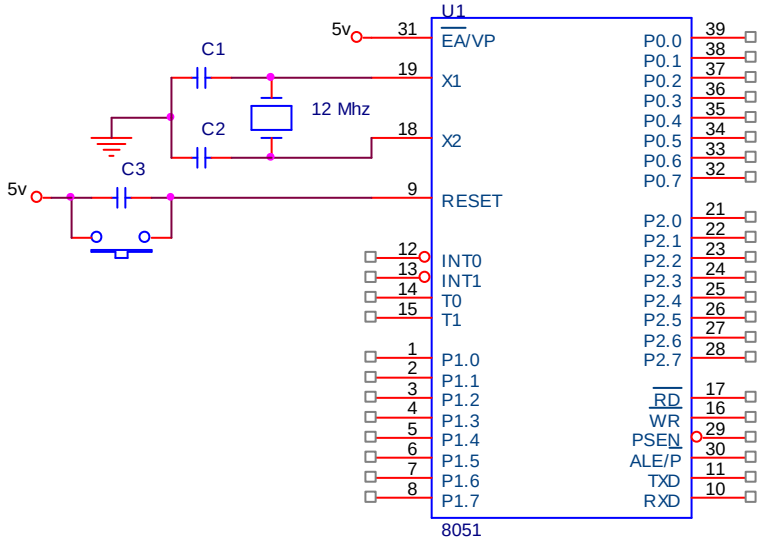
Eléments de mise en œuvre d'un fréquencemètre

ꕥ

I - Etape n°1 : câblage du 8051

Il s'agit de mettre en œuvre le 8051, sur une application simple de façon à permettre au étudiant de prendre en main les outils de développement et la base du hardware du microcontrôleur. Le schéma suivant est directement inspiré des recommandations de la documentation techniques du 8051.



Le programme consiste à complémenter le bit P3.0. La structure fonctionnelle peut être :

Le pseudo code :

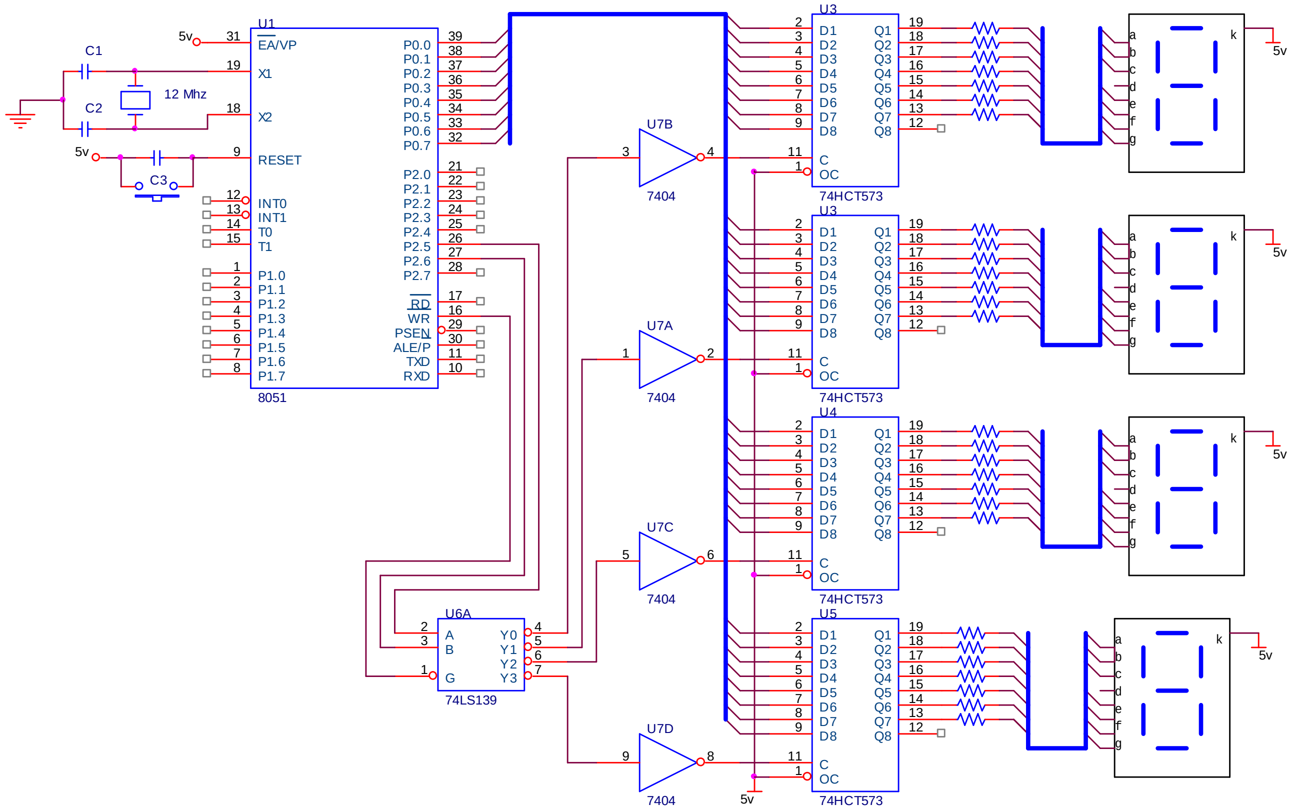
```
Début       Tant que(1)
                Faire  P3.0 <- P3.0/
            Fin Tant que
Fin
```

