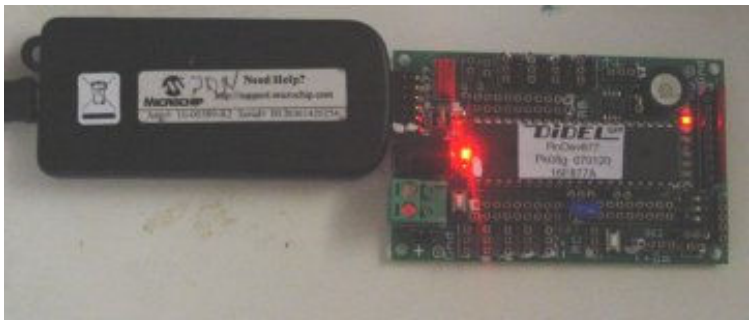
Fichier www.didel.com/pic/Pict877.pdf avec sources www.didel.com/pic/Pict877.zip

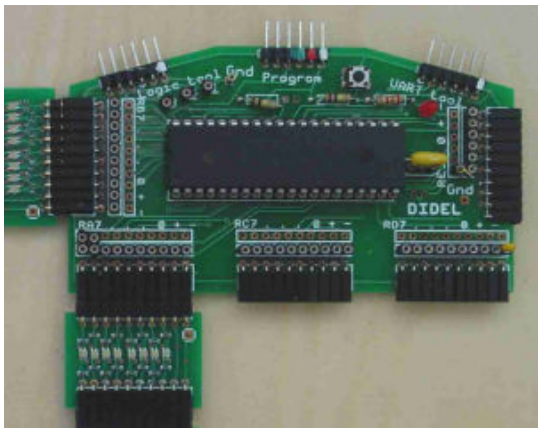
Programmes de test pour 16F877 (091028, a continuer)

La documentation du 16F877 (http://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf) fait 218 pages. Les exemples de cette notice permettent de se familiariser progressivement aux principales fonctionnalités, que l'on retrouve souvent sans modification dans les PIC plus petits.
Les exemples utilisent le portD pour afficher des valeurs binaires et l'interface série RS232 dans quelques exemples. Ils sont compatibles avec carte Dev877, mais tout carte ayant ces deux caractéristiques peut être utilisée. Les Microdules, carte M2840 avec us 16F877A et particulièrement bien adaptés pour tester des programmes.
La plupart de ces programmes de test ont été adaptés pour d'autre processeurs des familles 16F, 12F, 10F. (en préparation).

L'environnement utilisé est SmileNG. L'assemblage crée un fichier dans le format .Hex, qui est chargé dans le programmateur. Le programmateur conseillé est le PicKit2 de Microchip (Farnell). D'autres environnement comme IcProg peuvent convenir si le programmateur alimente la carte en 5V à la fin de la programmation, ce qui évite des déconnexion/reconnexion à chaque assemblage.



PicKit2 programmant la carte Dev877



Microdule M2840 avec 2 afficheurs

T877 - sorties

Ce premier programme oscille toutes les sorties du 877, sauf RA4 qui est seulement une entrée. Si le RS232 est câblé, il faut encore éviter de mettre en sortie l'entrée RxD (connectée à la sortie du circuit 232).
Le programme initialise les ports, le Motif est copié dans W et inversé après appel d'une boucle d'attente de 256x256x3 microsecondes (à 4 MHz). Pour inverser W on utilise l'instruction Xor #16'FF,W, ou Xor #2'11111111,W, ou Xor #-1,W. Ces trois écritures génèrent la même instruction. On pourrait copier la valeur Motif dans une variable Temp, et utiliser l'instruction Not Temp pour inverser.
L'initialisation des registres de direction de fait en Banque 1, en activant le bit RP0 du registre Status. Pour les port A,B,C, ou pourrait accéder directement les registres de direction avec les instructions Move W,TrisA, etc. Mais cela ne joue pas pour le PortD, donc autant les traiter tous de la même manière. Microchip déconseille d'utiliser les registre Tris, plus utilisés dans la famille 18F.
Le portA a une initialisation plus compliquée, puisque par défaut le convertisseur analogique est partiellement en service. Il faut initialiser le registre AdCon1 avec la valeur 16'06, comme expliqué page 31.
Le texte à la fin du programme est un commentaire très pratique pour vérifier que le bon programme a été chargé sur le programmateur. La configuration à l'adresse 2007 dit principalement que l'oscillateur est un quartz.

