J.D. Nicoud, Mouette 5
CH-1092 Belmont, Switzerland
Tel +41 21 728-6156, Fax 728-6157
Email info@didel.com, www.didel.com

DIDEL

Fichier www.didel.com/picg/doc/DocPage.pdf

0.94

Programmes multi-page

Les PICs 14 bits n´ont que 11 bits pour l´adresse dans les instructions Jump et Call. Au-delà de l´adresse 16´7FF, il faut donc comprendre comment commuter de page et tenir compte de ce que sait faire l´assembleur. La page 0 va de 0 à 7FF, la page 1 de 800 à FFF, etc. Seuls les processeurs 16F873,874,876,877, 16F87/88, 16F73,74,76,77 ont assez de mémoire (4k ou 8k) pour présenter ces problèmes d´adressage.

A noter que l´on parle aussi de pages de 256 bytes (8 bits d´adresse, 16´100 en hexa) pour les tables accédées par l´instruction Add W,PCL. Le même registre PcLath contient les bits 8-13 de l´adresse, et ce registre insère automatiquement les bits qui manquent dans l´instruction. Pour l´instruction Add W,PCL, ce sont les bits 8-13; pour les sauts en page 1,2,3 qui nous préoccupent ici, ce sont les bit 12 (processeurs 4k) ou 12,13 (processeur 8k).

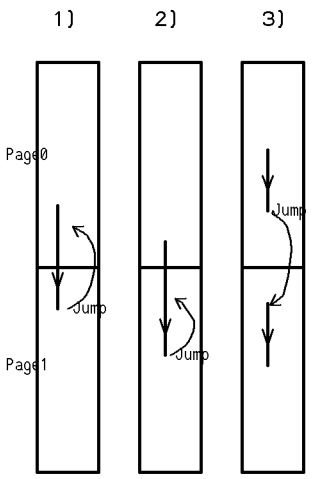
Les banques sont dans l´espace des variables et n´ont rien à voir avec les pages en mémoire programme.

1er cas Le programme déborde en page 1, et contient à la fin un saut en page 0. Cette situation est catastrophique; l´assembleur ne se rend pas compte du débordement; le code en 800 vient écraser le code en zero lors du chargement.

2e cas Le programme déborde en page 1, et contient un saut dans la page 1. L´assembleur va refuser ce jump, car l´adresse a plus de 10 bits et ne peut pas être mise dans le champ de l´instruction. En plus de cela, le problème du cas précédent subsiste.

3e cas Le saut se fait de la page 0 à la page1. La page 1 doit être préparée dans Pclath, pour que le bit d´adresse qui manque soit ajouté au moment de l´exécution du Jump. Il faut d´autre part masquer ce bit de trop pour l´assembleur.

```
Move    #16´800/16´100,W
Move    W,Pclath
Jump    Adr .AND. 16´7FF
```



Le Pclath n´a pas besoin d´être repréparé si l´on reste dans la même page, mais les adresses doivent être masquées. La valeur de APC est correcte, mais le code des instructions Jump et Call n´a pas la place pour ces bits que le processeur ira chercher dans Pclath.
Si l´on saute en page 0, il faut préparer Pclath, mais il n´y a pas besoin de masquer l´adresse.

Stratégies

Si on doit aller au-delà de la page 0, il faut bien définir ce qui est en page 0 et dans les pages suivantes, et s´astreindre à une discipline rigoureuse.

1) Le dernier module de programme prévu en page 0 doit être terminé par un

```
.If     APC .AND  16´800
  Aie, les instructions débordent la page 0
.Endif
```

Il faut surveiller de façon similaire le débordement à la fin des autres pages.

2) Les adresses de sauts dans l´une des pages 1,2,3 doivent être masquées pour n´avoir que 10 bits

3) Tous les sauts dans une page autre que la page courante doivent être précédés de l´initialisation du Pclath (ce n´est pas nécessairement l´instruction juste précédente).

On pourrait systématiquement annoncer la page avant les Jumps et Call, et écrire une macro qui raccourci l´écriture.

```
    Move    #Adresse/16´800,W  ; No de pa
    Move    W,PcLath
    Jump    Adresse.AND.16´7FF
Ceci modifie W, ce qui peut être gênant
```

```
    Move    W,SaveW
    Move    #Adresse/16´800,W  ; No de page
    Move    W,PcLath
    Move    SaveW,W
    Jump    Adresse.AND.16´7FF
```

Maintenant, c´est le fanion Z qui est modifié, selon la valeur de W (Move SaveW,W modifie F=Status). On sait sauver W et F comme pour les interruptions, mais cela fait 10 instructions pour chaque saut.