Le code 8051 est :

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;   Fréquencemetre : étape 1
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
#include <sfr51.inc>

    org 0000h
    ljmp debut       ; branchement au reset
    org 0030h

debut:
;   tant que (1) faire
          cpl P3.0        ; P3.0 <- P3.0/
          ljmp debut

        end
```

II - Etape n°2 : Mise en œuvre des afficheurs

Le montage se construit autour de LATCH type 74HCT573 qui mémorisent le contenue du bus de données lorsque le 8051 effectue une écriture (WR/ à 0) à l'adresse des afficheurs. Le décodage des adresses s'effectue à l'aide d'un DECODEUR DEMUX 2 vers 4 type 74LS139 qui décode les bits A14 et A15 du bus d'adresses.



Les afficheurs sont des afficheurs à anode commune pour des raisons de courant de sortie des LATCH. La résistance de protection est de l'ordre de 150Ω pour un courant de 20mA par segment.

Il y a d'autre solutions et des possibilités d'amélioration. Par exemple, on se servir de la broche laissé libre pour faire clignoter les afficheurs pour réduire le courant dans les segments.

Pour tester cette application, il suffit de faire clignoter chacun des afficheurs en affichant segment par segment. Pour cela, il est possible de mettre en œuvre une fonction dont la structure fonctionnelle est :

Il y a plusieurs méthodes pour faire afficher une valeur, la première consiste à écrire le code sept segments sur l'afficheur. Si pour allumer un segment, il faut un courant de 20mA, il faut 140mA par afficheur soit 560mA maximum. Ce courant peut être réduit en faisant clignoter l'afficheur. Cependant, cette technique provoque des images fantômes. Une autre méthode consiste à allumer qu'un segment à la fois par afficheur et de le faire tourner : segment a, puis b, puis c etc.. Pour faire afficher une valeur il suffit simplement de faire un masque entre le code sept segments et le segment affiché. Le courant absorbé par cette méthode est alors de 20mA par afficheur soit 80mA maximum. Le pseudo code de la fonction "Afficher" peut être :

```
Début
        segment<- 01h
        faire       @(afficheur) <- segment/ ou valeur
                    valeur <- valeur * 2
        tant que(segment<80h)
fin     fin tant que
```

Le programme principale est alors :