Chargez et exécuter ce programme. Modifiez le Motif. Raccourcissez le délai en initialisant CX2 à 100, 20.

```
\prog;T877Cli|Clignote les sorties
\b;RA RB0..7 RC0..7 RD0..7 RE0..2 période 0.65 s
; Ne pas utiliser si des sorties sont court-circuitées

.Proc   16f877
.Ref    16F877

Motif   = 16'55

CX1     = 16'20
CX2     = 16'21

.Loc    0
Deb:
        Clr     PortA
        Set     Status:#RP0
        Move    #16'06,W
        Move    W,AdCon1
        Move    #0,W            ; sorties
        Move    W,PortA
        Move    W,PortB
        Move    W,PortD
        Move    #2'10000000,W   ; Rx 232 en entrée
        Move    W,PortC
        Move    #2'111,W ; PortE en entrée
        Move    W,PortE
        Clr     Status:#RP0

        Move    #Motif,W
Loop:
        Xor     #-1,W
        Move    W,PortA
        Move    W,PortB
        Move    W,PortC
        Move    W,PortD
        Move    W,PortE
; Attente
 A$:    DecSkip,EQ      CX1
        Jump    A$
        DecSkip,EQ      CX2
        Jump    A$

        Jump    Loop

.Align  16'8
.16     "T","8","7","7","C","l","i"

.Loc    16'2007
.16     16'3F39

.End
```

T877aig - Aiguillage selon des entrées

Ce programme lit les entrées sur le port E et selon la valeur lue, lance des programmed différents qui agissent sur le portD. Ne pas initialiser la direction du port E avec la valeur –1, des bits de ce registre de direction ont des effets spéciaux sur le portD.
Il y a de nombreuses façons pour faire un aiguillage. S'il y a beaucoup de valeurs consécutives, on passe par une table (T877Ta.asm). S'il y a quelques valeurs parmi beaucoup de combinaisons, on fait des comparaisons successives (T877CoAig.asm).

```
\prog;T877Aig|Lit le portE et aiguille
;Il faut faire un reset pour changer de mode
.Proc   16F877
.Ref    16F877

; Variables
CX1     = 16'20
CX2     = 16'21
Temp    = 16'22

.Loc    0

Deb:
        Clr     PortA
        Set     Status:#RP0
        Move    #16'06,W
        Move    W,AdCon1
        Move    #0,W            ; sorties
        Move    W,PortA
        Move    W,PortB
        Move    W,PortD
        Move    #2'10000000,W   ; Rx 232 en
entrée
        Move    W,PortC
        Move    #2'111,W ; PortE en entrée
        Move    W,PortE
        Clr     Status:#RP0

        Move    #Motif,W
Deb:
        Clr     PortA
        Set     Status:#RP0
        Move    #0,W            ; sortie
        Move    W,PortD
        Move    #2'111,W ; PortE en entrée
        Move    W,PortE
        Clr     Status:#RP0
        Move    PortE,W
        And     #2'111,W
        Skip,NE
        Jump    Do0     ; =0
        Dec     PortE
        Skip,NE
        Jump    Do1     ; =1
        Jump    Do234567

Do0:    ; On décompte sur le portD
A$:     Dec     PortD
        Call    Delai
        Jump    A$

Do1:    ; On décale à droite (rotation)
        Clr     PortD
        SetC            ; Carry =1
A$:
        RRC     PortD
        Call    Delai
        Jump    A$
Do234567:       ; On clignote les bits 0 et 7
        Move    #2'10000001,W
        Move    W,Temp
A$:
        Move    Temp,W
        Move    W,PortD
        Move    #2'10000001,W
        Xor     W,Temp
        Call    Delai
        Jump    A$

Delai:  ; Attente  de 65ms
 A$:    DecSkip,EQ      CX1
        Jump    A$
        DecSkip,EQ      CX2
        Jump    A$
        Ret

.Align  16'8
.16     "T","8","7","7","A","i","g"

.Loc    16'2007
.16     16'3F39

.End
```

