# Réalisation d'un système d'acquisition série pour PC

Mathieu Allard

M1MNE 2013/2014 Semestre 1

## Présentation et Objectifs

On se propose de réaliser un système d'acquisition d'une grandeur analogique pour PC, via le port série, on utilisera un micro-contrôleur PIC16F877. Le développement se fera à l'aide du logiciel MPLAB, en assembleur. On divise le programme en plusieurs sous-programmes que l'on a testé au fur et à mesure, séparément.

1. Acqusition d'une valeur
2. Emission de caractères ascii sur l'hyperterminal (port série) avec l'USART
3. Transcodage Binaire-Ascii
4. Utilisation du timer et des interruptions
5. Utilisation des interruptions clavier (réception de caractères ascii)
6. Choix d'un mode de fonctionnement

# 1 Acquisition d'une valeur

On commence avec le fichier "tutor_pic.asm" qui envoie un nombre codé en binaire sur les diodes de la "Demo Board" du contrôleur, selon la tension réglée par un potentiomètre présent sur cette même carte.

On le modifie de telle sorte que le nombre ne soit pas envoyé sur les leds (PORTB) mais dans une valeur "valeur" que l'on stocke en 0x71 (0x70 étant réservé par le debugger).

## Description du programme

*lignes 114-119 et 78-83 en annexe*

*NB : La partie concernant TRISB/PORTB n'est pas présente dans le code final, on ne veut faire qu'une simple acquisition dans un registre que l'on va utiliser par la suite pour convertir et afficher la valeur.*

TRISB permet de choisir l'utilisation de PORTB en entrée ou en sortie, en initialisant tous ses bits à 0 avec clrf, les 8bits de PORTB seront utilisés en tant que sortie.
Des leds sont branchées sur PORTB, permettant d'observer la valeur que l'on va passer au registre.

Les lignes suivantes configurent le convertisseur Analogique/Numérique.
ADCON1 ('00001110')paramètre les sorties affectées :

- Le bit 7, ADFM donne le format du résultat, ici
- Les bits 6,5,4 ne sont pas utilisés
- les bits PCFG3-0 configurent les ports : ici on utilise juste AN0/RA0 en entrée analogique et les autres (AN1-7/RA1-7) en entrée-sortie numérique. Cela affecte aussi les valeurs VDD et VSS à VREF+ et VREF-.

ADCON0 ('01000001') paramètre la conversion des données :

- les 2 bits les plus forts ADCS1,ADCS0 à "01" indiquent la fréquence utilisée, ici fosc/8.
- Les 3 bits suivant CHS2,CHS1,CHS0 "000" sélectionnent la "channel", soit les pins des entrées à convertir.
- Le bit suivant GO, à "0" indique qu'il n'y a pas de conversion en cours.
- Le bit1 ? ? ?
- Enfin, le dernier bit, ADON à "1" active la conversion anlogique-numérique.

Le résultat est stocké dans ADRESH

Pour démarrer une conversion on met le bit GO de ADCON0 à 1 Le programme entre dans une bouclei tant que GO n'est pas repassé à 0, indiquant que la conversion est terminée pour s'assurer d'avoir la bonne valeur.

Et à la fin, la valeur convertie dans ADRESH est déplacée dans PORTB pour être affichée sur les leds. Dans notre cas, on change "PORTB" par "valeur" pour que l'acquisition soit stockée dans un registre utilisable par la suite.

# 2 Communication de caractères ascii sur le port série

On va maintenant écrire un sous-programme permettant de communiquer avec l'hyperterminal, on veut y afficher un caractère ascii précédemment tapé au clavier.

## Description du code

*lignes 85-93 pour l'initialisation, 28-29 RCREG, 176-209 TXREG*

### Initialisation

Il faut choisir une vitesse de transmission, on veut ici un baud rate de 9600, d'après la doc(p98) il faut donner au registre SPBRG la valeur 25, en décimal pour un processeur à 4Mhz à une telle vitesse.

Ensuite, le registre RCSTA ('10010000') concerne la réception par le port série :

- Le bit 7 active le port série, il faut le mettre à 1
- '0'sur le bit 6 permet de travailler en 8bits
- On utilise le mode asynchrone, le bit 5 n'a pas d'importance
- En activant le bit 4, on active la réception de données en continu (necessaire pour la suite puisqu'on veut travailler avec des interruptions clavier)
- les derniers bits ne nous intéressent pas, on désactive leurs fonctions.

