SYNTHÈSE DU LOGARITHME SUR µC

Le logarithme est une fonction très utilisée dès lors que l'on traite des données physiques. Qu'il s'agisse de mesure d'intensité sonore, lumineuse, ou de calcul de temps d'intégration dans une capacité, on utilisera les fonctions ln(x) (logarithme népérien) ou Log(x) (logarithme décimal) pour mener à bien les calculs.

```
; Calcule le ln de R4:R5:R6:R7. R4:R5 est la partie entière, R6:R7 la partie
; fractionaire. La valeur 1.0 par ex est codée 00010000H, la valeur 3.0 est
; codée 00030000H, la valeur 3.5 00038000H, etc. La valeur maximum est
; 65535,9999847 codée FFFFFFFFH. La valeur maximum est 1/65536 et codée 1H.
; Le résultat est signé et est renvoyé dans R4:R5:R6:R7 avec R4:R5 partie
; entiére et R6:R7 partie fractionnaire. L'erreur due a l'approximation est
; de l'ordre de 1/10000.
LN2     EQU     45426           ; ln(2) X 10000H = 0.69314718056 X 65536

_ln:                            ; exprime x sous forme 2^n * r
                        ; ou 0.5 =< r < 1
        MOV             R0,#0           ; n = 0
        CLR     F0              ; à priori n > 0 (x > 1)
        MOV             A,R4
        ORL             A,R5
        JNZ     supa1           ; si R4:R5 <> 0, valeur >= 1
infer:  MOV             A,R6
        JB      ACC.7,infa1     ; si dernier bit a 1, 0.5 =< x < 1
        CALL    _mul2           ; sinon, multiplie par deux
        INC     R0              ; n = n+1       (valeur negative)
        SETB            F0              ; et marque négatif (si n = 0, F0 = 0)
        AJMP    infer           ; jusqu'a cadrer la valeur entre 0.5 et 1
supa1:  CALL            _div2           ; divise par deux
        INC     R0              ; n = n+1
        MOV             A,R4
        ORL             A,R5            ; teste si >= 1
        JNZ     supa1           ; oui, divise encore
infa1:  MOV             A,R0
        PUSH            ACC             ; sauve n
        CALL            _ln05a1         ; calcule -ln(r) (r entre 0.5 et 1)
        POP             ACC
        MOV             R4,#0
        MOV             R5,A            ; récupère n
        MOV             R6,#HIGH LN2            ; partie haute de ln(2)
        MOV             R7,#LOW LN2             ; partie basse de ln(2)
        CALL            _mul16          ; résultat dans R4:R5:R6:R7
        JNB     F0,pos
        CALL            _neg            ; n négatif, inverse le résultat
pos:    CLR     C
        MOV             A,R7
        SUBB            A,R3            ; soustrait -ln(r) pour avoir
        MOV             R7,A            ; ln(x) = ln(r) + ln(2) X n
        MOV             A,R6            ; si x = r X 2^n
        SUBB            A,R2
        MOV             R6,A
        MOV             A,R5
        SUBB            A,#0
        MOV             R5,A            ; le msb (R4) n'est pas affecté (00 ou FF)
        RET                             ; retour avec ln(x) = R4:R5:R6:R7
```

■ *Listing 2*

Nous l'avons vu lors des précédents articles sur le calcul des sinus et des arctangentes, les développements limités sont une bonne approche pour calculer certaines fonctions. Dans le cas du logarithme, le développement limité le plus connu est le développement (1) qui permet de calculer le logarithme népérien de 1+x. En faisant un changement de variable, on obtient le développement (2) qui permet de calculer directement le logarithme népérien de x. Hélas comme dans le cas de la fonction arctangente, ces développements convergent très lentement, aussi comme dans le cas précédent, on cherchera d'autres types de développements.

Ordre	1	3	5	7	9	11
Intervalle 0,5 à 1	2,6e-2	8,4e-4	2,6e-5	7,6e-7	2,3e-8	7e-10
Intervalle 0,707 à 1	3,4e-3	2,7e-5	2,1e-7	1,8e-9	0	

■ *Tableau 2*

Ordre	1	3	5
Constantes pour 0,5 à 1	a=2,0592326826	a=1,9992286236 b=0,3465082807	a=2,0000081491 b=0,2810013053 c=0,6140568850
Erreur de 0,5 à 1	0,0067	5,3e-5	4,0e-7
Constantes pour 0,707 à 1	a=2,0149618989	a=1,9999503980 b=0,3366590157	a=2,0000001333 b=0,2702649321 c=0,6035805051
Erreur de 0,707 à 1	0,00086	1,7e-6	3,3e-9

■ *Tableau 3*

(1) $\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots$

(2) $\ln(x) = 2\left(z + \frac{z^3}{3} + \frac{z^5}{5} + \frac{z^7}{7} + \ldots\right)$ avec $z = \frac{x-1}{x+1}$