T877ana - entrées analogiques

Le programme T877a.asm lit une entrée analogique et affiche la valeur sur le portD. Le mieux est de câbler un potentiomètre, mais en touchant avec le doigt, on crée des tensions parasites qui sont affichées.
Comme expliqué pages 113 à 121, il faut initialiser 3 registres:
AdCon1 qui définit quelles broches sont liées aux convertisseurs interne. On a vu que la valeur 6 dit "aucune". La valeur 2 dit "toutes", sauf RA4 qui est spécial. A cause de cela AN4 est sur RA5.
AdCon0 sélectionne l'entrée qui va être lue. Un aiguillage interne se positionne et il faut attendre 20 microsecondes pour que la valeur soit stable à l'entrée du convertisseur. On active alors le bit Go, et on attend qu'il passe à zéro pour savoir que la conversion est finie et que la valeur 10 bits peut être lue dans les registres AdResH et AdResL. Avec l'initialisation choisie (bit7 dans AdCon1), les 2 bits de poids faible du résultat sont dans AdResL. Si le câblage n'est pas très soigné avec un plan de masse, la précision n'est en général pas suffisant pour travailler avec ces deux bits. Faites vos tests!

Le programme lit la valeur sur RA0. Il faut changer l'instruction après Loop: pour lire l'une des autres entrées.

```
\prog:T877Ana.asm| AN0 --> PortD

.Proc     16F877
.Ref      16F877

CX1       = 16'20
CX2       = 16'21
IniAdCon1  = 2'00000010   ; RA5 RA3 RA2 RA1 RA0
                          ; Poids forts dans AdResH
InAdCon0          = 2'00001110        ; RA0 seul
IniAdCon310       = 2'00000100        ; RA3 RA1 RA0
; les autres cas prévoient des tensions de référence
SelAd0    = 2'11000001
SelAd1    = 2'11001001
SelAd2    = 2'11010001
SelAd3    = 2'11011001
SelAd4    = 2'11100001        ; AN4 sur RA5 !

.Loc      0
Deb:
          Clr       PortA
          Set       Status:#RP0
          Move      #IniAdCon1,W
          Move      W,AdCon1
          Move      #2'011111,W
          Move      W,PortA           ; in
          Move      #0,W              ; sorties
          Move      W,PortB
          Move      W,PortC
          Clr       Status:#RP0

Loop:     Move      #SelAd0,W
          Move      W,AdCon0
          Call      Del20
          Set       AdCon0:#Go
 C$:      TestSkip,BC AdCon0:#Go
          Jump      C$
          Move      AdResH,W
          Move      W,PortD

 A$:      DecSkip,EQ        CX1
          Jump      A$
          DecSkip,EQ        CX2
          Jump      A$

          Jump      Loop

\rout:Del20| 20 \mu;s pour convertisseur AD
\mod:W
Del20:    Move      #6,W
 A$:      Add       #-1,W
          Skip,EQ
          Jump      A$
          Ret

.Align    8
.16       "T","8","7","7","A","n","a"

.Loc      16'2007
.16       16'3F39
.End
```

T877T0x - Timer

Le timer 0 TMR0 est identique sur tous les PICs, même les plus petits. C'est un compteur qui tourne sans arrêt et que l'on peut lire. Il est précédé d'un pré-diviseur qui permet de s'adapter aux vitesses voulues par l'application. Ce facteur de pré-division est défini dans le registre Option (page 21) mais le TMR0 est expliqué pages 49-51.

Le programme T877T0a.asm affiche l'état interne du timer, copié sur le PortD

```
\prog;T877T0a.asm|Test timer0
\b;On affiche le Timer avec un prédiviseur maximum
\b;Le portD fait un tour en 256x256 mirosecondes (0.065
sec)

.Proc     16F877
.Ref      16F877

.Loc      0
          Clr       PortA
          Set       Status:#RP0
          Move      #16'06,W
          Move      W,AdCon1
          Clr       W
          Move      W,PortD
          Clr       Status:#RP0
          Move      #2'00000111,W     ; Prescaler : 256
          Move      W,Option

Loop:
          Move      Tmr0,W
          Move      W,PortD

          Jump      Loop

.Align    8
.16       "T","8","7","7","A","n","a"

.Loc      16'2007
.16       16'3F39
```

```
.End
```

Lorsque le timer déborde, passe de 16'FF à 0, le bit TOIF dans le registre Option (page 21) est activé. On peut donc mettre à zéro ce bit, initialiser le timer et surveiller le passage à un de TOIF pour avoir un délai sans boucle d'attente.