Et TXSTA ('00100100') concerne l'émission de données :

- En asynchrone, le bit 7 n'importe pas
- On sélectionne une transmission 8-bits en mettant le bit 6 à '0'
- On active la transmission avec le bit 5 à '1'
- le bit 4 à '0' choisit le mode de fonctionnement asynchrone
- le bit 3 n'est pas implémenté
- Le bit 2 à '1' sélectionne "high speed" en mode asynchrone
- Le bit 1 (TRMT) contrôle si le registre TSR est vide ou plein
- Le dernier bit ne sert qu'au mode de transfert 9 bits

### programme

Les valeurs en ascii tapées au clavier sont stockées dans le registre RCREG, on va alors copier sa valeur dans "command" le registre initialisé au début en 0x77.

Ensuite, la valeur en code ascii dans TXREG est affichée dans l'hyperterminal, on va donc y placer les caractères que l'on veut afficher.

# 3 Conversion Binaire-Ascii

Maintenant que l'on peut faire une acquisition de valeur et envoyer des caractères ascii sur l'hyperterminal, on veut pouvoir convertir les valeurs mesurées en ascii pour pouvoir les afficher. La valeur avec laquelle on travaille est codée sur 8 bits, c'est un nombre "valeur" décimal compris en 0 et 255. Il représente en réalité une tension comprise en 0 et 5 volts, on va donc faire la conversion.

L'unité est le résultat de la divion de "valeur" par 51, 255 étant égal à 5, le maximum. On procède par soustractions successives, en incrméntant un compteur, tant que le reste est positif.

Exemple, si valeur =110 : $110 - 51 = 59; unit = unit + 1$
$59 - 51 = 8; unit = unit + 1$
$8 - 51 = -43; ons'arrte$

Pour la partie décimale on va procéder de la même façon avec le reste des soustractions précédentes, attention à rajouter 51 à ce reste pour retrouver sa véritable valeur.

Une unité = 51, une decimale vaudra donc $51/10 = 5.1$, seulement, on ne peut pas avoir cette précision, on prendra donc 5 plutôt que 5.1, et on effectue à nouveau les soustractions, avec 5 cette fois et en incrémentant un autre compteur correspondant aux décimales.

Enfin, il reste un petit problème, dû à l'approximation $5.1 = 5$, si le reste après calcul de l'unité est égal à 50, on va se retrouver avec une partie décimale de 10. Ce cas est peu probable : il y a 5 chances sur 255 mais mieux vaut y pallier. La solution est simple : si on rencontre ce cas, on ne va pas calculer la partie décimale, mais l'arrondir à 9. Enfin, pour encoder une valeur décimale est un caractère ascii y correspondant, on ajoute 30. (30=0 31=1 etc..)

## description du code

*lignes 124-168 en annexe* On a besoin de plusieurs variables pour cette partie : valeur, qui contient la valeur analogique, unite la partie entière, et decim la partie décimale.

L'instruction btfsc/btfss on peut tester le bit d'un registre et sauter l'intruction suivante selon le résultat. On teste ici le bit C de STATUS : s'il est à '0', cela signifie qu'il n'y a pas de reste à notre soustraction. (ou que ce reste est <0)
De même, le bit Z est à '1' si le résultat d'une opération arithémtique est égale à 0, et à '0' sinon.

Lorsque l'on arrive dans la partie conversion, on s'assure d'abord que le registre "unite" est remis à 0 après une précédente conversion, puis on affecte "51" en décimal à w, que l'on va retirer à la valeur mesurée. (substract W from F). On fait alors le test sur le bit C de status pour savoir si on incrémente ou non unite, selon le résultat du reste de la division, si on a incrémenté unite, alors on va procéder à une soustraction suivante en revenant au début de la boucle.

Une fois le reste nul, on passe à la suite, on remet la valeur en positif en lui ajoutant "51", et on effectue les calculs pour la partie décimale.

Juste avant de procéder de la même façon pour les soustractions successives pour la partie décimale, on s'assure que le reste à tester n'est pas égal à 50. C'est dans cette partie qu'on teste le bit Z de STATUS après soustraction de "50" à valeur, si Z est à 1, on va directement au résultat après avoir affecté "9" à la partie décimale.

# 4 Timer et interruptions

On veut maintenant être capable d'afficher les valeurs à intervalles réguliers d'une seconde, la fonction timer du PIC va nous permettre de le réaliser. On utilise timer0. Le micro-contrôleur est cadencé à fosc=4Mhz, en entrée du timer on a fosc/4=1Mhz. Timer0 compte sur 8bits et lève un flag TOIF de INTCON lorsqu'il arrive au bout, soit toutes les 256 micro-secondes. On peut l'utiliser avec un prescaler à un ratio jusqu'à 1/256, ce qui signifie que notre timer va lever un flag toutes les 256*256=65ms.

