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

This file: www.didel.com/doc/DopiBinD.pdf

Routines de conversion Binaire-Décimal et inverse

Un nombre binaire doit souvent être converti en décimal pour être affiché. La librairie XDec* contient les routines de conversion binaire-décimal et inverse pour des nombres entiers de 8 bits. Pour les nombres fractionnaires, que l'on trouve par exemple si l'on veut lire le capteur de température DS75, seule la routine de conversion binaire fractionnaire-décimal a été écrite. La librairie est formée des fichiers suivants:

XDecV.asi	Variables pour les conversions entières
XDecR.asi	Routines de conversion entière
XDec.asm	Programme de test
XFDecV.asi	Variables pour la conversion fractionnaire
XFDecR.asi	Routine de conversion fractionnaire
XFBin.asm	Programme de test pour la conversion fractionnaire

Nombres décimaux

Les nombres binaires sont le plus souvent représentés en hexadécimal: c'est plus compact pour l'affichage, et plus rapide pour l'introduction au clavier. Une variable du PIC contient un nombre binaire entre 0 et 16'FF (=255 décimal).

Il y a plusieurs représentations pour les nombres décimaux. Les chiffres de 0 à 9 sont codés en BCD, qui prennent 4 bits, soit un quatret (nibble en anglais). Un quatret peut être mémorisé dans un mot de 8 bits, avec les 4 bits de poids fort inutilisés (format simple). On peut compactifier 2 chiffres BCD dans un mot de 8 bits (format compactifié, packed en anglais). Avec l'instruction Swap et des masquages, c'est facile de passer de la représentation commpactifiée à la représentation simple, et inversément.

Le plus souvent, on a besoin d'une représentation en Ascii, pour envoyer le nombre sur un écran. Les nombres BCD 0000 - 1001 en format simple sont convertis facilement en chiffres Ascii de 0 à 9 (qui sont consécutifs dans le code Ascii) en ajoutant le code du chiffre 0, dont la valeur est connue par l'assembleur: il suffit d'écrire "0".

Par exemple, pour afficher sur un terminal le nombre 73, compactifié dans W avec la valeur 2'01110011 = 16'73 (attention, 73 tout seul serait converti en binaire par l'assembleur), il faut écrire:

```
Move    W,SavBin
Swap    SavBin,W        ; Ne modifie pas SavBin
And     #2'1111,W       ; Passage en représentation simple
Or      "0",W           ; Un Add a le même effet
Call    SndSer          ; Envoi au terminal (7)
Move    SavBin,W
And     #2'1111,W       ; Passage en représentation simple
Or      "0",W           ; Un Add a le même effet
Call    SndSer          ; Envoi des poids faibles (3)
```

Conversion de Binaire en Décimal

La conversion des 8 bits va fournir une valeur entre 0 et 255, soit trois chiffres décimaux. L'algorithme est de soustraire les plus possible de centaines, puis de dizaines, et de compter le nombre de fois. L'algorithme est donc relativement lent (max 55 μs à 4 MHz). Si on pouvait facilement diviser par 10, cela serait un peu plus rapide, surtout s'il fallait convertir des nombres de 16 bits ou plus. Notre routine est donc:

Routine **BinDec** **Convertit un nombre binaire 8 bits en trois quatrets (représentation simple)**

in: Binaire valeur 8 bits
out: Centaine Dizaine Unite
mod: Binaire

```
BinDec:
        Clr     Centaine
        Clr     Dizaine
        Move    #100,W
        Sub     W,Binaire
        Skip,CS
        Jump    D1
        Inc     Centaine
        Sub     W,Binaire
        Skip,CS
        Jump    D1
        Inc     Centaine
        Sub     W,Binaire       ; Max 256, inutile tester
D1:     Add     W,Binaire
D2:     Move    #10,W
        Sub     W,Binaire
        Skip,CS
        Jump    U
        Inc     Dizaine
        Jump    D2
U:      Add     W,Binaire
        Move    Binaire,W
        Move    W,Unite
        Ret
```

Le programme de test est facile à écrire et suppose que l'on dispose des routines RecHex (pour entrer au clavier le nombre binaire en hexa) et la routine SndSer (pour afficher le code Ascii des digits obtenus). Ces routines sont expliquées dans la librairie XTrSer (www.didel.com/doc/DopiSer.pdf).