Le programm T877T0b incrémente le portB a chaque dépassement, et le timer est initialisé pour que le dépassement se fasse toutes les 50ms. Etant donné que TMR0 est un compteur, il faut l'initialiser avec 0-TimDel (le complément de TimDel)

```
\prog;T877t0b.asm|Teste le timer0
\b;Incrémente le portD toutes les 50ms
\b;Fait un tour complet en 256 x 50ms = 12.8 secondes
.Proc     16F877
.Ref      16F877

IniOption = 2'00000111        ; Prescaler : 256  periode
0.256 ms
TimDel    = 193     ; periode 50ms

.Loc      0
          Clr       PortA
          Set       Status:#RP0
          Move      #16'06,W
          Move      W,AdCon1
          Clr       W
          Move      W,PortD
          Clr       Status:#RP0
          Move      #IniOption,W
          Move      W,Option
          Move      #-TimDel,W
          Move      W,TMR0

          Clr       PortD
Loop:
; on attend que le timer déborde
 W$:      TestSkip,BS  IntCon:#TOIF
          Jump      W$
          Clr       Intcon:#TOIF
          Move      #-TimDel,W
          Move      W,TMR0

          Inc       PortD

          Jump      Loop
.End
```

T877T0Int - Interrupt du au Timer

Le programme T877T0Int agit sur le portD par deux tâches en parallèle. D'une part le programme principal incrémente le portD lentement, comme on sait le faire, et d'autre part le timer et un compteur supplémentaire ont pour effet de mettre le portD à zéro. Il ne peut donc jamais compter plus loin que 15 (16'F) environ.
Pour activer les interruptions lorsque T0IF (Timer 0 Interrupt Flag) est activé, il faut activer dans IntCon le bit T0IE (Timer 0 Interrupt enable) et activer le bit général qui autorise toutes les interruptions GIE (page 22).
Dans ces conditions, lorsque TOIF est activé, le processeur saute à l'adresse 4. Il faut sauver le registre W et l'état F (noté aussi Status) avant de quittancer TOIF, faire le travail et rétablir W et F. Ne nous attardons pas maintenant sur l'astuce de ce rétablissement.

```
\prog;T877T0Int.asm|Timer par interruption
; Compte sur le PortD et l'interruption réinitialise le portD
.Proc     16F877
.Ref      16F877

\var;Registres|
SaveF     = 16'20
SaveW     = 16'21
CInt      = 16'22
CX1       = 16'23
CX2       = 16'24

\var;Ports|

.loc      0
          Jump      Deb
.loc      4
; Interrupt tous les 256 x 16 us
          Move      W,SaveW   ; Ne modifie pas F
          Swap      F,W       ; A cause du retour
          Move      W,SaveF
; On désactive le bit qui a demandé l'interruption
          Clr       Intcon:#TOIF

          DecSkip,EQ  CInt  ; Compteur pour ralentir
          Jump      F$
          Clr       PortD ; Chaque 256x256x16 us = ~1s
F$:
          Swap      SaveF,W
          Move      W,F
          Swap      SaveW
          Swap      SaveW,W
          RetI
\b;Programme principal
Deb:
          Move      #0,W              ; All outputs
          Move      W,TrisD
          Move      #2'00000011,W     ; Prescaler :16,
                                      ; 16 us TMR0 clock period
          Move      W,Option
          Clr       IntCon:#TOIF
          Move      #2'10100000,W     ; GIE and TOIE on
          Move      W,IntCon
Loop:
          Inc       PortD
; Attente pour incrémenter env 15 fois par seconde
  A$:     DecSkip,EQ        CX1
          Jump      A$
          DecSkip,EQ        CX2
          Jump      A$

          Jump      Loop
.End
```

