SYNTHÈSE DES FONCTIONS SINUS SUR MICROCONTRÔLEURS

Les programmeurs en assembleur sont souvent désorientés quand il devient nécessaire d'utiliser des fonctions trigonométriques. Ils utiliseront alors souvent des langages de haut niveau, BASIC ou C disposant de ces fonctions pour éviter de les programmer eux-même. Ces langages permettent de programmer de manière efficace mais, outre l'investissement nécessaire, l'utilisation de fonctions trigonométriques impliquera alors presque nécessairement une programmation utilisant des nombres flottants, généralement coûteuse en temps de calcul et en taille mémoire.

```
; Les fonctions _sin16 et _cos16 calculent le sinus et le cosinus d'un angle
; R6:R7. Un angle de 0° est représenté par 0, 90° par 4000H, 180° par 8000H ...
; la précision sur les angles est donc de 360°/65536 = 0.0055°
; le résultat est 2^14 X sin (x) (ou cos x), la valeur +1 est codée 4000H la
; la valeur -1 est codée -4000H = C000H. L'erreur maximum sur le résultat
; est due aux troncatures est de 1 LSB soit de 1/16384 = 0.000061
_sin16:   MOV     A,R6            ; calcule le sinus a partir du cosinus
          ADD     A,#0C0H         ; sin(x) = cos(x - 90°) = cos(x + 270°)
          MOV     R6,A            ; 270° = C000H
_cos16:   MOV     A,R6            ; teste si angle négatif
          JNB     ACC.7,cos0
; angle négatif, cosinus(-x) = cosinus(x)
          CALL    _neg16          ; attention -8000H = 8000H
          JNB     ACC.7,cos0      ; cas général, différent de 180°
          MOV     R6,#0C0H        ; R6:R7 = 8000H, retourne C000H soit -1
          RET
; dans R6:R7, un angle de 0 à 7FFFH (0 à 180° exclu)
; A = R6 le MSB de l'angle
cos0:     JB      ACC.6,cosneg    ; angle >= 90°, cosinus négatif
          JNB     ACC.5,cos45     ; angle < 45° traite directement
; angle de 45 à 90 cos(x) = sin(x - 90°)
          CALL    _neg16          ; calcule -x
          MOV     A,R6            ; puis 4000H - x, en degres : 90° - x
          ADD     A,#40H
          MOV     R6,A
          AJMP    sin45           ; cos(x) = sin(90 - x)
; cosinus négatif, calcule le cosinus positif et inverse
cosneg:   JB      ACC.5,c135      ; angle > 135°
; angle de 90 à 135 cos(x) = - sin(x - 90)
          ANL     A,#1FH          ; en fait x - 2000H, en degrés : x - 90°
          MOV     R6,A
          ACALL   sin45
          AJMP    _neg16          ; et inverse le résultat
; angle de 135 à 180° cos(x) = -cos(180° - x)
c135:     CALL    _neg16
          MOV     A,R6            ; puis 8000H - x, en degrés (180° - x)
          ADD     A,#80H
          MOV     R6,A
          CALL    cos45
          AJMP    _neg16
```

■ *Listing 1*

Sur des microcontrôleurs équipés de peu de mémoire (cas des 87C75X ou même des banaux 80C51), il faudra recourir à une programmation en assembleur avec des calculs entiers.
Une solution est de recourir à l'utilisation de tables dans lesquelles sont stockées les valeurs des différents sinus. Cette solution largement utilisée quand on ne désire qu'une résolution de 8 bits, est inapplicable pour des résolutions de 16 bits ou plus.
Il faut donc recourir aux calculs des sinus en utilisant des développements en série limités (1) et (2). En ce qui concerne le sinus (ou le cosinus), ces séries convergent relativement vite et sont donc directement exploitables moyennant quelques aménagements. En effet, ceux qui désireraient programmer ces séries telles se heurteront au calcul des factorielles qui dépassent très rapidement la taille des entiers. Aussi, on améliorera le calcul en exprimant chaque terme en fonction du précédant, éliminant ainsi le délicat problème du calcul de la factorielle (3), ensuite une factorisation adéquate permettra de limiter le nombre de multiplications (4) et (5). La question suivante qui se pose est «combien de termes sont nécessaires pour mon application ?». Chaque terme ajouté dans le développement améliore la précision du résultat. Par ailleurs, la va-