Fin du programme XDecBD.asm

```
; Définitions, variables série et initialisation comme dans XTrSer.asm.
; Ajouter ou insérer les variables dans XDecV.asi

Loop:   Call    RecHex          ; Taper 2 chiffres hexa et un espace
        Move    W,Binaire
        Call    SndHex          ; si le terminal n'a pas
        Call    SndSpace        ; fait l'echo
        Call    BinDec
        Move    Centaine,W
        Or      "0",W
        Call    SndSer
        Move    Dizaine,W
        Or      "0",W
        Call    SndSer
        Move    Unite,W
        Or      "0",W
        Call    SndSer
        Call    SndCR
        Jump    Loop

.Ins    XDecR.asi
.Ins    XTrSer.asi
```

Le programme disponible sur Internet (www.didel.com/doc/XDec.asm) teste la conversion dans les deux sens.

Conversion de Décimal en Binaire

Limitons-nous à introduire 2 chiffres décimaux, donc un maximum de 99 = 16'63. Le premier problème est de savoir si on vérifie la validité de la donnée (chiffres entre 0 et 9) avant d'appeler la routine de conversion. Cela coûte quelques comparaisons, et il faut décider ce que l'on fait en cas d'erreur. La routine elle-même est très simple, puisqu'il suffit de multiplier les dizaines par 10 (= 2'1010) et d'ajouter les unités. Suivant que l'on part de données en représentation simple ou compactifiée, on a deux variantes de programme.

Routine **DecBin** **Chiffre dizaines et unités dans 2 mots de 8 bits**

in: Dizaines Unites
out: Binaire

```
DecBin:
; Dizaines * 10 = 2'1010
        ClrC
        RLC     Dizaine,W
        Swap    Dizaine
        RRC     Dizaine
        Add     Dizaine,W
; Plus qu'à ajouter les unités
        Add     Unite,W
        Move    W,Binaire
        Ret
```

Routine **Dec8Bin** **BCD 8bits**

in: BCD 8 bits "packed"
out: Binaire
mod:

```
Dec8Bin:
        RRC     BCD,W
        And     #2'01111000,W
        Move    W,Binaire
        ClrC
        RRC     Binaire
        RRC     Binaire
        Add     W,Binaire
        Move    BCD,W
        And     #2'00001111,W
        Add     W,Binaire
        Ret
```

Le coeur du programme de test de la première routine (qui ne vérifie pas que l'on tape un chiffre entre 0 et 9) est le suivant:

Centre du programme XDecBD.asm

```
Loop:   Call    RecSer          ; Résultat dans W et SerData
        Call    SndSer          ; Si le terminal ne fait
        Move    SerData,W       ;  pas l'echo des caractères
        And     #2'1111,W       ; Format simple
        Move    W,Dizaine
        Call    RecSer
        Call    SndSer          ; Si le terminal ne fait
        Move    SerData,W       ;  pas l'echo des caractères
        And     #2'1111,W       ; Format simple
        Move    W,Unite
        Call    SndSpace
        Call    DecBin
        Call    SndHex
        Call    SndCR
        Jump    Loop
```

Un programme prêt à être testé est accessible sur Internet (www.didel.com/doc/XDec.asm); il demande deux chiffres décimaux, convertit et affiche en binaire et reconvertit en décimal.

Conversion fractionnaire

Un nombre fractionnaire 8 bits est par exemple 0.1000000 = 0.5, 0.01000000 = 0.25 ou 0.00110001 = 0.19140625. Le nombre de chiffres théoriques du résultat peut être élevé, mais comme la précision du nombre binaire est de 1/256 eme, cela n'a aucun sens de calculer plus de trois décimales. Contentons-nous de deux avec la routine FBinDec. L'algorithme utilise des multiplications par 10=2'1010, avec extraction de la partie entière après chaque multiplication. Le résultat est en format simple dans Dixieme et Centieme. Il serait facile de rajouter le calcul des millièmes.

Un problème est l'arrondi. Pour un arrondi précis, il faut ajouter 0.005 = 2'0.00000001010001.... En ajoutant 0.00000001 on commet une erreur que l'on remarque en convertissant 0.00100000 (16'0.20) qui devrait afficher 0.13 (le résultat téorique est 0.125) et affiche 0.12 (mais 16'21 affiche 0.25). Une autre chose à laquelle il faut penser (c'est le programme de test qui l'a mis en évidence!), est que l'arrondi en dessus demande un traitement particulier si l'on convertit 16'FF. Le résultat 0.99610 est arrondi à 00 et il faut tenir compte du dépassement pour ajouter une unité à la partie entière. Cela rajoute donc une dizaine d'instructions pour faire les choses proprement.