On travaille ici avec les interruptions, cela signifie qu'à chaque fois que le timer a compté 256*256, le flag TOIF se lève et on interromp le programme pour entrer dans une routine d'interruption. Lors d'une interruption, le programme va éxécuter les instructions en 0x04 jusqu'à retfie qui le fait revenir où il était précédemment. Pour obtenir un affichage par seconde il va falloir compter 15 interruptions (65*15=975ms soit environ 1 seconde, la précision n'étant pas importante dans notre cas) et afficher notre valeur seulement après. On va donc utiliser un compteur dans la routine d'interruption qui retourne attendre 15 interruptions timer avant d'appeler le programme d'acquisition/conversion/affichage.

## description du code

*initialisation lignes 94-98, interruption lignes 53-71 en annexe*

### Initialisation

On utilise timer0 qui s'initialise avec le registre OPTION_REG ('00000111')
- Les deux premiers bits ne nous intéressent pas
- le bit 5 (TOCS) à '0' sélectionne l'horloge interne
- le bit TOSE à '0' choisit une incrémentation à front montant
- le bit 3 (PSA) assigne le prescaler à timer0
- les bits PS2 :PS0 sélectionnent le ratio du prescaler, ici 1 :256 le maximum

### Compteur

Dès le début, on remet à '0' manuellement le flag TOIF, pour que les interruptions suivantes puissent avoir lieu.

Une variable "count" est incrémentée avec incf à chaque passage.
Puis on effectue l'opération de soustraction de w=15 à count, on place le résultat dans w pour ne pas avoir à remettre count à sa "bonne" valeur à chaque fois.

On teste alors le résultat de la soustraction avec le bit Z de STATUS, tant qu'il est à 0, on quitte l'interruption directement.

Lorsque l'on a atteint la 15eme interruption timer, le bit Z de status sera à '1', on va alors appeler le programme d'acquisition/conversion/affichage puis remettre count à 0, et enfin quitter l'interruption.

# 5 Interruptions clavier et mode de fonctionnement

Pour finaliser le programme, on veut qu'il fonctionne dans deux modes différents :
- Automatique : une acquisition chaque seconde
- Manuel : une acquisition lorsque l'on appuie sur la touche
- Il faut que le programme réponde à une touche pour changer de mode : a pour passer en automatique, XXX pour passer en manuel

Il va donc falloir executer différentes instructions selon la touche appuyée pour construire les deux modes.

Pour autoriser les interruptions clavier on autorise les interruptions usart, périphériques, et globales. Pour cela on met à 1 le bit RCIE de PIE1 et les bits PEIE et GEIE du registre INTCON. On contrôle également le bit 4 de RCSTA qui active la détection de valeurs venant du clavier en continu. Le programme s'interromp maintenant de la même façon qu'avec le timer dès que l'on presse une touche sur le clavier. Il va donc falloir tester dans cette routined'où viennent les interruptions.

## Structure du programme

Lorsque le timer ou le clavier provoque une interruption, on entre dans une routine à org0x04, et on effectue des opérations pour déterminer la provenance d'une interruption :
D'abord on regarde si ça vient du clavier : si ce n'est pas le cas elle vient donc du timer, on est en mode automatique (on prend soin de désactiver le timer si on passe en mode manuel) on va alors à notre compteur auto, et lorsqu'il arrive à 15 on appelle le programme (acqusition-conversion-affichage)
. Si l'interruption vient du clavier on va tester différentes lettres :
- si c'est r, on désactive les interruptions timer pour passer en mode manuel
- si c'est a, on va dans le mode automatique (cette routine active (ou réactive) les interruptions clavier
- si c'est d, on appelle le programme, après s'être assuré que les interruptions soient désactivées (donc que l'on est bien en mode manuel)

## description du code

*initialisation lignes 100-104 choix du mode lignes 24-50* La routine d'interruption va faire appel au programme d'acquisition soit quand on appuie sur une touche, soit chaque seconde :

Les premières instructions testent le bit PIR1 de RCIF qui est à 1 si le clavier a reçu une valeur, si ce n'est pas le cas, on va en mode automatique, où l'on fait une acquisition par seconde, son fonctionnement est décrit à la page précédente.

Si l'interruption vient du clavier, il va falloir tester de quelle touche elle provient pour décider du comportement du programme.
Le test est à chaque fois une soustraction de la valeur en ascii d'un caractère avec "command" le registre dans lequel on stocke la valeur de RCREG.

