépassement (Carry=1). Pour que les pas ne soient pas trop rapide, il faut ajouter une boucle d'attente qui évite de travailler avec des vitesse trop petites, donc avec des différences importantes entre chaque vitesse.

```
Bcle:     ; Attente 100us
     Move  #30,W; Vitesse
     Add   W,Cnt
     Skip,CS
     Jump  Bcle
     Call  StepAv
     Move  W,PortB
```

Le programme **r42a.asm** fait tourner le moteur gauche

Il est maintenant très facile de faire tourner les moteurs droite et gauche à des vitesses différentes. Le programme **r42b.asm** assigne des vitesse 5 et 20, avec un délai supplémentaire de 100 us dans la boucle. Que va-t-il se passer si le délai est de 200 us? Déterminer la vitesse maximale, est-elle la même pour les deux moteurs (à vide, robot posé sur le côté). Est-ce que la vitesse minimale est bonne (il est inutile qu'elle soit trop lente au détriment d'une bonne continuité dans les vitesses rapides).

La prochaine étape est naturellement de pouvoir donner des vitesses négatives. Il suffit de tester le signe, inverser les valeurs négatives, appeler les tables correspondantes. Lire attentivement le programme **r42c.asm** avant de tester quelques valeurs de paramètres.

4.3 Solution par interruption

Le programme moniteur (ordre Hexo nnV par exemple) gère les deux moteurs avec des routines similaires, un peu plus complètes en fait car elles mesurent la distance (nombre de pas). Le but n'est pas ici d'apprendre à les utiliser. Contentons-nous comme pour l'ordre V de mettre en route les deux moteurs. Il suffit pour cela d'activer les interruptions, autoriser l'exécution du module de commande de moteur, et assigner la vitesse.

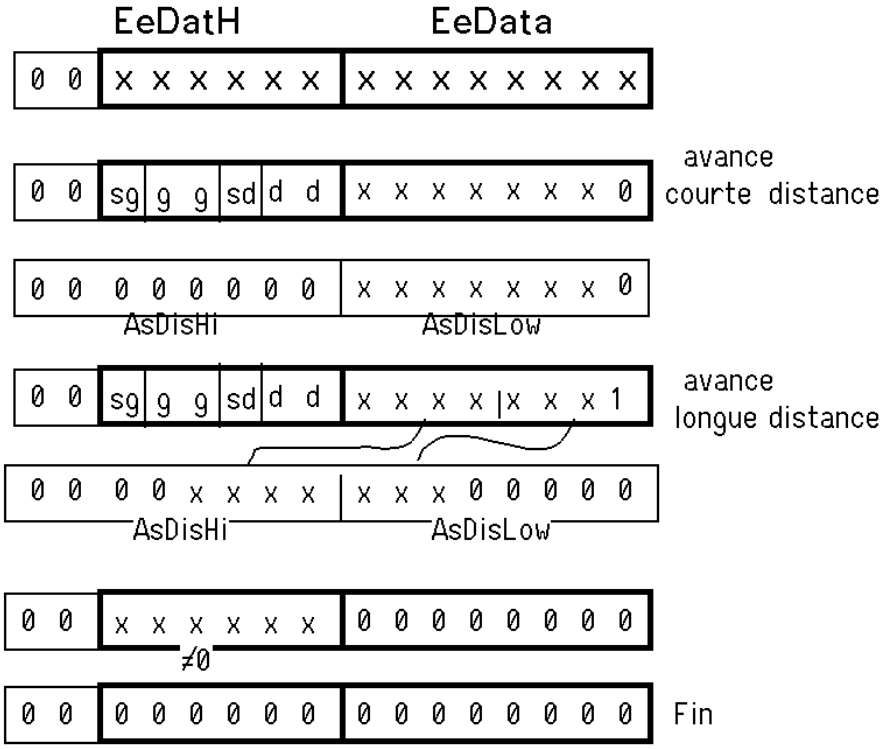
```
_IOn
_StepOn
Move  #-20,W
Move  W,_VitG
Move  #-20,W
Move  W,_VitD
Jump  APC
```

Le programme **r43a.asm** n'a que des importations en plus. Toute la complexité est cachée dans le système; c'est l'intérêt des routines systèmes.
On peut facilement compliquer le programme pour faire évoluer les vitesses, ou tenir compte des distances parcourues, en utilisant les variables **_Distlow**, **_DistHi**, **_Angle**. **_DistHi** et **_DistLow** correspondent à la somme des pas effectués par les deux moteurs. **_Angle** correspond à la différence. Les ordres Hexo Y, X, T lisent ces trois valeurs.

De plus, ***_AsDistHi et*** **_AsDistLow** décomptent et en activant **_Mode:#_bAvance.**
On peut tester le passage à zéro de **_bAvance** quand la distance est nulle.
Par exemple pour dessiner un carré, on répète 4 fois une avance suivie d'un tour de 90 degrés. Les ordres Hexo U, I, J, K utilisent ces primitives.
Il faut définit la vitesse et la distance à parcourir et activer _bAvance. Le programme **R43b.asm** montre comment dessiner un carré.