```
Début   valeur <- 00h  ; code sept segments pour un 8
    Tant que(1)
    Faire  afficher(0000h,valeur)
           afficher(2000h,valeur)
           afficher(4000h,valeur)
           afficher(6000h,valeur)
    Fin tant que
Fin
```

.

Le code assembleur est :

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;   Fréquencemetre : étape 2
;   gestion des afficheurs
;   auteur : Ph. Meyne
;   date : 12/10/2006
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
#include <sfr51.inc>

    org 0000h
    ljmp debut      ; branchement au reset
    org 0030h

debut:
;   fonction principale
        mov R0,#00h     ; affichage d'un 8
        ; afficheur 1
        mov DPTR,#0000h
        lcall afficher
        ; afficheur 2
        mov DPTR,#2000h
        lcall afficher
        ; afficheur 3
        mov DPTR,#4000h
        lcall afficher
        ; afficheur 4
        mov DPTR,#6000h
        lcall afficher
        ljmp debut

; fonction afficher
;    entree :   dptr adresse de l'afficheur
;               R0      valeur a afficher
;       table des variables
;           A segment
;           B variable intermédiaire
afficher:   mov A,#01h          ; segment <- 01h
faire:      mov B,A             ;  faire      sauvegarde de la valeur de A
            cpl A               ;       segment <- segment/
            orl A,R0            ;      segment <- segment|valeur
            movx @DPTR,A       ;      écriture à l'adresse des afficheur
            mov A,B             ;       restitution de la valeur initial de segment
            rl A                            rotation à droite
            cjne A,#80h,faire   ; tant que(segment != 80h)
            ret                        ; retour
```

III – Etape n°3 : Mise en œuvre de l'application

Le principe de la mesure est le suivant : un compteur compte le nombre de période pendant un temps fixé à 1s. Le nombre de période est alors égal à la fréquence du signal par définition. L'application peut être divisée trois grandes fonctions : une fonction de génération d'un signal de référence de période de l'ordre de 1s, la mesure en elle-même et le décodage du résultat de la mesure en digit codé 7 segments pour l'affichage.

3.1 – Génération d'un signal de période 1s

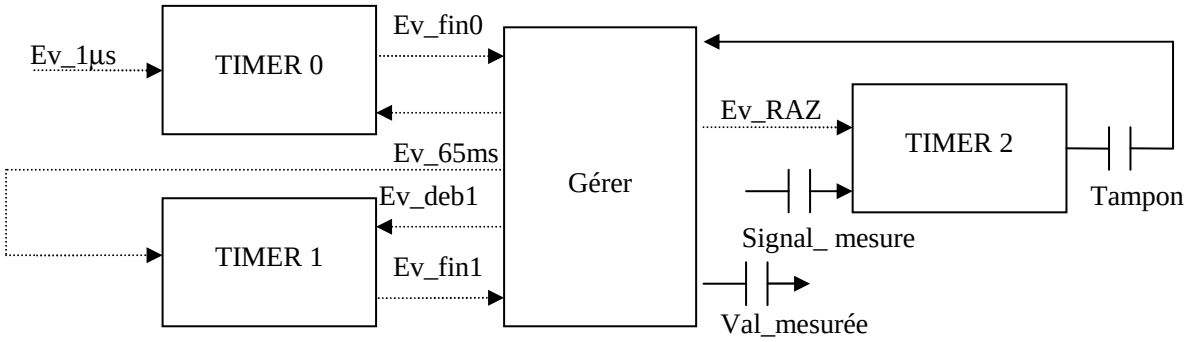
Il y a deux solutions pour produire ce signal : la première consiste à utiliser une horloge externe, la seconde d'utiliser les TIMER internes pour produire un temps de 1s. La première solution implique l'utilisation de condensateur et de résistance pour produire une horloge de grande période donc de cumuler les erreurs sur la valeur des

composants. En outre la stabilité des condensateurs et résistances dans le temps peut poser problème. Elle ne me semble donc pas très intéressante du point de vue de la précision de mesure. La seconde s'appuie du le quartz du 8051 ce qui me semble meilleure du point de vue de la précision de mesure.