(3) $$\ln(x) = \cfrac{2z}{1 - \cfrac{z^2}{3\left(1 - \cfrac{4z^2}{3*5\left(1 - \cfrac{9z^2}{5*7\left(1 - \cfrac{16z^2}{7*9\left(1 - \cfrac{25z^2}{9*11(1 - \ldots)}\right)}\right)}\right)}\right)}}$$

(4) $\mathrm{Log}_a(x) = \frac{\ln(x)}{\ln(a)}$

(5) $\ln(x) = (\ln(2) * n) + \ln(r)$ avec $x = 2^n * r$

Ordre 1 : $\ln(x) \approx 2z$

Ordre 3 : $\ln(x) \approx \frac{6z}{3 - z^2} = \frac{az}{1 - bz^2} = \frac{x^2 - 1}{x^2\left(\frac{1-b}{a}\right) + 2x\left(\frac{1+b}{a}\right) + \left(\frac{1-b}{a}\right)}$

Ordre 5 : $\ln(x) \approx \frac{2z(15 - 4z^2)}{15 - 9z^2} = \frac{az(1 - bz^2)}{1 - cz^2}$

Ordre 7 : $\ln(x) \approx \frac{2z(105 - 55z^2)}{105 - 90z^2 + 9z^4}$

Ordre 9 = $\ln(x) \approx \frac{2z(945 - 735z^2 + 64z^4)}{945 - 1050z^2 + 225z^4}$

Ordre 11 : $\ln(x) \approx \frac{2z(1155 - 1190z^2 + 231z^4)}{1155 - 1575z^2 + 525z^4 - 25z^6}$

■ *Tableau 1*

L'équation (3) exprime le logarithme népérien sous forme de fraction continue. Comme c'est souvent le cas quand les développements limités convergent lentement, le développement correspondant en fraction continue converge rapidement. En arrêtant le calcul a l'ordre 1 on obtiendra une approximation simple de ln(x) qui sera simplement égal a 2(x-1)/(x+1). En s'arrêtant à un ordre plus élevé, on pourra simplifier l'équation obtenue de manière a obtenir une fonction qui sera un rapport de polynômes. On trouvera dans le **tableau 1** le résultat des calculs pour les ordres de 1 à 11. On remarquera que ces développements sont optimaux pour des valeurs de z proches de zéro, ce qui correspond à des valeurs de x proches de 1. Comme toujours dans des cas semblables, la précision sera d'autant meilleure que l'on sera proche de 0. Si on exprime x sous la forme $r2^n$, l'équation (5) nous permettra de réduire l'intervalle du calcul du logarithme. Au lieu de calculer le logarithme de x pour toutes les valeurs positives, il faut maintenant calculer le logarithme de r pour des valeurs de 0,5 à 1.

On pourra réduire encore l'intervalle avec l'égalité suivante :

$x = r2^{(n/2)}$

Si n est pair, il n'y a pas de changement par rapport au calcul précédant, si n est impair il faudra simplement multiplier par racine de 2, en réduisant l'intervalle de calcul du logarithme de 0,70711.. à 1.

Le **tableau 2** indique les erreurs obtenues suivant les ordres choisis et l'intervalle d'entrée choisi. On peut améliorer la précision de manière importante en modifiant légèrement les constantes utilisées dans les calculs.

Le **tableau 3** indique les précisions obtenues pour les ordres 1, 3, et 5 pour les deux intervalles d'entrée 0,5 à 1 et 0,707 à 1 ainsi que les constantes «améliorées».

Le **listing 1** réalise le calcul du logarithme népérien pour des valeurs variant de 0,5 à 1 en utilisant une approximation d'ordre 3. Le **listing 2** utilise la fonction précédante pour calculer le logarithme pour une valeur variant de 65535 à 1/65536. La valeur d'entrée est simplement décalée (dans un sens ou dans l'autre) n fois pour faire cadrer la valeur dans l'intervalle 0,5 à 1. Une fois cadrée, le logarithme de la valeur est calculé puis additionné à n fois ln(2) ainsi qu'indiqué dans l'équation (5). La valeur de ln(x) renvoyée dans R4:R5:R6:R7 est une valeur signée. La partie entière est dans R4:R5, la partie fractionnaire dans R6:R7. Les lecteurs qui désirent effectuer les calculs en flottant pourront bénéficier de la représentation interne des nombres flottants dans laquelle la mantisse est déja cadrée dans un intervalle de 0,5 à 1 et l'exposant disponible pour le calcul sans nécessiter de décalages.

Les fonctions _mul16 et _div32 ont été décrites dans le numéro 556 d'Electronique Radio Plan et sont disponibles ainsi que les fonctions présentes sur le serveur ERP.

