CH-1092 Belmont, Switzerland
info@didel.com, www.didel.com

Conversion d'un quatret (nibble) en hexadécimal Ascii

Le problème est de partir d'un mot binaire 8 bits et l'envoyer en hexadécimal sur un terminal. Par exemple 2'10011100 = 16'9C génère les codes ASCII "9" puis "C".

Le problème revient à convertir 4 bits en un chiffre hexa. Si le module de conversion qui nous intéresse s'apelle BinToHex et le module d'envoi Send, l'envoi du byte Data s'écrit

```
        Swap    Data,W
        BinToHex
        Send            ; envoi poids forts
        Move    Data,W
        BinToHex
        Send            ; envoi poids faibles
```

La conversion n'est pas évidente car les codes Ascii ne se suivent pas: il y a un trou entre le 9 et le A.

Table de conversion

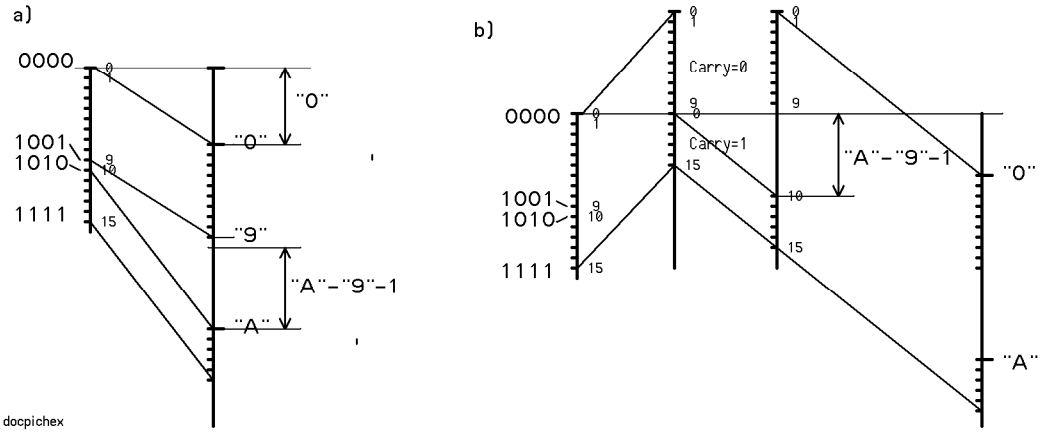
La solution la plus facile à comprendre passe par une table. Si la table est dans la même page que l'appel, on écrit simplement

```
BinToHex:
        Call    Conv
        ... (call Send, envoie le caractère dans W)
Conv:   And     #2'00001111,W
        Add     W,PCL
        RetMove #"0",W
        ...
        RetMove #"9",W
        RetMove #"A",W
        ...
        RetMove #"F",W
```

Le temps d'exécution est 7 μs à 4MHz et le code utilise 19 bytes

Solution algorithmique

Le principe est d'ajouter au mot binaire la valeur "0" (code Ascii du chiffre 0, qui vaut 16'30), et de rajouter le supplément nécessaire si le chiffre est supérieur à 9. Le problème avec le PIC c'est qu'il n'y a pas d'instruction de comparaison, et qu'il faut soustraire, ce qui modifie le résultat. Un variable auxiliaire évite cette perte (on peut aussi rajouter après, au bon endroit).
Il y a de nombreuses variantes pour tester si le chiffre est supérieur à 9.



0.95

La solution la plus élégante est décrite dans la partie b) de la figure ci-dessous. On soustrait 9+1 (ajoute -10) pour avoir le Carry qui vaut 1 pour les nombres supérieurs à 10, on corrige et on passe en hexa.

```
BinToHex:
        And     #2´00001111,W
        Add     #-10,W
        Skip,CC
        Add     #"A"-"9"-1,W
        Add     #"0"+10,W
```

Le temps d´exécution est 5 μs à 4MHz et le code utilise 5 bytes.

Le programme de test, qui envoie les 2 caractère Ascii vers un terminal est

```
Conversion binaire 8 bits - 2 chiffres hexa
Variable à déclarer: SaveW

Routine BinHex   in:  W
  out: routine Send appelée 2 fois pour traiter W code Asc
  mod: SaveW
BinHex:
        Move    W,SaveW
        Swap    SaveW,W
        And     #2´00001111,W
        Add     #-10,W
        Skip,CC
        Add     #"A"-"9"-1,W
        Add     #"0"+10,W
Call    Send
        Move    SaveW,W
        And     #2´00001111,W
        Add     #-10,W
        Skip,CC
        Add     #"A"-"9"-1,W
        Add     #"0"+10,W
Jump    Send
```

Conversion Hexa en binaire

Le problème inverse est de recevoir 2 chifres hexa et d´en former un nombre 8 bits. En appliquant le raisonnement de tout-à-l´heure en sens inverse, on a les instructions:

```
        ; On ne vérifie pas que c´est bien un chiffre hexa
        Add     #-["0"+10),W    ; ! Sub W,#"0"+10,W n´a pas le même effet
        Skip,CC
        Add     #-["A"-"9"-1),W
        Add     #10,W
```

Pour lire deux quatrets hexa et former le nombre binaire correspondant, on peut écrire:

```
  out: W = SavBin
RecHex:
        Call    RecSer
        Add     #-["0"+10),W
        Skip,CC
        Add     #-["A"-"9"-1),W
        Add     #10,W
        Move    W,SavBin
        Swap    SavBin
        Call    RecSer
        Add     #-["0"+10),W
        Skip,CC
        Add     #-["A"-"9"-1),W
        Add     #10,W
        Or      W,SavBin
        Move    SavBin,W
        Ret
```

Il est en fait plus pratique depuis un clavier de lire des chiffres hexa jusqu´à un termin

Si on veut vérifier que l´on reçoit bien des chiffres hexadécimaux, la seule solution est

```
; Le caractère Ascii est dans la variable Code
        Move    #"0",W      (non testé)
        Sub     W,Code,W
        Skip,CC
        Jump    Lower0
        Move    #"9"+1,W
        Sub     W,Code,W
        Skip,CS
        Jump    OK09
        Move    #"A",W
        Sub     W,Code,W
        Skip,CC
        Jump    Higher9LowerA
        Move    "F"+1,W
        Sub     W,Code,W
        Skip,CC
        Jump    HigherF
; OKAF
        Move    #"A"-10,W
        Sub     W,Code,W
        Jump    Fini
OK09:
        Move    #"0",W
        Sub     W,Code,W
        Jump    Fini
Lower0:
Higher9LowerA:
HigherF:        ; erreur à gérer ?
```

Solution des langages évolués

En C on écrirait pour convertir un nibble en Ascii

```
If Nibble =< 9  Ascii=Nibble+48  (Le C ne connaît que le décimal)
Else Ascii=Nibble+(65-10)
```

En Basic

```
If Nibble < 10 Then Ascii = Nibble+48
If Nibble > 9 Then Ascii = Nibble+(65-10)
```

C´est très intéressant de regarder le code produit et de comparer avec les 5 instructions du haut de la page.