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \dots \quad (1)$$

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!} + \frac{x^8}{8!} - \dots \quad (2)$$

$$T(n+1) = \frac{x^2 T(n)}{2n(2n+1)} \quad (3)$$

$$\sin(x) = x\left(1 - \frac{x^2}{2\times3}\left(1 - \frac{x^2}{4\times5}\left(1 - \frac{x^2}{6\times7}\left(1 - \frac{x^2}{8\times9}\left(1 - \dots\right)\right)\right)\right)\right) \quad (4)$$

$$\cos(x) = 1 - \frac{x^2}{2}\left(1 - \frac{x^2}{12}\left(1 - \frac{x^2}{30}\left(1 - \frac{x^2}{56}\left(1 - \dots\right)\right)\right)\right) \quad (5)$$

Bits	90°(sin)	45°(sin)	90°(cos)	45°(cos)
8	4	2	4	4
16	5	4	6	4
24	7	5	7	5
40	9	7	9	7
64	12	10	13	10

Termes nécessaires pour les fonctions sinus/cosinus

■ *Tableau 1*

leur calculée oscille autour de la valeur exacte, ce qui veut dire que la taille du dernier terme doit être inférieure à la précision demandée. Inversement, si le nième terme de la série est inférieur à la précision désirée, alors une série de n termes correspondra à notre critère de précision (**tableau 1**).

Une autre manière d'augmenter la précision est de diminuer l'étendue dans laquelle on va calculer la fonction. En effet, les développements en série sont d'autant plus précis que l'on est proche de zéro. On voit tout de suite que grâce aux symétries des fonctions sinusoïdales il est inutile de calculer le sinus pour des angles de plus de 90° ($\pi/2$). De même, on pourra restreindre le calcul à des angles de 45° en calculant soit la série sinus, soit la série cosinus, suivant l'angle à évaluer. Si les fonctions sont calculées avec des entiers, il faudra mettre à l'échelle les valeurs des angles et les valeurs de sortie des fonctions. Les angles pourront être simplement codés en prenant 0 pour 0° ou 360°, 2000H pour 45° ... 8000H pour 180°, C000H pour 270° ou -90°. On aura donc une résolution sur les angles de 360°/65536 = 0.00549°. Pour ce qui est de la valeur retournée par la fonction, celle-ci est comprise entre - 1 et +1. Si on accepte de perdre un bit de précision, il sera commode de coder la valeur +1 avec 4000H et la valeur -1 avec C000H ; la fonction calculée ne sera donc pas sin(x) mais 16384 X sin(x). Tous ces changements d'échelle ont bien sûr une répercussion sur la valeur des coefficients (6) et (7). Si on tient compte du fait qu'on utilise une multiplication 16 X 16 avec un résultat dont on ne garde que le MSB, les nouveaux coefficients sont donnés en (8) et (9). Le code source est disponible sur le serveur ERP ainsi que le code de la multiplication 16 X 16 (_mul16).

J. L. VERN

Application pour calcul entier avec une résolution de 16 bits :

$$16384 \cdot \sin(x) = u \otimes (s_1 - v \otimes (s_2 - v \otimes s_3)) \quad (6)$$

$$16384 \cdot \cos(x) = c_0 - v \otimes (c_1 - v \otimes (c_2 - v \otimes c_3)) \quad (7)$$

$$\text{avec: } a \otimes b = \frac{a \times b}{65536}, \; u = x \cdot 2, \; v = (u \times 2) \otimes (u \times 2)$$

s_1 = C910H, s_2 = 52A7H, s_3 = 09F1H (8)

Les cœfficients s_1, s_2 et s_3 ont été modifiés en utilisant les polynômes de Chebyshev pour minimiser l'erreur maximum.

