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

http://www.didel.com/pic/Compati200.pdf

Transferts 16F877 – 10F200

Le but est d'écrire pour le 10F200-206 un programme testé sur le 16F877 ou 16F630

Taille mémoire On écrira en fin de programme 16F877

```
.If      .Not. T200 ; aussi 202
 .If     APC .GE. 16'100
\b ;Attention - programme trop long
 .Endif
.Endif
```

```
.If      .Not. T204 ; aussi 206
 .If     APC .GE. 16'200
\b ;Attention - programme trop long
 .Endif
.Endif
```

Ce message est un avertissement. Il se peut que le code 877 soit plus long que le code 200 pour faciliter le déverminage. L'assembleur vérifie la longueur du code selon le .Proc inséré (à vérifier).

Taille des variables On écrira en fin de programme de déclaration des variables

```
.If      .Not. T200 ; aussi 204 206
 .If     APC .GE.  DebVar+16'10
\b ;Attention  trop de variables
 .Endif
.Endif
```

```
.If      .Not. T202 ; aussi  206
 .If     APC .GE.  DebVar+16'18
\b ;Attention  trop de variables
 .Endif
.Endif
```

Il y a plus de variables pour le 10F202 et 10F206, mais elles commencesnt pls tôt (valeur DebVar selon le .Ref 10F20x). Les variables propres au déverminage sous 877 seront placées à la fin dans un .If T877 .. .Endif.

I/O port 6 (GP = RB)

Le port est à l'adresse 6, comme le portB d'autres processeurs. Il es appelé portB. RB3 est en entrée seulement. Pendant la programmation il recoit des impulsions de 12V et ne doit pas être connecté à un signal de faible résistance interne on ne supportant pas la tension.
Pour les autres bits du ports, et pour trouver on bit de même comportement sur le 877, voir la documentation. Il n'y a pas de comparateur sur le 877, la mise au point pourra se faire avec un 630.

Instructions manquantes et modifiée – Ret et Call

Il manque 3 instructions sur le 10F20x et le Call est limité sur les 204/206
L'instruction Ret n'existe pas. Il faut écrire RetMove #0,W qui modifie W. Il ne faut donc pas passer des paramètres de retour par W et savoir qu'un Call modifie W.
Avec les 10F204 et 206, qui n'ont que 512 mots de mémoire programme, on doit mettre les débuts de routines dans les adresses 0 à 255 (adresse 8 bits).
Pour conserver l'instruction Ret dans les programmes, on écrira

```
.If  T200
  .Macro  Ret
        RetMove  #0,W
  .Endmacro
.Endif
```

Attention : il n'y a que 2 niveaux de routines!! l'appel à une table (instruction Add W,PCL) fait aussi un niveau.

Instructions manquantes - Add,Sub immédiat

Les instructions Add #Constant,W et Sub W,#Constant,W n'existent pas dans les Pic avec des instructions de 12 bots de long, comme le 10F20x. C'est facile à convertir, et dans les cas simple, le programme n'est pas plus long. Il faut donc apprendre à programmer pour un 10F et il n'y a pas besoin de prévoir la compatibilité.

On replace	par
Move Variable,W Add #Constant,W	Move #Constant,W Add Variable,W

Move Variable,W
Sub W,#Constant,W
	Move #Constant,W
	Move W,Temp
	Move Variable,W
	Sub W,Temp,W
	ou
	Move #-Constant,W
	Add Variable,W

Sub W,#0,W (complément à 2, instruction NEG W)	Clr Temp Sub W,Temp,W

Interruptions et registre IntCon

Les 10F200 n'ont pas d'interruptions, le registre **Intcon** a été supprimé et le flag **TmrIF**, qui indique que le Timer0 a débordé, n'existe pas. Il faut regarder si le Timer passe par zéro avec l'instruction Move TMR0,W, mais attention, il ne vas pas rester à zéro. Il faut donc éviter de surveiller le passage par zéro, ou le faire dans une boucle plus petite que le nombre de cycles faisant avancer le timer.

Par exemple, pour faire une opération toutes les 100 microsecondes (qui dure moins de 92 us) on peut écrire

```
A$:   Move Tmr0,W
      Skip,EQ
      Jump  A$
      Move  #256-(100/4)+2,W   ; 100 \mu;s, prediviseur par 4
      Move  W,Tmr0
```

La boucle A$ utilise 4 cycles. Ne pas prédiviser par 2, le passage par zéro ne sera pas vu une fois sur deux.

Saturer une table est indispensable pour éviter des plantées.

Sur un 16Fxx on écrit d'habitude

```
...
      Call   TaDur
      ...
TaDur:
      Move  Var,W
      Sub   #Max,W
      Skip,CC
      Clr   W
      Add   #Max,W
      Add   W,PCL
A:    RetMove     #…,W
 ….
B:    RetMove     #…,W
 Max  = B-A ; longueur de la table
```

Sur un 12F50x, c'est tout aussi rapide d'écrire

```
...
      Call   TaDur
      ...
TaDur:
      Move  #Max,W
      Sub   W,Var
      Skip,CC
      Clr   Var
      Add   Var,W
      Move  W,Var ; si Var doit être rétabli
      Add   W,PCL
A:    RetMove     #…,W
….
B:    RetMove     #…,W
 Max  = B-A
```