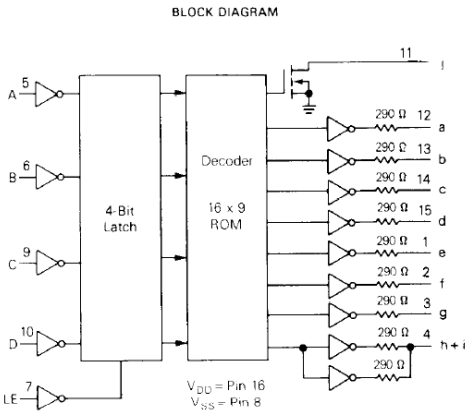
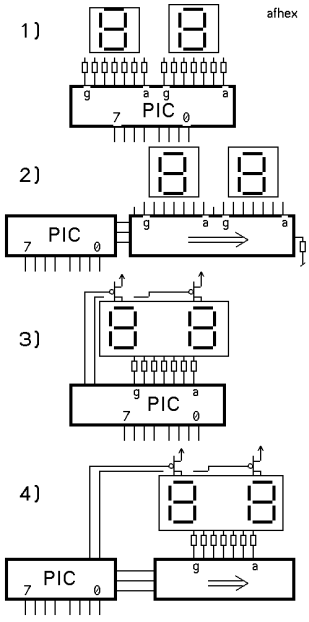
Etude d'une application – le Microdule AfHex

On veut afficher en hexadécimal, donc comme deux chiffres à 7 segments, un mot binaire 8 bits. Les logidules #178 se 1980 faisaient cela très bien, avec deux circuits NC14495 de 16 broches :
4 entrées pour le mot de 4 bits, 7 sorties pour les segments.
Ce circuit ne se fabrique plus depuis 15 ans, mais on en trouve encore à un prix excessif ($2.75), et il faut deux circuits. Une bonne solution serait d'utiliser une FPGA-CPLD genre Xilinx xc2c32a qui a les 8 entrées et 14 sorties nécessaires, et probablement assez de cellules.



Un microcontrôleur peut aussi faire ce travail, avec toutefois un problème pour le latch. On plusieurs solutions, par exemple :



1) Contrôleur avec 8 entrées et 7+7 sorties. Le processeur peut être un 28 pattes, 16F882 ou 16F737. Il faut 14 résistances (16 si on commande les points décimaux).

2) Contrôleur avec un registre série 16 bits
Le circuit TB62706 a une seule résistance qui fixe le courant sur chaque segment. Le processeur peut être un 14 pattes, par exemple les 16F630.

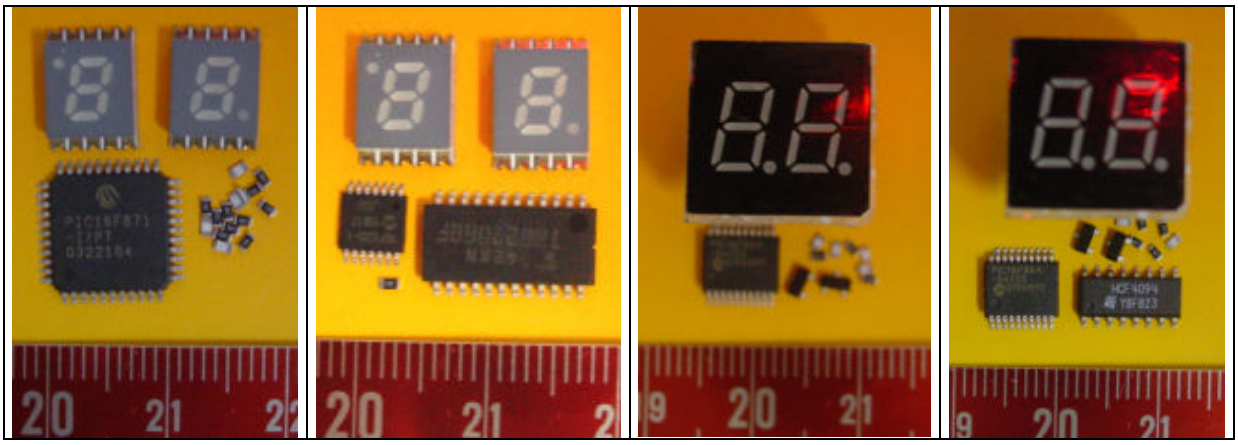
3) Contrôleur avec 8 entrées et 7 sorties, plus 2 sorties pour commander des transistors. L'affichage est mutliplexé. Le processeur peut être un 20 pattes, par exemple le 16F690. Il faut 7 résistances et 2 transistors

4) Contrôleur avec un registre série 8 bits et 2 transistors. L'affichage est multiplexé. Le processeur peut être un 18 pattes, par exemple le 16F627/628.