Pour ne pas modifier W et F, il est plus simple et plus efficace d´agir sur le bon bit

```
Set     Pclath:#11                  Clr     Pclath:#11
; Passe en page 1 si on est en page 0   ; Passe en page 0 si on est en page 1
```

Il est donc nécessaire de gérer les pages au mieux et d´utiliser des macros pour tous les Jump et Call en page supérieure à 1. Malheureusement, le .If n´accèpte pas un symbole qui n´est pas connu en première passe et la macro ne peut pas vérifier si tout est correct.

Macros JumpInPage JumpOutPage CallInPage CallOutPage

La macro est exécutée à une adresse APC avec comme premier paramètre (%1) l´adresse de saut.

```
.Macro  JumpInPage      ; On veut sauter dans la
        Jump    %1.AND.16´7FF
.Endmacro
```

```
.MacroJumpOutPage ; On veut sauter dans un autre page
        Move    #%1/16´800,W
        Move    W,PcLath
        Jump    %1.AND.16´7FF
.Endmacro
```

Les macros CallInPage et CallOutPage sont évidentes. On pourrait ajouter des macros JumpOutW, CallOutW qui sauvent W, ou séparer la préparation de la page, qui peut se faire avant que l´on prépare W pour le Jump ou Call, et le saut masqué.

Exemple: Fichier Pict873p

Program **Pict873p.asm** clignote

```
RA RB0..7 RC période 0.65 s
.Proc 16f877
.Ref  16F870
.MacroJumpInPage            ; On veut sauter dans la
        Jump    %1.AND.16´7FF
.Endmacro
.MacroCallOutPage ; On veut sauter dans une autre page
        Move    #%1/16´800,W
        Move    W,PcLath
        Call    %1.AND.16´7FF
.Endmacro
C1    = 16´20
C2    = 16´21
.Loc  0
Deb:
        Move    #0,W        ; sorties
        Move    W,TrisB
        Move    #16´55,W
Loop:
        Xor     #-1,W
        Move    W,PortB
; Attente
        CallOutPage  Delai
        Jump    Loop

.Loc  16´800
Delai:
  A$:DecSkip,EQ  C1
        JumpInPage  A$
        DecSkip,EQ  C2
        JumpInPage  A$
        Ret

JumpInPage Loop     ; Test, KO
  L´assembleur accèpte, mais Loop n´et pas dans la mê

JumpOutPage Loop    ; Test, OK
  Mais il faut se souvenir que W a été modifié

.End
```

```
0000
```

Program **Pict873p.asm** clignote

```
0000                        RA RB0..7 RC période 0.65
0000                        .Proc   16f877
0000                        .Ref    16F870
0000                        .Macro  JumpInPage  ; On
0000                                Jump    %1.AND.
0000                        .Endmacro
0000                        .Macro  CallOutPage  ; On
0000                                Move    #%1/16´
0000                                Move    W,PcLath
0000                                Call    %1.AND.
0000                        .Endmacro
0000    00000020  C1  = 16´20
0000    00000021  C2  = 16´21
0000    00000000        .Loc    0
0000                    Deb:
0000    0030                        Move    #0,W
0001    6600                        Move    W,TrisB
0002    5530                        Move    #16´55,
0003                    Loop:
0003    FF3A                        Xor     #-1,W
0004    8600                        Move    W,PortB
0005                    ; Attente
0005_   0130                        Move    #Delai/16
0006_   8A00                        Move    W,PcLath
0007_   0020                        Call    Delai.AND
0008    0328                        Jump    Loop
0009
0800    00000800 .Loc   16´800
0800                    Delai:
0800    A00B              A$:       DecSkip,EQ  C1
0801_   0028                        Jump    A$.AND.
0802    A10B                        DecSkip,EQ  C2
0803_   0028                        Jump    A$.AND.
0804    0800                        Ret

0805_   0328                        Jump    Loop.AND

0806_   0030                        Move    #Loop/16
0807_   8A00                        Move    W,PcLath
0808_   0320                        Call    Loop.AND
0809
0809                        .End
```