Routine **FBinDec** **Conversion binaire fractionnaire 8 bits**

Adapté pour capteur de température DS75 I2C

in: W partiue fractionnaire 8-bits
out: Dixieme Centieme Ascii digits
out: CS si résultat supérieur à 0.99 (si arrondi en dessus)
mod: Fraction Dixieme Centieme

```
; L'algorithme multiplie par 5 puis par 2 (pour éviter
; de travailler en 16 bits avec une multiplication par 10)
; et extrait la partie entière
FBinDec:
; Pour un arrondi en dessus, on ajoute 0.005 = 2'00000001
; mais il faut tenir compte que 0.995 sera arrondi en 00
        Clr     Dixieme
        Clr     Centieme
        Add     #2'00000001,W       ; seulement si
        Skip,CC                     ; arrondi
        Ret                         ; en dessus
        Move    W,Fraction
        ClrC
        RLC     Fraction
        RLC     Dixieme             ; x 2
        RLC     Fraction
        RLC     Dixieme             ; x 4
        Add     W,Fraction          ; x 5
        Skip,CC
        Inc     Dixieme             ; dépassement
        ClrC
        RLC     Fraction            ; x 10
        RLC     Dixieme             ; Valeur 4 bits
        Move    Fraction,W
        ClrC
        RLC     Fraction
        RLC     Centieme            ; x 2
        RLC     Fraction
        RLC     Centieme            ; x 4
        Add     W,Fraction          ; x 5
        Skip,CC
        Inc     Centieme
        ClrC
        RLC     Fraction
        RLC     Centieme            ; x 10
        ClrC                        ; Pas de dépassement
        Ret
```

Le programme de test demande une valeur hexa (il n'y a pas les instructions pour l'écho) et affiche le résultat en tenant compte du dépassement possible. La partie centrale de ce programme (www.didel.com/doc/XDecF.asi) est donnée ci-après.

```
TestFrac:
        Call        SndCR
        Call        RecHex
        Call        FBinDec
        Move        #"0",W
        Skip,CC
        Move        #"1",W
        Call        SndSer
        Move        #".",W
        Call        SndSer
        Move        Dixieme,W
        Or          #"0",W          ; Convert to Ascii
        Call        SndSer
        Move        Centieme,W
        Or          #"0",W          ; id
        Call        SndSer
        Jump        TestFrac
.Ins    XBcdR.asi
.Ins    XTrSerR.asi
```

Division par 10

La multiplication par 10 se fait facilement par des décalages, et le PIC se prête bien pour des décalages multiprécision. La division par 10 est plus délicate à programmer; il faut appliquer l'algorithme par soustraction, rétablissement en cas de résultat négatif et décalages. Un exemple en 16 bits donne le résultat en 12 cycles.

Diviser par 10 n'apporte aucun avantage particulier, à part le fait que l'on sait comment faire l'alignement initial. La routine pourrait être midifiée pour diviser 16 bita par 8 ou 16 bits, quotient 16 bits.

```
 11010110 10110110
-1010
 0110                    1
-1010
  -                      0
 1101
-1010
 0111                    1
-1010
  -                      0
 1110
-1010
  100 1                  1
 -101 0
  -                      0
 100 10
 -10 10
  10 001                 1
  -1 010
     1011                1
   -1010
     0010                1
    -1010
      -                  0
     0101
    -1010
      -                  0
     1011
    -1010
      0010
    -1010
      -                  0
Reste de la division 2'10  Quotient 2'101010111000
```

Le programme correspondant (www.didel.com/doc/XDiv10V.asi, XDivR.asi et XDiv.asm) utilise les variables DdHigh et DdLow comme dividende, DivH et DivLow comme diviseur, ResH et ResLow pour le quotient. Le programme de test XDiv10 demande DdHigh et DdLow et affiche ResHigh ResLow.

Avec cette routine, il est facile de convertir un nombre binaire 16 bits en nombre décimal 5 digits. Cinq divisions successives par 1010 donnent comme reste des divisions successives les unités, dizaines, etc. Pour convertir un nombre décimal fractionnaire de décimal en binaire, l'algorithme général demande aussi des divisions par 10.