5) Un FPGA est une solution à considérer si on a les outils de développements

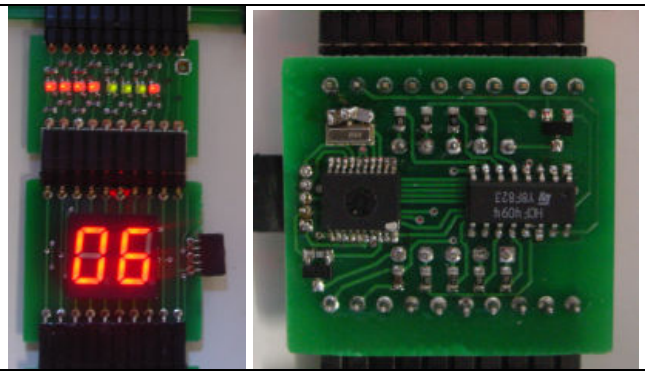
On peut encore imaginer de lire les 8 bits en entrée avec un registre à décalage, et utiliser comme en 2) un registre 16 bits en sortie pour l'affichage avec le TB62706. Le processeur a alors besoin de 4 lignes seulement. Si le registres est 8 bit et l'affichage multiplexé comme en 3), il faut 6 lignes de commandes et un processeur à 8 pattes comme le 12F508 convient.

Quelle est la solution la meilleure marché ? Quelle est la solution la plus compacte pour en faire un circuit imprimé ? Quels composant facilement disponibles utiliser ?
La question de la vitesse est sans importance. Il faut rafraîchir à 50 Hz, le pire des programmes n'arrivera pas à être aussi lent. La question de la facilité de programmation n'est non plus pas significative Il y aura au maximum 50 instructions et une table de 16 valeurs.



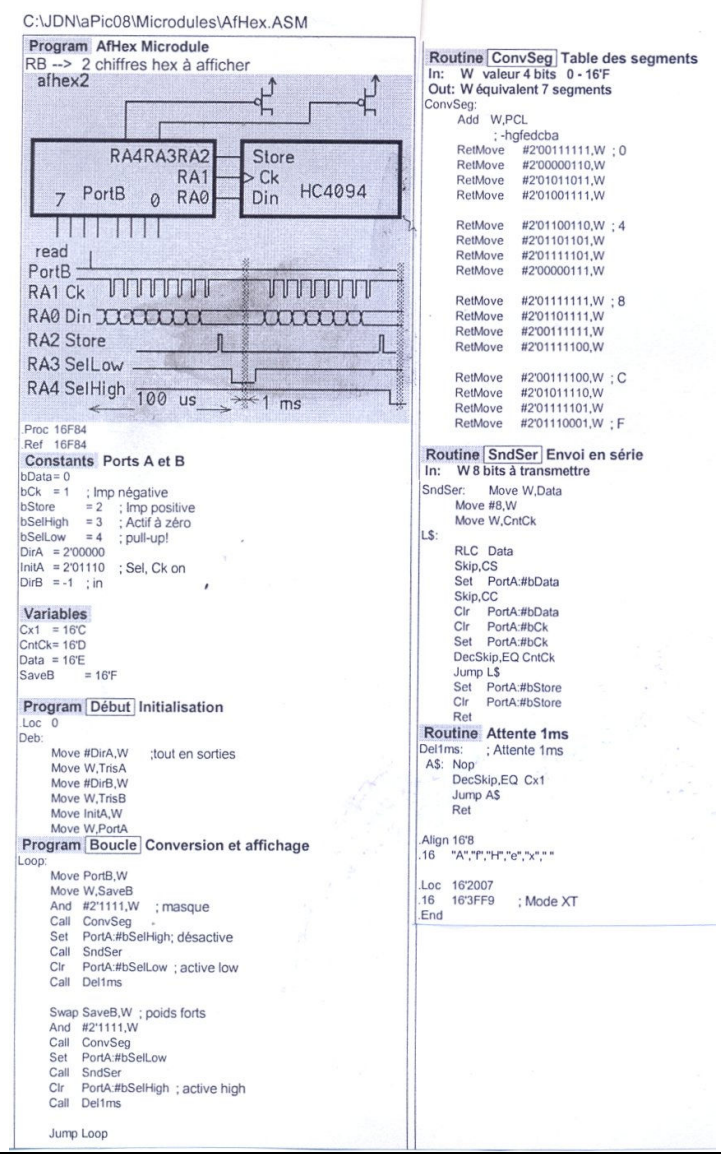
Circuits SMD associés à chaque solution

Pour avoir un Microdule affichage hexa, mieux adapté pour certaines applications que le module binaire, nous avons choisis la solution 4) avec un processeur 16F84 (un F88 serait préférable du point de vue prix/disponibilité, mais nous avons des F84 en stock). Le registre à décalage est un HC4094. L'affichage contient deux digits. Le module a ainsi les dimensions standard des Microdules, 30x28mm.