T877Alpha

Connectons un PC à la carte 16F877 via un adaptateur RS232 et si nécessaire un adaptateur RS232-USB.
Le PC est préparé avec le programme Teraterm initialisé à 2400 bits/s avec le bon port COM.
Le programme envoie un + sur le terminal. Inutile de continuer si ce + n'est pas affiché, il y a un problème de câblage ou de configuration du PC.
Si le + a été affiché, l'action sur une touche du clavier affiche le code de cette touche en binaire sur le portD et lance l'impression de l'alphabet.
Il faut initialiser les registres associés à la liaison série selon la documentation pages 97-106. Les routines et la génération de l'alphabet sont facile à comprendre.

```
\prog:T877alpha.asm|Envoie alphabet à chaque touche
; Affiche le code Ascii de la touche sur le portD
; Terminal à 2400 bits/s

.Proc    16F877
.Ref     16F877

.Macro   Bank0to1
         Set     Status:#RP0
.Endmacro
.Macro   Bank1to0
         Clr     Status:#RP0
.Endmacro

\const;PortC| Série,
bTxD     = 6
bRxD     = 7
DirC     = 2'10000000      ;
InitC    = 2**bTxD         ; 1er car ok
IniRcSta = 2'10011000
IniTxSta = 2'00100100
IniSpBrg = 10'103  ; 2400 bits/s

\b;Ascii|
CR       = 16'0D
LF       = 16'0A
BEL      = 16'07

\var: |
.Loc     DebVar
Temp:    .16     1
Alpha:   .Blk.16 1         ; Pointeur dans la chaîne
CAlpha:  .Blk.16 1         ; Longueur de la chaîne

\prog; |
.Loc     0
Debut:
         Clr     PortA
         Bank0to1
         Move    #16'06,W
         Move    W,AdCon1
         Move    #0,W
         Move    W,PortD
         Bank1to0
         Move    #DirC,W
         Move    W,TrisC
         Move    #InitC,W
         Move    W,PortC

         Call    Iniser
         Move    #"+",W
         Call    SndSer
Loop:
         Call    SndCR
         Call    RecSer   ; On recommence à chaque
touche pressée
         Call    SndSer
         Move    W,PortD
         Call    SndCR
Alpha$:
         Move    #"A",W
         Move    W,Alpha
         Move    #26,W
         Move    W,Calpha
TA$:
         Move    Alpha,W
         Inc     Alpha
         Call    SndSer

b;Routines série

\rout;SndSer.asi|Serial transfer for 16F87x
\in:W
SndSer:
 A$:     TestSkip,BS PIR1:#TxIF
         Jump    A$
         Move    W,TxReg
         Ret

\rout;SndCr|Envoi CRLF
SndCR:
         Move    #CR,W
         Call    SndSer
         Move    #LF,W
         Jump    SndSer

\rout;SndSpace|Envoi espace
SndSpace:
         Move    #" ",W
         Jump    SndSer

\rout;SndBell|Envoi Bell
SndBell:
         Move    #Bel,W
         Jump    SndSer

\rout:RecSer|Serial receive at 9600 bit/s
\out:W
RecSer:
 R$:     TestSkip,BS PIR1:#RcIF
         Jump    R$
         Move    RcReg,W
         Ret

\rout:IfSer|Test if new character arrived
\out: CC no car W unchanged  CS car arrived DataRec = W
IfSer:
         TestSkip,BS PIR1:#RcIF
         Jump    No$
         Move    RcReg,W
         SetC
         Ret
No$:     ClrC
         Ret

\rout:IniSer|Serial port init (after portC init)
IniSer:
         Move    #IniRcSta,W
         Move    W,RcSta
 Bank0to1
         Move    #IniTxSta,W
         Move    W,TxSta
         Move    #IniSpBrg,W
         Move    W,SpBrg
 Bank1to0
         Clr     PIR1
         Ret

.Align   16'8
.16      "T","8","7","0","A","l","p","h","a"

.Loc     16'2007
.16      16'3F39

.End
```

```
DecSkip,EQ  Calpha
Jump        TA$

Jump        Loop
```