J-L VERN

```
K1   EQU      21422              ; 65536 X (1-b)/a
K2   EQU      44139              ; 65536 X (1+b)/a
;K3  EQU      1403904251         ; 65536 X K1 (non arrondi)

; _negACCU calcule - ACCU32.
; ici la valeur 1 - ACCU32 est codée de la même manière que -ACCU32
_negACCU:
     CLR      C
     CLR      A
     SUBB     A,ACCU32+3
     MOV      ACCU32+3,A
     CLR      A
     SUBB     A,ACCU32+2
              MOV          ACCU32+2,A
              CLR          A
              SUBB         A,ACCU32+1
              MOV          ACCU32+1,A
              CLR          A
              SUBB         A,ACCU32+0
              MOV          ACCU32+0,A
              RET

; inverse R4:R5:R6:R7
_neg:
              CLR          C
              CLR          A
              SUBB         A,R7
              MOV          R7,A
              CLR          A
              SUBB         A,R6
              MOV          R6,A
              CLR          A
              SUBB         A,R5
              MOV          R5,A
              CLR          A
              SUBB         A,R4
              MOV          R4,A
              RET

; divise la valeur 32 bits R4:R5:R6:R7 par deux
_div2:  CLR   C
              MOV          A,R4
              RRC          A
              MOV          R4,A
              MOV          A,R5
              RRC          A
              MOV          R5,A
              MOV          A,R6
              RRC          A
              MOV          R6,A
              MOV          A,R7
              RRC          A
              MOV          R7,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

; calcule l'oppose du logarithme néperien de R6:R7 pour une valeur variant de
; 0.5 à 1 (1 exclu). La valeur 0.5 est codée 8000H la valeur 1 codée 10000H.
; Le résultat de l'opération se trouve dans R2:R3
; Par exemple pour calculer ln(0.7) on mettra dans R6:R7 la valeur :
; 0,7 X 65536 = 45875 (0B333H) : R6 = B3, R7 = 33.
; le résultat de -ln(0,7) est 0,35667494 la valeur retournée dans R2:R3 sera
; 0,35667494 X 65536 soit 23375 soit 5B4F (la valeur réelle retournée est 5B4E)
; ln(x) est approché par (x^2 - 1)/((x^2(1-b)/a) + (x(1+b)/a) + (1-b)/a)
; avec a = 1.9992286236 et b = 0.3465082807. Les calculs sont effectuées
; de manière à minimiser les effets des troncatures dus aux calculs entiers.
_ln05a1:
              MOV          A,R6        ; sauve R6:R7 dans R2:R3
              MOV          R4,A        ; et dans R4:R5 (pour calculer le carré)
              MOV          R2,A
              MOV          A,R7
              MOV          R5,A
              MOV          R3,A
              CALL         _mul16            ; calcule le carré de R6:R7
              MOV          ACCU32+0,R4       ; sauve le résultat (32 bits) dans ACCU32
              MOV          ACCU32+1,R5
              MOV          ACCU32+2,R6
              MOV          ACCU32+3,R7
              MOV          R6,#HIGH K1
              MOV          R7,#LOW K1                ; R6:R7 = K1
              CALL         _mul16            ; calcule x^2 X (1-b)/a
              MOV          A,R7              ; puis additionne (1-b)/a
              ADD          A,#0FBH           ; 1403904251 = 53ADE0FBH
              MOV          R7,A
              MOV          A,R6
              ADDC         A,#0E0H
              MOV          R6,A
              MOV          A,R5
              ADDC         A,#0ADH
              MOV          R5,A
              MOV          A,R4
              ADDC         A,#053H
              MOV          R4,A
              CALL         _div2             ; divise par deux
              MOV          A,R2              ; récupère x
              XCH          A,R4
              MOV          DPH,A             ; en sauvant le résultat précédant
              MOV          A,R3
              XCH          A,R5
              MOV          DPL,A             ; partie haute du résultat sauvée dans DPTR
              MOV          A,R6              ; et partie basse dans R2:R3
              MOV          R2,A
              MOV          A,R7
              MOV          R3,A
              MOV          R6,#HIGH K2               ; R6:R7 = K2
              MOV          R7,#LOW K2
              CAL          _mul16            ; calcule x X (1+b)/a dans R4:R5:R6:R7
              MOV          A,R7              ; et additionne au résultat précédant
              ADD          A,R3              ; (contenu dans DPH:DPL:R2:R3)
              MOV          A,R6              ; propage la retenue des LSB
              ADDC         A,R2
              MOV          A,R5
              ADDC         A,DPL
              MOV          R7,A              ; et garde que les 16 (ou 17) bits forts
              MOV          A,R4              ; (en fait divise le résultat par 10000H
              ADDC         A,DPH
              MOV          R6,A
              CLR          A
              MOV          R4,A
              ADDC         A,#0
              MOV          R5,A              ; si retenue, 17 ieme bit
              CALL         _negACCU                ; calcule 1-x^2 (x^2 dans ACCU32)
              CALL         _div32            ; puis calcule le quotient des deux polynomes
              MOV          A,ACCU32+1
              RRC          A
              MOV          A,ACCU32+2
              RRC          A
              MOV          R2,A
              MOV          A,ACCU32+3
              RRC          A
              MOV          R3,A
              RET
```

■ ***Listing 1***

é RADIO PLANS