- Si c'est r, on passe le bit TOIE de INTCON à 0, ce dernier masque le flag TOIF, on aura donc plus d'interruptions venant du timer : on est en mode manuel
- si c'est a, on va en mode automatique, où l'on remet TOIE à '1' et où l'on fait une acquisition chaque seconde.
- si c'est d, on apelle directement le programme d'acquisition/conversion/affichage, mais seulement après avoir vérifié que TOIE est à '0' soit que l'on est bien en mode manuel.

# 6 Programme complet

*cette partie récapitule l'utilisation des sous-programmes et le déroulement général de notre programme*

### les différentes parties du programme :

— Initialisation
— main
— irq, routine d'interruption
— Programme, contenant les directives d'acquisition, de conversion et d'affichage

Lorsque l'on lance le programme il passe dans une phase d'initialisation où l'on configure le convertisseur numérique, la communication avec l'hyperterminal, les autorisations d'interruption, et le timer, par défaut on est en mode automatique. Il va ensuite dans la boucle "main", et attend une interruption.

En général, lors d'une interruption, on sauvegarde le contexte, cependant ce n'est pas utile ici, d'où les lignes concernant la sauvegarde de status et w commentées.
La routine d'interruption paramètre les différents modes en activant/désactivant les interruptions dûes au timer. C'est également elle qui appelle le programme dans les différents cas correspondants au mode de fonctionnement en cours.