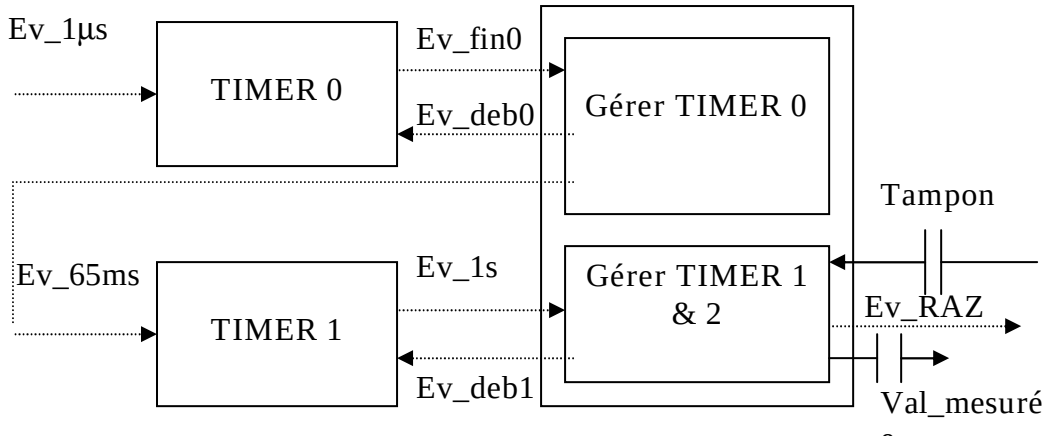
Si le 8051 a une horloge à 12 MHz alors le temps maximum que met un TIMER sur 16 bits pour compter est de 65,536ms. Un deuxième TIMER peut alors produire un temps d'une seconde s'il compte 65520 soit fff0h. Pour mettre en œuvre cette solution, il faut générer une horloge sur un bit d'un port par exemple P1.7 et l'utiliser comme horloge externe du second TIMER. C'est la solution TIMER que je propose de mettre en œuvre.

3.2 – Mesure de la fréquence

Il faut donc mettre en œuvre deux TIMER pour réaliser la base de temps et un TIMER qui pendant le temps de mesure compte les fronts du signal à mesurer. Une première structure fonctionnelle de cette application est donc la suivante :



La fonction "Gérer" peut être implantée comme une fonction d'interruption le problème est qu'elle est pilotée par deux interruptions. Il est possible de découper cette fonction en deux fonctions d'interruption. La structure fonctionnelle devient :



La fonction "Gérer TIMER 0" est activée par l'événement de fin de comptage issu du TIMER 0. Elle relance le comptage du TIMER 0 de façon à avoir un top toutes les 65,536ms permettant de générer une horloge de période 32,768ms . La fonction "Gérer TIMER 1 & 2" est activée par l'événement de fin de comptage du TIMER 1. Elle

mémorise la valeur du tampon du TIMER 2, les remet à zéro et relance le comptage. Le pseudo code de la fonction "Gérer TIMER 0" sur l'événement Ev_65ms est le suivant :

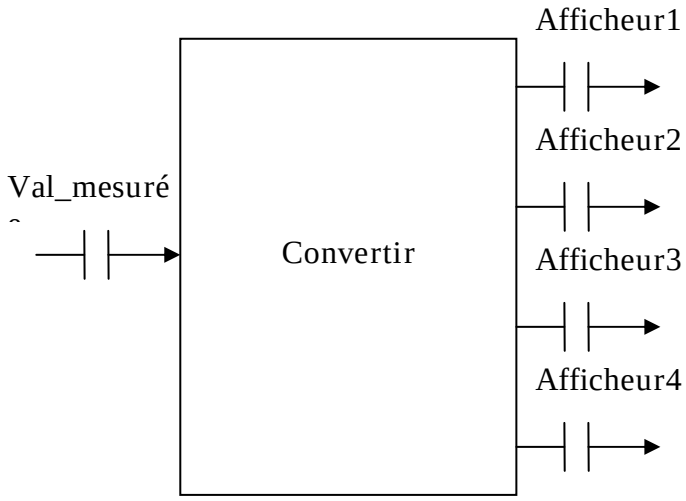
```
Début     sortie_horloge <- sortie_horloge/
          initialisation TIMER 1
          Lancer TIMER 1
Fin       retour
```