c_0 = 4000H, c_1 = 4EF5H, c_2 = 103CH, c_3 = 0156CH (9)

$$c_1 = 2^{14}, \; c_1 = 2^{12} \times \left(\frac{\pi^2}{2!}\right), \; c_2 = 2^{10}, \times \left(\frac{\pi^4}{4!}\right), \; c_3 = 2^8 \times \left(\frac{\pi^6}{6!}\right)$$

```
; calcule l'inverse de R6:R7
; attention, -8000H = 8000H
_neg16:  CLR     A              ; calcule -x
         CLR     C
         SUBB    A,R7           ; -x = 0 - x, plus rapide que complement + 1
         MOV     R7,A
         CLR     A
         SUBB    A,R6
         MOV     R6,A
         RET
; multiplie R6:R7 par deux
_mul2:   MOV     A,R7           ; plus rapide qu'un décalage (pas de CLR C)
         ADD     A,R7
         MOV     R7,A
         MOV     A,R6
         ADDC    A,R6
         MOV     R6,A
         RET
; effectue le calcul K - (x^2 X R6:R7). K est la constante pointée par le DPTR
; pour des facilités de calcul, les constantes sont stockées LSB:MSB
; table des coefficients cosinus
C3       EQU     0156H          ; 2^8  X (PI^6/6!)
C2:      DB      3CH            ; 2^10 X (PI^4/4!)
         DB      10H
         DB      0F5H           ; 2^12 X (PI^2/2!)
         DB      4EH
         DB      00H            ; 2^14 X (1.0)
         DB      40H
; table des coefficients des sinus
S3       EQU     09F1H          ; approx. 2^10 X PI^5/5! (2.485336730)
S2:      DB      0A7H           ; approx. 2^12 X PI^3/3! (5.165694407)
         DB      52H
         DB      10H            ; approx. 2^14 X PI    (3.141576918)
         DB      0C9H
k_ax2:   MOV     A,R2           ; restaure x^2
         MOV     R4,A
         MOV     A,R3
         MOV     R5,A
         call    _mul16         ; calcule x^2 X R6:R7 (C=0 en sortie de _mul16)
         CLR     A              ; prend la constante suivante
         MOVC    A,@A+DPTR      ; commence par le LSB
         INC     DPTR
         SUBB    A,R5           ; et soustrait le résultat de la multiplication
         MOV     R7,A
         CLR     A
         MOVC    A,@A+DPTR      ; puis MSB
         INC     DPTR
         SUBB    A,R4
         MOV     R6,A
         RET                    ; sort avec R6:R7 = @DPTR - (x^2 X R6:R7)
; calcule le carré de R6:R7 (en fait le cadrage est tel que le calcul effectué
; est R6:R7 X R6:R7 / 2^14 = (R6:R7)^2 / 16384
calc_x2:
         CALL    _mul2          ; de 0 a 8000H
         MOV     A,R6
         MOV     R4,A
         MOV     A,R7           ; R4:R5 = R6:R7
         MOV     R5,A
         CALL    _mul16         ; résultat de 0 a 4000H
         MOV     A,R4
         MOV     R2,A
         MOV     A,R5           ; sauve x^2 dans R2:R3
         MOV     R3,A
         RET
; calcule le sinus d'un angle R6:R7 de 0 à 2000H
; résultat dans R6:R7 = 4000H X sin(2 X PI X R6:R7 / 10000H)
sin45:   CALL    _mul2          ; maintenant x de 0 a 4000H
         PUSH    AR6            ; sauve x
         PUSH    AR7
         ACALL   calc_x2        ; en sortie, x^2 cadre de 0 a 4000H dans R2:R3
         MOV     R6,#HIGH S3    ; dernière constante sinus
         MOV     R7,#LOW S3
         MOV     DPTR,#S2       ; pointe sur les constantes sinus
         ACALL   k_ax2          ; calcule M2 = S2 - (x^2 X S3)
         ACALL   k_ax2          ; calcule M1 = S1 - (x^2 X M2)
         POP     AR5            ; restaure x
         POP     AR4
         CALL    _mul16         ; calcule sin = x X M1
         MOV     A,R4
         MOV     R6,A
         MOV     A,R5
         MOV     R7,A
         RET                    ; résultat dans R6:R7
; calcule le cosinus d'un angle de 0 à 2000H
; résultat dans R6:R7 = 4000H X cos(2 X PI X R6:R7 / 10000H)
cos45:   CALL    _mul2          ; de 0 a 4000H
         CALL    calc_x2
         MOV     R6,#HIGH C3
         MOV     R7,#LOW C3
         MOV     DPTR,#C2       ; pointeur sur les constantes cosinus
         ACALL   k_ax2          ; calcule M2 = C2 - (x^2 X C3)
         ACALL   k_ax2          ; calcule M1 = C1 - (x^2 X M2)
         AJMP    k_ax2          ; calcule cos = C0 - (x^2 X M1)
```

■ *Listing 2*