## Annexe : Code PIC utilisé

```
 list p=16f877
        ; Include du fichier de description des définitions du 16f877
        include "p16f877.inc"

;on initialise "valeur" la valeur reçue par le PIC
valeur                  EQU 0x71                ; stockage de la valeur mesurée
unite                   EQU 0x72                ; partie entière après conversion
decim                   EQU 0x73                ; partie décimale après conversion
count                   EQU 0x74                ; compteur pour le timer
;w_tmp                  EQU 0x75                ; tampon pour le sauvegarde du contexte
;status_tmp             EQU 0x76                ; Sauvegarde de status
command                 EQU 0x77                ; stocke la valeur a,r,d
;commence à l'adresse 0000
                org     0x000
                nop
                goto    initialisation          ;initialisation du programme (ligne 78)

;******** INTERRUPTION : TIMER ET SEELECTION DU MODE*********
;***********************************************************
; par défaut, on se placer en mode automatique

irq             org 0x004                       ; début interruption
; test de la valeur reçue
                banksel PIR1
                btfss   PIR1,RCIF               ; interruption dûe au clavier?
                goto    auto

                banksel RCREG
                movfw   RCREG
                movwf   command                 ; "command" = valeur tapée

                movlw   "r"                     ;si c'est r : changement de mode
                SUBWF   command,w
                btfsc   STATUS,Z
                bcf     INTCON,T0IE             ;T0IE à 0 : pas d'interruption timer

                movlw   "a"                     ;si c'est a on change de mode
                SUBWF   command,w
                btfsc   STATUS,Z
                goto    auto

                movlw   "d"                     ;si c'est d : acquisition
                SUBWF   command,w
                btfsc   STATUS,Z
                goto acqmanu
                goto sortie

acqmanu         btfss   INTCON,TOIE             ;acquisition ssi mode manuel
                call    programme
                goto    sortie

;timer (auto)
auto
                bsf     INTCON,T0IE             ; interruptions timer autorisées
                bcf     INTCON,T0IF             ; flag interruption à 0
                incf    count
                movlw   .15
```

```
58                  subwf   count,w
59                  btfsc   STATUS,Z                ;  on afiche la valeur
60                  goto    auto2
61                  goto    sortie                  ; sinon, on quitte l'interruption
62  auto2
63                  call    programme               ; on appelle le programme complet
64                  clrf    count                   ; on réinitialise le compteur
65                  goto    sortie
66  ;sortie de l'interuption
67  sortie
68  ;                 movfw   status_tmp                      ;
69  ;                 movwf   STATUS                          ;
70  ;                 movfw   w_tmp                           ; restauration du contexte
71                  retfie                          ; return from interrupt
72
73  ;*************************************************
74                  ; FIN D'INTERRUPTION
75
76  initialisation
77  ; config de l'ADC
78                  banksel ADCON1
79                  movlw   B'00001110'             ;Left justify,1 analog channel
80                  movwf   ADCON1                  ;VDD and VSS references
81                  banksel ADCON0
82                  movlw   B'01000001'             ;Fosc/8, A/D enabled
83                  movwf   ADCON0
84  ;vitesse de transmission, config de l'USART
85                  banksel SPBRG
86                  movlw   .25                      ; baud rate =9,6k
87                  movwf   SPBRG
88                  banksel RCSTA
89                  movlw   B'10010000'
90                  movwf   RCSTA
91                  banksel TXSTA
92                  movlw   B'00100100'
93                  movwf   TXSTA
94  ; INITIALISATION DU TIMER
95                  banksel OPTION_REG
96                  movlw   B'00000111'             ; initialisation du timer0 /256
97                  movwf   OPTION_REG
98                  bsf     INTCON,T0IE             ; autorisation inter timer (mode a)
99  ;clavier
100                 banksel PIE1
101                 bsf             PIE1,RCIE       ; autorisation des inter usart
102                 banksel INTCON
103                 bsf             INTCON,PEIE     ; autorisation des inter périphériques
104                 bsf             INTCON,GIE      ; autorisation IRQ (interruption globale)
105
106 main    nop                                     ; boucle en attendant une inter
107                 goto    main
108
109 ;ici commence le programme complet
110 programme
111
112                 ; RECEPTION ;
113
114                 banksel ADCON0
115                 bsf     ADCON0,GO               ; demarrage de la conversion
116 non             BTFSC   ADCON0,GO               ; attendre la fin de conversion
```

```
117                 goto    non
118 oui             movfw   ADRESH
119                 movwf   valeur
120
121
122                 ; CONVERSION ;
123
124 ;valeur est entre 0-255, on la veut entre 0-5
125                 clrf    unite
126 divu    MOVLW   .51
127                 SUBWF   valeur,f                ; F-W
128                 btfsc   STATUS,C                ;skip si pas de reste
129                 incf    unite                   ; unite est incrémentée
130                 btfsc   STATUS,C                ;skip si pas de reste
131                 goto    divu
132
133                 movlw   .51
134                 ADDWF   valeur,f                ; on retrouve le "vrai" reste
135
136 ; on passe maintenant à la partie décimale
137 ; si valeur = 50, on va arrondir la partie décimale à 9
138                 movlw    .9
139                 MOVWF   decim                   ; on initialise la valeur à 9
140                 MOVLW   .50
141                 SUBWF   valeur,f;
142                 btfsc   STATUS,Z                ; voir p18
143                 goto    result                  ; si valeur = 50 goto résultat
144
145 ;si ce n'est pas le cas, on calcule la partie decimale
146
147                 movlw   .50
148                 ADDWF    valeur,f               ; on remet valeur à sa valeur
149                 clrf     decim
150 divd    MOVLW   .5
151
152
153                 SUBWF   valeur,f                ; F-W
154                 banksel STATUS
155                 btfsc   STATUS,C                ;skip si reste =0
156                 incf    decim                   ; partie decimale est incrémentée
157                 btfsc   STATUS,C                ;skip si reste=0
158                 goto    divd
159
160 result  nop
161
162 ; on a maintenant une partie entière 'unite' et une partie decimale 'decim'
163 ;pour les afficher en ascii il faut ajouter 30
164
165                 movlw   0x30
166                 ADDWF   unite,f
167                 MOVLW    0x30
168                 ADDWF    decim,f
169 ;On va contrôler l'affichage avec un timer, pour avoir une valeur par seconde
170
171                 ; AFFICHAGE ; ; ;
172
173
174
175
```

```
176                     banksel TXREG
177                     movfw   unite
178                     movwf   TXREG
179                     banksel TXSTA
180 att0    btfss   TXSTA, TRMT
181                     goto    att0
182                     banksel TXREG
183                     movlw   ","
184                     movwf   TXREG
185                     banksel TXSTA
186 att1    btfss   TXSTA, TRMT
187                     goto    att1
188                     BANKSEL TXREG
189                     movfw   decim
190                     movwf   TXREG
191                     banksel TXSTA
192 att2    btfss   TXSTA, TRMT
193                     goto    att2
194                     BANKSEL TXREG
195                     movlw   "V"
196                     movwf   TXREG
197                     banksel TXSTA
198 att3    btfss   TXSTA, TRMT
199                     goto    att3
200                     BANKSEL TXREG
201                     movlw   0x0D                    ; retour chariot
202                     movwf   TXREG
203                     banksel TXSTA
204 att4    btfss   TXSTA, TRMT
205                     goto    att4
206                     BANKSEL TXREG
207                     movlw   0x0A                    ; debut de ligne
208                     movwf   TXREG
209                     return
210
211
212
213                     end
```