La variable sortie_horloge peut être le bit P1.7. Il faut alors relier cette sortie à l'entre d'horloge externe T1. Le pseudo code de la fonction "Gérer TIMER 1 & 2" sur l'événement Ev_1s est le suivant :

```
Début     initialisation TIMER 1
          Lancer TIMER 1
          Val_mesurée <- Tampon_TIMER2
          Tampon_TIMER2 <- 0
          Lancer TIMER2
Fin       retour
```

3.3 – Mesure de la fréquence

La variable "Val_mesurée" est codée sur 16 bits et représente une valeur de fréquence comprise en 0Hz et 2^{16}-1Hz. L'affichage sera limité à 9999Hz pour ne pas compliquer l'application. Il faut donc une fonction qui convertie le résultat hexadécimal en décimal puis en code sept segments et qui provoque l'affichage. La structure fonctionnelle de cette fonction est :



Le pseudo code de la fonction est :

```
Fonction : Convertir
Début Initialisation du système
      Tant que(1)
      Faire  Val_aff1 <- tab_conv(Val_mesurée&000fh,0)
             Afficher(0000h,Val_aff1)
             Val_aff2 <- tab_conv (Val_mesurée&00f0h>>4,retenue)
             Afficher(2000h,Val_aff2)
```

Val_aff3 <- tab_conv (Val_mesurée&0f00h>>8,retenue)
Afficher(4000h,Val_aff3)
Val_aff4 <- tab_conv (Val_mesurée&f000h>>12,0)
Afficher(6000h,Val_aff4)
Fin tant que
Fin

Cette fonction est naturellement la fonction principale de notre système. Elle est donc structurée autour d'une boucle sans fin qui est précédée d'une phase d'initialisation du système.

Remarque : je n'ai pas trouvé de méthode simple pour réaliser l'affichage en décimal, l'affichage s'effectue donc en hexadécimal.

3.4 – Code

Je n'ai pas simulé le code suivant. Cela me semble difficile.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;   Fréquencemetre : étape 3
;   codage de l'application
;   auteur : Ph. Meyne
;   date : 12/10/2006
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
#include <sfr51.inc>

; registre TIMER 2 du 8051
CT2     bit 0xC9    ; choix horloge externe TIMER 2
TR2     bit 0xCA    ; lancement TR2
EXN2    bit 0xCB    ; choix externe mode capture
TF2     bit 0xCF    ; flag de fin de comptage
TL2     equ 0xCC    ; adresse tampon TIMER 2 Pf
TH2     equ 0xCD    ; adresse tampon TIMER 2 PF

    org 0000h
    ljmp debut           ; branchement au reset
    org 000bh
    ljmp gerer_TIMER0    ; branchement sur IT TIMER 0
    org 001Bh
    ljmp gerer_TIMER1    ; branchement sur IT TIMER 1

    org 0030h

debut:
; initialisation du systeme
    ; initialisation des interrruptions
    setb EA      ; autorisation des IT
    setb ET0     ; autorisation de l'IT TIMER 0
    setb ET1     ; autorisation de l'IT TIMER 1
    ; initialisation des TIMER 0 et 1
    mov TMOD,#51h    ; intialisation des TIMER en monostable 16 bits
                     ; TIMER à partir d'un hrologe externe
    mov TH0,#00h     ; comptage 65ms
    mov TL0,#00h
    mov TH1,#FFh     ; comptage de 1 seconde à partir de l'horloge P1.7 à
32,768ms
    mov TL1,#E1h     ;
```

```
    clr TF0
    clr TF1
    ; initialisation TIMER 2
    setb CT2        ; horloge externe T2
    clr EXN2        ; mode capture
    clr TF2
    ; lancement des IT
    setb TR0
    setb TR1
    setb TR2
;   fonction principale
;       table des variables
;       Val-mesurée PF  : R1
;       Val-mesurée Pf  : R2