4.4 Dessin

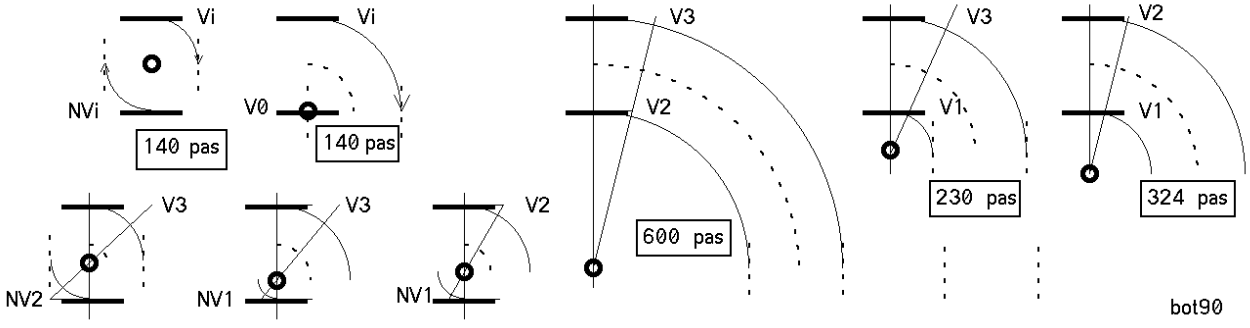
Allons plus loin dans la simplicité pour définir un dessin, au détriment d'une routine Trajet dans le moniteur, qui interprète les ordres alignés en mémoire.
L'idée a été détaillée dans Sons.doc. La mémoire contient des mots de 14 bits que l'on peut lire. Définissons les instructions de déplacement suivantes:



On voit que l'on peut coder la distance parcourue (somme des impulsions sur les moteurs, ne tenant pas compte du sens) entre 0.34mm et 6 mètres en ligne droite. On remarque le truc pour avoir à la fois des distances courtes et longues: les distance paires sont courtes, les impaires sont multipliées par 16.

2 → 0.34mm
100 → 17 mm
7 → 16.3mm (6 x 16 x 0.17)
101 → 272mm
254 → 43 mm
255 → 690 mm

En virage c'est plus compliqué de déterminer l'angle, qui dépend des vitesses des 2 moteurs. Si le robot tourne sur lui-même, son empattement étant de 30.5mm, 90 degrés correspond à 140 pas (à ajuster expérimentalement étant donner l'imprécisions du 30.5mm et du 0.17mm.



Rappelons que l'on peut jouer avec le moniteur Hexo (Serie.doc) pour vérifier ces valeurs, mais lorsque le robot dessine un carré, les erreurs sont très visible.

Avec seulement 3 bits pour chaque vitesse il a fallu définir une table de correspondance pour les 7 vitesses logiques:

```
_V0    vitesse nulle
_V1    vitesse 16'8  = 8
_V2    vitesse 16'14 = 20
_V3    vitesse 16'20 = 32
_NV1   vitesse 16'F8  =  -8
_NV2   vitesse 16'EC = -20
_NV3   vitesse 16'E0  = -32
```

Une macro appelée **_Av** facilite la génération du mot de 14 bits à partie des 3 paramètres qui définissent un mouvement: VitesseDroite, VitesseGauche, Distance. Ainsi, pour dessiner un carré de 34mm de côté, on code

```
Carre: _Av    _V3,_V3,200
       _Av    _V3,_NV3,140
       _Av    _V3,_V3,200
       _Av    _V3,_NV3,140
       _Av    _V3,_V3,200
       _Av    _V3,_NV3,140
       _Av    _V3,_V3,200
       _Av    _V3,_NV3,140
       _Fin
```

L'exécution ne peut pas être plus simple

```
Bcle:  Move  #Carre,W
       Call    _Trajet
       Jump  0       ; si on ne veut pas répéter
       Jump  Bcle    ; si on veut répéter sans cesse
```

A noter que si on répète, le carré est défini par 2 lignes, ce qui est fait dans le programme complet **R44a.asm.**

A vous de jouer pour faire des étoiles, écrire votre nom, optimiser la vitesse de dessin en sessinant en arrière si cela réduit les angles. Vous avez de la place pour 220 déplacement élémentaires. Courage poue en taper autant!
Dans les démos en mémoire processeur, la demo 3 modifie la vitesse d'un des moteurs dans une boucle d'attente. Cela ne peut pas se faire en appelant la routine Trajet. Il faut programmer comme R43a et R43b.
Les demos 4 et 5 utilisent Trajet et sont codés comme suit si vous voulez les modifier:

```
Des4: Av V2,V2,  21       ; côté 55mm
      Av NV1,V2, 21       ; arc 190 degrés
      Av V2,V2,  21       ; côté 55mm
      Av NV2,V2, 250      ; tourne sur lui-même
      Fin

Des5: Av V3,V2,  25       ; arc 60mm
      Av NV2,V2, 20       ; petit angle
      Av NV3,NV2, 25      ; recule arc 60mm
      Av V2,NV2, 100      ; tourne sur lui-même
      Fin
```