Le logiciel est facile : on lit le portB et on prends les 4 bits de poids faibles. Une table donne les segments qu'il faut allumer. Ils sont décalés en série et le transistor est activé pendant 1 à 10 ms. On lit l'autre moitié du portD pour afficher les poids forts. L'affichage est multiplexé, il faut mettre des résistance plus faibles pour avoir la même intensité lumineuse.
Le listage SmileNG est donné ci-dessous. On appréciera l'insertion d'une image explicative et les titres lisibles. Word ne décodant pas les séquences ''boa'', il a fallu scanner le listage.

jdn 090822



C:\JDN\aPic08\Microdules\AfHex.ASM

Program AfHex Microdule
RB --> 2 chiffres hex à afficher

afhex2


```
Proc 16F84
Ref  16F84
Constants  Ports A et B
bData= 0
bCk   = 1    ; Imp négative
bStore  = 2    ; Imp positive
bSelHigh  = 3    ; Actif à zéro
bSelLow   = 4    ; pull-up!
DirA  = 2'00000
InitA  = 2'01110   ; Sel, Ck on
DirB  = -1   ; in

Variables
Cx1   = 16'C
CntCk= 16'D
Data  = 16'E
SaveB      = 16'F

Program Début Initialisation
.Loc  0
Deb:
      Move #DirA,W     ;tout en sorties
      Move W,TrisA
      Move #DirB,W
      Move W,TrisB
      Move InitA,W
      Move W,PortA
Program Boucle Conversion et affichage
Loop:
      Move PortB,W
      Move W,SaveB
      And  #2'1111,W   ; masque
      Call  ConvSeg
      Set   PortA:#bSelHigh; désactive
      Call  SndSer
      Clr   PortA:#bSelLow  ; active low
      Call  Del1ms

      Swap SaveB,W  ; poids forts
      And  #2'1111,W
      Call  ConvSeg
      Set   PortA:#bSelLow
      Call  SndSer
      Clr   PortA:#bSelHigh  ; active high
      Call  Del1ms

      Jump Loop
```

```
Routine ConvSeg Table des segments
In:   W  valeur 4 bits  0 - 16'F
Out: W équivalent 7 segments
ConvSeg:
      Add   W,PCL
            ; -hgfedcba
      RetMove    #2'00111111,W  ; 0
      RetMove    #2'00000110,W
      RetMove    #2'01011011,W
      RetMove    #2'01001111,W

      RetMove    #2'01100110,W  ; 4
      RetMove    #2'01101101,W
      RetMove    #2'01111101,W
      RetMove    #2'00000111,W

      RetMove    #2'01111111,W  ; 8
      RetMove    #2'01101111,W
      RetMove    #2'00111111,W
      RetMove    #2'01111100,W

      RetMove    #2'00111100,W  ; C
      RetMove    #2'01011110,W
      RetMove    #2'01111101,W
      RetMove    #2'01110001,W  ; F

Routine SndSer Envoi en série
In:   W 8 bits à transmettre
SndSer:      Move W,Data
      Move #8,W
      Move W,CntCk
L$:
      RLC   Data
      Skip,CS
      Set   PortA:#bData
      Skip,CC
      Clr   PortA:#bData
      Clr   PortA:#bCk
      Set   PortA:#bCk
      DecSkip,EQ CntCk
      Jump L$
      Set   PortA:#bStore
      Clr   PortA:#bStore
      Ret
Routine Attente 1ms
Del1ms:      ; Attente 1ms
 A$:  Nop
      DecSkip,EQ  Cx1
      Jump A$
      Ret

.Align 16'8
.16   "A","f","H","e","x"," "

.Loc   16'2007
.16    16'3FF9      ; Mode XT
.End
```

2e génération
Le design a été refait selon la solution 1) avec un 16F737 ou 16F882. Un changement de l'affichage était nécessaire pour miniaturiser, et la solution avec 2 affichages séparé a été choisie.

Le schéma a moins de composants, et le routage est très simple en permutant astucieusement les lignes et en changeant l'ordre des bits dans les tables de segments.