        mov R1,#00h     ; affichage d'un 0 avant la première mesure
        mov R2,#00h
        ; afficheur 1
TQ:     ; affichage poids faible faible
        mov A,R1                        ; digit <- PF Val_mesurée
        anl A,#0fh                      ; isolation du poids le plus faible
        mov DPTR,#tab_conv              ; digit <- tab_conv(digit)
        movc A,@A+DPTR
        mov R0,A
        mov DPTR,#0000h                 ; adresse afficheur 1
        lcall afficher                  ; affichage
        ; affichage poids faibles fort
        mov A,R1                        ; digit <- PF Val_mesurée
        anl A,#f0h                      ; isolation du poids fort
        swap A                          ; echange poids faible poids fort
        mov DPTR,#tab_conv              ; digit <- tab_conv(digit)
        movc A,@A+DPTR
        mov R0,A
        mov DPTR,#2000h                 ; adresse afficheur 2
        lcall afficher                  ; affichage
        ; affichage poids fort faible
        mov A,R2                        ; digit <- PF Val_mesurée
        anl A,#0fh                      ; isolation du poids faible de R2
        mov DPTR,#tab_conv              ; digit <- tab_conv(digit)
        movc A,@A+DPTR
        mov R0,A
        mov DPTR,#4000h                 ; adresse afficheur 3
        lcall afficher                  ; affichage
        ; affichage poids fort fort
        mov A,R2                        ; digit <- PF Val_mesurée
        anl A,#f0h                      ; isolation du poids fort
        swap A                          ; echange poids faible poids fort
        mov DPTR,#tab_conv              ; digit <- tab_conv(digit)
        movc A,@A+DPTR
        mov R0,A
        mov DPTR,#6000h                 ; adresse afficheur 2
        lcall afficher                  ; affichage
        ljmp TQ

; fonction Afficher
;    entree :   adresse de l'afficheur  dptr
;               valeur a afficher       R0
;       table des variables
;           A segment
;           B variable intermédiaire
afficher:   mov A,#01h          ; segment <- 01h
```

```
faire:      mov B,A             ; faire sauvegarde de la valeur de A
            cpl A               ;       segment <- segment/
            orl A,R0            ;       segment <- segment|valeur
            movx @DPTR,A        ;       écriture à l'adresse des afficheur
            mov A,B             ;       restitution de la valeur initial de
segment
            rl A                ;       rotation à droite
            cjne A,#80h,faire   ; tant que(segment != 80h)
            ret                 ; retour

; fonction Gérer TIMER 0
;    entree :   néant
;    sortie :   EV_65ms P1.7

gerer_TIMER0:   clr TR0         ; arrêt TIMER 0
                clr TF0         ; mise à zéro du flag de fin de comptage
                cpl P1.7        ; complémentation de la sortie horloge
                mov TH0,#00h    ; comptage 65ms
                mov TL0,#00h    ;
                setb TR0        ; lancement TIMER 0
                RETI

; fonction Gérer TIMER 1
;    entree :   Tampon  TH2,TL2
;    sortie :   Ev-RAZ  TR2
;               Val_mesuree A
gerer_TIMER1:   clr TR1             ; arrêt TIMER 1
                clr TF1             ; mise à zéro du flag de fin de
comptage
                clr TR2             ; arrêt comptage TIMER 2
                clr TF2             ; mise à zéro du flag de fin de
comptage
                mov R1,TL2          ; Val_mesurée pf <- tampon TIMER 2
                mov R2,TH2          ; Val_mesurée pF <- tampon TIMER 2
                ; lancement TIMER 1
                mov TH1,#FFh        ; comptage de 1 seconde à partir de
l'horloge P1.7 à 32,768ms
                mov TL1,#E1h        ;
                setb TR1            ; lancement TIMER 1
                ; comptage TIMER 2
                mov TH2,#00h        ; mise à zéro du tampon TIMER 2
                mov TL2,#00h        ;
                setb TR2            ; lancement TIMER 2
                RETI

; table de transcodage
tab_conv:   db 40h      ; code du 0
            db 7Ch      ; code du 1
            db 24h      ; code du 2
            db 50h      ; code du 3
            db 0Ch      ; code du 4
            db 1Ah      ; code du 5
            db 0Ah      ; code du 6
            db 58h      ; code du 7
            db 00h      ; code du 8
            db 10h      ; code du 9
        end
```